Solution Manual for: Introduction to ALGORITHMS (Second Edition) by T. Cormen, C. Leiserson, and R. Rivest

John L. Weatherwax*

March 13, 2021

*wax@alum.mit.edu

To my family.

Acknowledgments

Special thanks to (most recent comments are listed first): Omran Khoja and Avi Cohenal for finding typos in these notes. All comments were (and are) much appreciated.

Chapter 1 (The Role of Algorithms in Computing)

1.1 (Algorithms)

Exercise 1.1-1 (sorting, optimally multiply matrices, and convex hulls)

Sorting is done in all sorts of computational problems. It is especially helpful with regard to keeping data in a understood ordering so that other algorithms can then work easily and efficiently on the underlying sorted items. One such example of such an algorithm is searching for a specific key in a sequence of elements. When the elements are sorted searching can be done more efficiently.

Selecting the optimal order to multiply matrices can occur in programs/algorithms that update their "state" through linear transformations. When this is the case, and the transformations can be cashed, in other words they don't need to be performed immediately, then computing an optimal ordering in which to calculate the individual matrix products could radically reduce the total computational time.

Finding the convex hull occurs in many graphics programs where the convex hull finding algorithm needs to determine the largest "box" required to contain all the given data points.

Exercise 1.1-2 (measures of efficiency)

Other common measures of efficiency used to compare algorithms could be anything that might be constrained in a real world setting. Examples of this are memory constraints both disk and random access memory (RAM), the number of memory accesses, determinism as opposed to a randomize algorithm, number of files created, number of sockets opened, number of Internet connections established etc.

Exercise 1.1-3 (an example data structure)

A common data structure often used is a linked list. Such a data structure can easily insert items into any location within the data structure once the desire insertion point is known. A linked list structure cannot locate new elements or locations quickly since it must effectively look at each element one at a time until the desired one is found.

Exercise 1.1-4 (shortest-path v.s. traveling-salesman problems)

In the shortest-path problem the path through the network is often only one way. It is not required to end up at the starting location, but just to end at the last destination desired. In the traveling-salesman problem the algorithm must start and *end* at the *same* location while visiting all other destinations en-route. This requirement that we start and end at the same location while visiting all intermediate locations makes the problem more difficult.

Exercise 1.1-5 (when only the optimal will do)

There are relatively few situations in real life where *only* the optimal will do. This is because often in formulating a physical problem into a framework for an algorithm to solve will involve approximations and simplifications in itself. Using an approximate algorithm (assuming that it is not too far from optimal) does not introduce errors greater than what has already been introduced in the approximations done earlier.

There are of course cases where we want no errors in the algorithms that we use, for example in any algorithm that involves monetary calculations.

1.2 (Algorithms as a technology)

Exercise 1.2-1

Modern day global positioning devices (GPS) that provide instructions on how to get from place to place using road networks are a application that uses algorithms like discussed in this book very heavily.

Exercise 1.2-2

For this exercise we want to determine the smallest value of n such that

$$T_{\text{merge sort}}(n) = 65n\lg(n) < 8n^2 = T_{\text{insertion sort}}(n)\,.$$

We can find such a value of n by plotting both sides of this inequality and observing where they two curves cross. We do that in the following `R` code

```
ns = 5:80
t_merge = 65 * ns * log( ns, base=2 )
t_insert = 8 * ns^2
plot( ns, t_merge, type='l', col='green', xlab='n', ylab='time',
      ylim=c(min(c(t_merge,t_insert)), max(c(t_merge,t_insert))) )
```

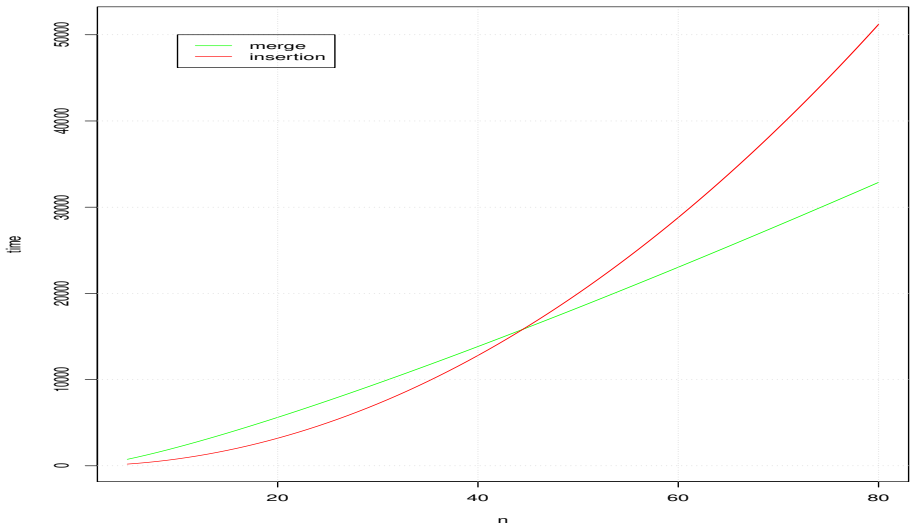

Figure 1: A comparison between the time to run "merge sort" and "insertion sort" on an input of size n.

```
lines( ns, t_insert, col='red' )
grid()
legend( 10, 50000, c("merge", "insertion"), col=c("green","red"), lty=c(1,1) )
```

Running the above code we get Figure 1. From that figure it looks like the two curves cross when $n \approx 45$. Thus insertion sort is actually *faster* than merge sort when n is smaller than this value. For values of n large we see that merge sort runs in less time than insertion sort.

Exercise 1.2-3

For this we want to determine the smallest value of n such that

$$100n^2 < 2^n \,.$$

We can do a plot like in the previous problem to determine this in the following R code

```
ns = 1:17
t_left = 100 * ns^2
t_right = 2^ns
plot( ns, t_left, type='l', col='green', xlab='n', ylab='time',
      ylim=c(min(c(t_left,t_right)), max(c(t_left,t_right))) )
lines( ns, t_right, col='red' )
grid()
legend( 1, 120000, c("quadratic growth", "exponential growth"),
        col=c("green","red"), lty=c(1,1) )
```

When we run the above code we get Figure 2. In that plot we see that after an initial section (where n is small) the value of 2^n becomes *much* larger than the quadratic competitor.

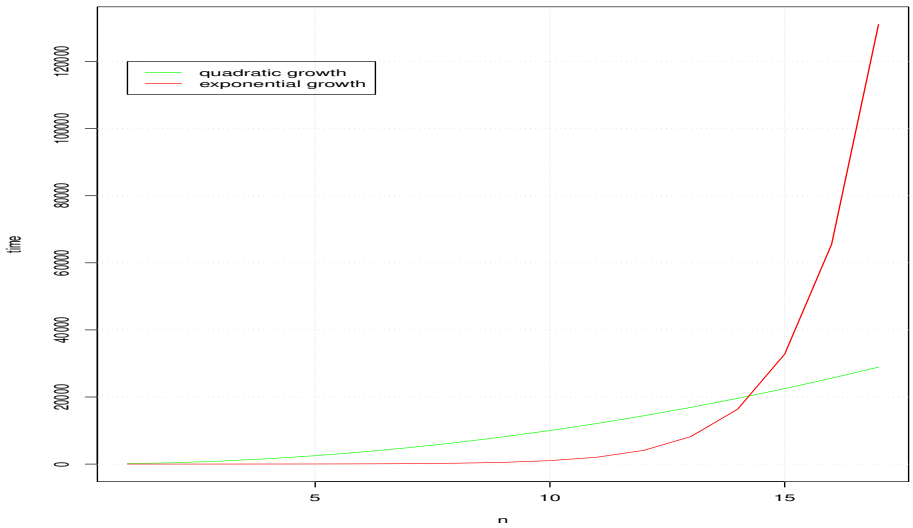

Figure 2: A comparison of the growth functions $100n^2$ and 2^n.

Problem 1.1

For each of the given functions in the first column of the given table we need to invert the equation

$$10^6 f(n) = \text{solve time limit in seconds}\,,$$

to determine n. For many of the given functions, $f(n)$, we can invert easily. Two that we cannot are the functions $n\lg(n)$ and $n!$. For these two the following `R` code provides "inverses" for them

```
invert_nlogn = function(y){
  # "solve" for an integer x such that: x log(x) = y
  #
  res = uniroot( function(x){ x*log(x) - y }, c(1,y) )
  return(res$root)
}

invert_nfactorial = function(y){
  # "solve" for an integer x such that: x! is close to y
  # i.e. x factorial "equals" y
  #
  x = 1
  while( factorial(x) < y ){
    x = x + 1
  }
  if( abs(factorial(x-1)-y) <= abs(factorial(x)-y) ){
    return(x-1)
  }else{
    return(x)
  }
}
```

Function	1 second	1 minute	1 hour	1 day	1 month	1 year	1 century
$\lg(n)$	∞	∞	∞	∞	∞	∞	∞
$\sqrt{n}$	$O(10^{12})$	$O(10^{15})$	$O(10^{19})$	$O(10^{21})$	$O(10^{24})$	$O(10^{29})$	$O(10^{33})$
n	$O(10^{6})$	$O(10^{7})$	$O(10^{9})$	$O(10^{10})$	$O(10^{12})$	$O(10^{14})$	$O(10^{16})$
$n\lg(n)$	$O(10^{4})$	$O(10^{6})$	$O(10^{8})$	$O(10^{9})$	$O(10^{11})$	$O(10^{13})$	$O(10^{15})$
n^2	$O(10^{3})$	$O(10^{3})$	$O(10'4)$	$O(10^{5})$	$O(10^{6})$	$O(10^{7})$	$O(10^{8})$
n^3	100	400	1500	4000	10000	100000	500000
2^n	20	26	32	36	41	49	56
$n!$	9	11	13	14	15	17	19

Table 1: The largest problem size n (approximate) that is solvable by the time in the column heading when the time to solve a problem of size n takes $f(n)$ microseconds. Here the value of ∞ represents numerical overflow when trying to invert the given function.

Using these functions we can compute the largest value of n that a problem can be solved in the given time using the following `R` code

```
# the time we can spend solving (in seconds)
solve_time = c( 1, 60, 60*60, 24*3600, 30*24*3600, 365*30*24*3600, 100*365*30*24*3600 )
solve_time_ms = solve_time / 10^(-6)

# Here we invert each function:
data = c( exp( solve_time_ms ),
          solve_time_ms^2,
          solve_time_ms,
          sapply( solve_time_ms, invert_nlogn ),
          sqrt( solve_time_ms ),
          solve_time_ms^(1/3),
          log( solve_time_ms, base=2 ),
          sapply( solve_time_ms, invert_nfactorial ) )
```

If we put the numbers we get from the above `R` code into a table we get Table 1. Notice that if we have a fixed time budget (i.e. we are looking at a particular column) then as we move downwards though the rows, the size of the problem we can solve in that time drops drastically.

Chapter 2 (Getting Started)

2.1 (Insertion sort)

Exercise 2.1-1

Given the array

$$31\,,41\,,59\,,26\,,41\,,58\,,$$

the sequence of steps followed by insertion sort would be as follows. We first note that the first three elements are in sorted order and that the fourth element is not. Thus our sorted list of cards starts with the first three and we want to insert the fourth element. Note that the fourth element goes at the first of the list thus we shift all of the first three elements to the right by one to make space for the 26. This gives

$$26\,,31\,,41\,,59\,,41\,,58\,.$$

Next we see that the fifth element (here a 41) needs to be at the third or fourth location so we shift the 59 one to the right to get

$$26\,,31\,,41\,,41\,,59\,,58\,.$$

Finally inserting the 58 into its correct position in the array gives

$$26\,,31\,,41\,,41\,,58\,,59\,.$$

Exercise 2.1-2

To change the INSERTION-SORT routine to sort the numbers in decreasing order we change the line that says

```
while i > 0 and A[i] > key
```

to

```
while i > 0 and A[i] < key
```

then we will be searching linearly though the sorted list to find the position where `A[i] >= key` and that inequality signals that the value of `key` should go at position `A[i+1]`.

A	B	carry	binary sum	least significant digit in sum	new carry value
0	0	0	0	0	0
0	1	0	1	1	0
1	0	0	1	1	0
1	1	0	10	0	1
0	0	1	1	1	0
1	0	1	10	0	1
0	1	1	10	0	1
1	1	1	11	1	1

Table 2: The sum of the binary digits A and B (with a possible carry bit)

Exercise 2.1-3 (linear search)

Pseudocode for this procedure is given as follows.

```
LINEAR-SEARCH(A, ν)
1  for i ← 1 to length[A]
2  if A[i] == ν
3      then return i
4  return NIL
```

Exercise 2.1-4 (adding binary arrays)

For this problem we have two binary arrays A and B each with n bits. We want to write pseudocode for the vector C such that $C = A + B$ bitwise. In doing this we will let the variable *carry* hold a carry bit if needed (the value of *carry* will be 0 or 1). We can consider all possible binary values for A, B, *carry* and their sum in Table 2. Using this mapping, the pseudocode for this procedure follows

```
BINARY-ADD(A, B, C)
 1  carry ← 0
 2  for i ← 1 to length[A]
 3        do
 4           ▷ perform the binary addition A[i] + B[i] + carry
 5           if carry == 0
 6              then if A[i] == 1 and B[i] == 1
 7                        then carry ← 1
 8              else  if A[i] == 0 and B[i] == 0
 9                        then carry ← 0
10           ▷ C[i] should get the least significant bit of the given binary addition
11           C[i] ← A[i] + B[i] + carry
12  ▷ update the last element of C
13  if carry == 1
14     then C[length[A] + 1] ← carry
```

Exercise 2.2-1

The given function is $\theta(n^3)$.

Exercise 2.2-2 (selection sort)

We can better understand this algorithm is we consider moving the elements from A "to the right" one at at time. In reality, we move each element into the left most locations of the array so that the "left side" of the array is sorted and the "right side" of the array has the elements that have yet to be sorted in them. When the algorithm is finished the original array A will contain a fully sorted list. What we consider here is for pedagogical purposes. Consider the same numerical elements found on page 10. To start we move/copy over the smallest element from A (which is 26) as

$$31, 41, 59, 26, 41, 58 \rightarrow 26$$

Here the left set of numbers represent the array A "before" and the right set of numbers represent the array of sorted numbers we are building up. We can then imagine replacing the 26 in A with some value that we would ignore in further passes of the algorithm (like infinity) here we denote that element as x. Looking for the next smallest number in the elements of A that remain we find it to be 31 which gives the algorithmic step

$$31, 41, 59, x, 41, 58 \rightarrow 26, 31$$

Replacing 31 with x and finding the next smallest number to be one of the 41's which gives the algorithmic step

$$x, 41, 59, x, 41, 58 \rightarrow 26, 31, 41$$

Replacing that 41 with x and finding the next smallest number to be the other 41 we get the algorithmic step

$$x, x, 59, x, 41, 58 \rightarrow 26, 31, 41, 41$$

Next we find 58 to have

$$x, x, 59, x, x, 58 \rightarrow 26, 31, 41, 41, 58$$

Finally we find 59 to get

$$x, x, 59, x, x, x \rightarrow 26, 31, 41, 41, 58, 59$$

On the right-hand-side we now have a sorted list obtained from the original list A.

Having done an example of how this algorithm works we now present pseudocode for its implementation

```
SELECTION-SORT(A)
 1  for i ← 1 to length[A] − 1
 2        do
 3           ▷ Find the smallest element in the array from A[i + 1] to A[length[A]]
 4           sindex ← i
 5           for j ← i + 1 to length[A]
 6                 do
 7                    if A[j] < A[sindex]
 8                       then sindex ← j
 9           ▷ Swap elements A[i] and A[sindex]
10           tmp ← A[i]
11           A[i] ← A[sindex]
12           A[sindex] ← tmp
```

The invariant of this code is that at each iteration of the outer loop all elements in the "left" half of the array $A[1 \ldots i-1]$ will be the sorted $i-1$ smallest elements from A. We only need to loop over the first $n-1$ elements since if we sort this many elements the last (and largest element) must be found at the last location of the array $A[n]$ and the full array A is sorted.

The best-case and worst-case performance for this algorithm are the same since the above loops are performed regardless of what elements are needed to be exchanged. Another way to say this is to note that we do not know a priori that one does not need to search all elements of the subarray $A[i+1 \ldots n]$ and thus have to do the work regardless of the input array.

Note that in the inner minimization loop we must search over a list of length $n-i$ and we do this for i starting from 1 to $n-1$. Thus the time complexity of this procedure is dominated

by this inner and outer loop. Thus we have

$$\begin{aligned} T(n) &= C\sum_{i=1}^{n-1}(n-i) + \text{lower order terms} \\ &= C\sum_{i=1}^{n-1} i + \text{lower order terms} \\ &= C\left(\frac{n(n-1)}{2}\right) + \text{lower order terms} = \theta(n^2)\,. \end{aligned}$$

We will see better sorting routines in later parts of this book.

Exercise 2.2-3

Without any special assumptions on the input array A I would say that on average $n/2$ elements would need to be searched to find a given element. This can be argued by assuming that on average $1/2$ of the elements in A will be larger than v and $1/2$ of the elements in A will be smaller than v. Thus on average $n/2$ elements would have to be searched. The best case for this algorithm is when the search key v is the first element of the array. The worst case for this algorithm is when v is the last element of the array. In all cases we have $T(n) = \theta(n)$.

Exercise 2.2-4

We could modify our algorithm to check if the input is equal to a specially chosen input and then have the algorithm return a precomputed result. This would really require no algorithmic work and thus the best case performance of this algorithm (on these special cases) is extremely fast.

Exercise 2.3-1

The given array starts as

$$< 3\,, 41\,, 52\,, 26\,, 38\,, 57\,, 9\,, 49 >$$

We first break this array into two parts

$$< 3\,, 41\,, 52\,, 26 > \text{ and } < 38\,, 57\,, 9\,, 49 >$$

Each of these arrays is then further broken up into two parts (each with only two elements)

$$< 3\,, 41 > \text{ and } < 52\,, 26 > \text{ and } < 38\,, 57 > \text{ and } < 9\,, 49 >$$

We would next break each pair of element into eight single element lists which we would then have to combine with the MERGE command. This results in each two element list being sorted and we get

$$< 3 , 41 > \text{ and } < 26 , 52 > \text{ and } < 38 , 57 > \text{ and } < 9 , 49 >$$

Next we merge these groups of two elements into two groups each with four elements to get

$$< 3 , 26 , 41 , 52 > \text{ and } < 9 , 38 , 49 , 57 >$$

Finally we merge these two sorted groups of four elements into on fully sorted array to get

$$< 3 , 9 , 26 , 38 , 41 , 49 , 52 , 57 >$$

Exercise 2.3-2

Recall that the procedure MERGE(A, p, q, r) merges two sorted subarrays of A namely $A(p, q)$ and $A(q + 1, r)$ into a single sorted list. To start the elements from $A(p, q)$ and $A(q + 1, r)$ are copied into two working arrays L (for left) and R (for right). This part of the algorithm is the same as that given in the text. Once we have our data in these two working arrays we can proceed to copy it without using sentinels. The pseudocode for this entire procedure is then given by

```
MERGE(A, p, q, r)
 1  ▷ the number of elements in the left array
 2  n_1 ← q − p + 1
 3  ▷ the number of elements in the right array
 4  n_2 ← r − q
 5  create arrays L[1 . . . n_1] and R[1 · · · n_2]
 6  for i ← 1 to n_1
 7        do L[i] ← A[p + i − 1]
 8  for j ← 1 to n_2
 9        do R[j] ← A[q + j]
10  i ← 1 ▷ holds the index into L
11  j ← 1 ▷ holds the index into R
12  k ← 1 ▷ holds the index into A
13  while i ≤ n_1 and j ≤ n_2
14        do if L[i] ≤ R[j]
15              then A[k] ← L[i]
16                   i ← i + 1
17              else  A[k] ← R[j]
18                   j ← j + 1
19           k ← k + 1
20  ▷ at this point one of L or R is now empty
21  while i ≤ n_1 ▷ never executed if the left array "runs out first"
22        do
23           A[k] ← L[i]
24           i ← i + 1
25           k ← k + 1
26  while j ≤ n_2 ▷ never executed if the right array "runs out first"
27        do
28           A[k] ← R[j]
29           j ← j + 1
30           k ← k + 1
```

Exercise 2.3-3

We want to show that for the given recurrence relationship that the solution is

$$T(n) = n \lg(n) \,,$$

when n is a power of two. Lets check this for the first power of two i.e. when $n = 2$ that the proposed solution is true. In this case we have $T(2) = 2\lg(2) = 2$ which is correct.

Next, as needed for mathematical induction assume that $T(n) = n\lg(n)$ for n a power of two up to some point. That is for $n \in \{2^1, 2^2, 2^3, \ldots, 2^{k-1}, 2^k\}$. Then for the next power of

two using the recurrence relationship for T we have

$$\begin{aligned} T(2^{k+1}) &= 2T\left(\frac{2^{k+1}}{2}\right) + 2^{k+1} = 2T(2^k) + 2^{k+1} \\ &= 2(2^k \lg(2^k)) + 2^{k+1} \quad \text{using the induction hypothesis} \\ &= k2^{k+1} + 2^{k+1} = 2^{k+1}(k+1) = 2^{k+1}\lg(2^{k+1})\,. \end{aligned}$$

As this last expression is $T(2^{k+1})$ we are done.

Exercise 2.3-4

Some very simple pseudocode of the suggested procedure it might be

RECURSIVE-INSERTION-SORT(A, n)
1 RECURSIVE-INSERTION-SORT$(A, n-1)$
2 $\triangleright$ Insert $A(n)$ into the sorted list $A[1 \ldots n-1]$

This is enough to see that if $T(n)$ represents the amount of time it takes to run this code on an array of length n we have that

$$T(n) = T(n-1) + \theta(n-1)\,,$$

since to do the insertion of $A(n)$ into $A[1 \ldots n-1]$ we must process at most n elements. Note we could use a bisection based search method to find the location where $A(n)$ should be placed in the array $A[1 \ldots n-1]$ which would require $\lg(n)$ work. Once this location is found we still need to move the elements of A "to the right" to insert it. This insertion process will require $\theta(n)$ work.

The above difference equation is one that we can solve using methods presented later in the text. However, since this equation is so simple there are other methods one can use to solve for $T(n)$. Using methods from [1] we can write it as

$$\Delta T(n) = \theta(n)\,,$$

so that "integrating" (reverse differencing in this case) both sides we get $T(n) = \theta(n^2)$. This book will present a general method for solving recurrences of this form in later chapters.

As an aside, pseudocode of the procedure RECURSIVE-INSERTION-SORT might be

```
RECURSIVE-INSERTION-SORT(A, n)
1  RECURSIVE-INSERTION-SORT(A, n − 1)
2  key ← A[n]
3  i ← n − 1
4  ▷ shift elements of A "to the right" to make space for A[n]
5  while i > 0 and A[i] > key
6        do
7           A[i + 1] ← A[i]
8           i ← i − 1
9  A[i + 1] ← key
```

Exercise 2.3-5

To start this problem we will write pseudocode for iterative binary search. In that procedure we assume that our input array A is sorted and we are given an element v to search for. In the pseudocode below, we will take p and q to be indexes into A such that we want to search for v from all elements between $A[p]$ and $A[q]$ inclusive. Our pseudocode for iterative binary search is then given by

```
ITERATIVE-BINARY-SEARCH(A, p, q, v)
 1  loc ← NIL
 2  while q − p ≥ 0
 3        do
 4           m ← floor((p + q)/2)
 5           if A[m] = v
 6              then return m
 7           if A[m] > v
 8              then q ← m − 1
 9              else  p ← m + 1
10  return NIL
```

Note that the worst case running time of this algorithm would happen when we have to divide the array in two the most number of times possible that is if we had to search $\lg(n)$ times, thus we would have $T(n) = \theta(\lg(n))$. We now present pseudocode for *recursive* binary search

```
RECURSIVE-BINARY-SEARCH(A, p, q, v)
1  if p − q < 0
2      then return NIL
3  m ← floor((p + q)/2)
4  if A[m] = v
5      then return m
6  if A[m] > v
7      then return RECURSIVE-BINARY-SEARCH(A, p, m − 1, v)
8      else  return RECURSIVE-BINARY-SEARCH(A, m + 1, q, v)
```

From the recursive search pseudocode we have that

$$T(n) = T\left(\frac{n}{2}\right) + \theta(1)\,.$$

The solution to this recurrence relationship (to be demonstrated later) is $T(n) = \theta(\lg(n))$ the same as we got in the earlier argument.

Note that we would initially call each procedure with the arguments $(A, 1, n, v)$.

Exercise 2.3-6

The answer to this question is no. Even if we could obtain the location of the index i in the array A where we would need to place the element $A[j]$ at *no* computational cost we still must move all the elements of A to insert $A[j]$ into the array $A[1 \ldots j]$. This shuffling of the element requires at most $\theta(j)$ work. Thus the worst case complexity of insertion sort is not changed from $\theta(n^2)$ by using binary search. While the above is a worst case complexity argument note that in practice this change would improve the performance of insertion sort. As there are better algorithms for sorting than insertion sort this observation is not that useful however.

Exercise 2.3-7

To start this problem, note that exhaustive search (over all pairs of elements of S) would require

$$\binom{n}{2} = \frac{n!}{(n-2)!2!} = \frac{n(n-1)}{2} = \theta(n^2)\,,$$

amount of work and thus is much more costly than the procedure we describe next.

To start, we will sort the set S dropping any duplicate elements. Using merge sort the complexity of this action is $\theta(n \lg(n))$. To derive the remainder of our algorithm we will take note of the following two facts. If we have two numbers a and b such that $a < \frac{x}{2}$ and $b < \frac{x}{2}$. Then we will have that

$$a + b < x\,.$$

In the same way if we have two numbers such that $a > \frac{x}{2}$ and $b > \frac{x}{2}$. Then we will have that

$$a + b > x\,.$$

Thus the point $\frac{x}{2}$ determine the location in our sorted list such that if there *are* two numbers that sum to x then one of them must be smaller than $\frac{x}{2}$ and the other one must be larger than $\frac{x}{2}$. The argument above shows that if that was not true then the two numbers would sum to a value that was less than x or to a value that was greater than x. Find the location where $\frac{x}{2}$ would be placed in the sorted list of original numbers. This location splits the sorted list into two smaller sorted lists $\mathcal{A}$ and $\mathcal{B}$ where all elements in $\mathcal{A}$ are less than $\frac{x}{2}$ and all elements of $\mathcal{B}$ are greater than $\frac{x}{2}$. If two elements sum to x one must come from $\mathcal{A}$ and another from $\mathcal{B}$. In fact if we take each value of $a \in \mathcal{A}$ and compute a "target" of $\tilde{b} = x - a$ then if we find that $\tilde{b} \in \mathcal{B}$ we are done. In the same way if we take each value of $b \in \mathcal{B}$ and compute $\tilde{a} = x - b$ then if $\tilde{a} \in \mathcal{A}$ we are done.

To turn these observations into an algorithm we do the following. For each value of $a \in \mathcal{A}$ we compute the value $x - a$ (there are $|\mathcal{A}|$ of these numbers) and for each value of $b \in \mathcal{B}$ we compute the value $x - b$ (there are $|\mathcal{B}|$ of these numbers). All of this work then costs $|\mathcal{A}| + |\mathcal{B}| = \theta(n)$.

As argued above, if we have two values a and b that sum to x then one of these numbers (formed from $x - a$ or $x - b$) that we just computed must *also* be in the original set S. If see if this is true we form a new list of length $2n$ consisting of the original set S and all of the numbers we just computed. We then sort this list which costs

$$\theta(2n \lg(2n)) = \theta(2n(\lg(n) + \lg(2)) = \theta(n \lg(n))\,,$$

work. We can then walk this sorted list looking for any adjacent *duplicate* elements. If we find any, we have that there must be two elements that sum to x. The total complexity of this procedure is $\theta(n \lg(n) + n) = \theta(n \lg(n))$.

Problem 2-1: (Insertion sort on small arrays in merge sort)

In this problem we break our initial list of n elements into $\frac{n}{k}$ smaller lists each of length k.

Part (a): We start by sorting each of the $\frac{n}{k}$ lists using insertion sort. As insertion sort requires $\theta(k^2)$ work and we have $\frac{n}{k}$ lists to sort the total amount of work required is

$$\left(\frac{n}{k}\right)\theta(k^2) = \theta(nk)\,.$$

Part (b): One way (not optimal) to merge these $\frac{n}{k}$ smaller lists into one complete list would be the following. As each $\frac{n}{k}$ lists are sorted (say in ascending order) we first select the single smallest value from the set of the first $\frac{n}{k}$ values in each smaller list. This can be done in $\frac{n}{k}$ work (as we scan a single value in each of the smaller arrays) and then copy the smallest of these to our output array. We now compare another $\frac{n}{k}$ values, find the smallest, and copy

its value to our output array. This needs to be done $\theta(n)$ times (once for each element in the original array) and each minimization requires $\theta\left(\frac{n}{k}\right)$ work giving

$$\theta\left(\frac{n^2}{k}\right),$$

work under this algorithm.

A better algorithm would be to perform merges of adjacent pairs of smaller sublists. The first step of this algorithm will merge pairs of lists (each of size k) and produce sorted lists of size $2k$. The merge requires $\theta(2k)$ work and there are $\frac{n}{2k}$ pairs to merge giving a total amount of work in the first step of

$$\theta\left(2k\left(\frac{n}{2k}\right)\right) = \theta(n).$$

Merging these smaller lists of size $2k$ with $\frac{n}{4k}$ pairs gives a total amount of work of

$$\theta\left(2(2k)\left(\frac{n}{4k}\right)\right) = \theta(n),$$

again. Continuing this argument a third time we again get $\theta(n)$ amount of work. Thus at each step we need to perform $\theta(n)$ amount of work. We have to perform $\lg\left(\frac{n}{k}\right)$ of these steps to get the final sorted list of length n. Thus the total work under this algorithm is

$$\theta\left(n\lg\left(\frac{n}{k}\right)\right).$$

Part (c): The combined algorithm then must do an amount of work given by

$$\theta\left(nk + n\lg\left(\frac{n}{k}\right)\right).$$

Recall that directly applying merge sort on a list of length n takes $\theta(n\lg(n))$. Thus we want to pick k such that

$$\theta\left(nk + n\lg\left(\frac{n}{k}\right)\right) \sim \theta(n\lg(n)).$$

To get a understanding of what k might need to be we can try to "divide" by n on both sides to get

$$\theta\left(k + \lg\left(\frac{n}{k}\right)\right) \sim \theta(\lg n)).$$

In the above expression to leading order we must take

$$k = \theta(\lg(n)).$$

Part (d): In practice we could run timing tests to see how large k could be to have insertion sort be faster than merge sort.

Problem 2-2: (Correctness of bubblesort)

Part (a): We also need to prove that the elements of A' are the same as the elements of A i.e. that the elements of A' are a permutation of the elements of A.

Part (b): Let the loop invariant for the inner loop be the following. At the beginning and end of the inner loop we will have $A[j]$ to be the smallest element from all those that come after it. That is $A[j] = \min\{A[i] : j \leq i \leq n\}$. We need to show that this loop invariant is satisfied at the start of the loop and is true at the end of the loop.

At the start of the loop $j = n = \text{length}[A]$ and so $A[j]$ is trivially the minimum of the set with only one element $\{A[n]\}$.

For a general index value j, if there was an element of $A[i]$ smaller than $A[j]$ for $j < i \leq n$ then it would have been exchanged at some point in the loop and would have to come before $A[j]$. At the completion of this loop we will have the smallest value at location j and the loop invariant holds.

Part (c): Let the loop invariant for the outer loop be the following. At index i the elements in $A[1 \ldots i-1]$ are the smallest elements from A and are in sorted order.

Before the first iteration we have $i = 1$ and the loop invariant trivially holds.

Consider iteration i of the outer loop. Then from the loop invariant on the previous outer loop iteration we have that $A[1 \ldots i-1]$ holds the sorted smallest $i-1$ elements from A. From the loop invariant for the inner loop when it ends we have that $A[i]$ is the smallest value from the remaining elements of A. It stands to reason that $A[1 \ldots i]$ is the sorted smallest i elements from A and the loop invariant is true at the end of the outer loop.

Part (d): To determine the worst-case running time for bubblesort note that the code that is executed inside the two nested loops takes $\theta(1)$ time. The innermost loop has this code executed $n - (i+1) + 1 = n - i$ times for each value of i in the outer most loop. The outermost loop runs for $i = 1$ to $i = n$. Thus the total work done is

$$\sum_{i=1}^{n} \theta(n-i) = \sum_{i=1}^{n} \theta(i) = \theta(n^2)\,.$$

This is the same worst-case running time for insertion sort.

Problem 2-3: (Correctness of Horner's rule)

Part (a): The while loop in the given pseudocode will execute $\theta(n)$ times and thus this algorithm has $\theta(n)$ running time.

Part (b): To evaluate a polynomial at x we will compute

$$P(x) = \sum_{k=0}^{n} a_k x^k\,,$$

directly. This means that as a "subroutine" we need to compute the terms $a_k x^k$. We do this in the following pseudocode

```
NAIVE-POLYNOMIAL-EVALUATION(a, x)
 1  ▷ the polynomial coefficients are held in the array a such that a_0 = a[1], a_1 = a[2] etc.
 2  y ← 0
 3  k ← 0
 4  while k ≤ n
 5        do
 6           ▷ Compute the term a_k x^k
 7           i ← 1
 8           term ← a[k]
 9           while i ≤ k
10                 do
11                    ▷ multiply by x
12                    term ← term ×x
13                    i ← i + 1
14           y = y + term
15           k ← k + 1
16  return y
```

The innermost while loop takes $\theta(1)$ work and is executed $\theta(k)$ times. The outermost loop then takes

$$T(n) = \theta\left(\sum_{k=0}^{n} k\right) = \theta\left(\frac{n(n+1)}{2}\right) = \theta(n^2)\,.$$

This is "worse" than the procedure for Horner's rule.

Part (c-d): To show that the given expression for y is a loop invariant we must show that it is true before the loop starts and after the loop ends. At the start of the loop $i = n$ and the value of the proposed loop invariant y is zero by the convention of a summation with no terms (since the upper limit of the sum is -1). Assuming the loop invariant is true at the top of the loop we can write it as

$$y = \sum_{k=0}^{n-(i+1)} a_{k+i+1}x^k = a_{i+1} + a_{i+2}x + a_{i+3}x^2 + \cdots + a_{n-1}x^{n-i-2} + a_n x^{n-i-1}\,.$$

Using this we see that the value of y at the end of the loop is given by

$$\begin{aligned} y &\leftarrow a_i + xy \\ &= a_i + a_{i+1}x + a_{i+2}x^2 + a_{i+3}x^3 + \cdots + a_{n-1}x^{n-i-1} + a_n x^{n-i} \\ &= \sum_{k=0}^{n-i} a_{k+i+1}x^k\,, \end{aligned}$$

which shows that the loop invariant holds at the end of the loop. Thus at the top of the last iteration of the loop we have $i = 0$ where y is given by

$$y = \sum_{k=0}^{n-1} a_{k+1}x^k\,,$$

so that at the end of the while loop we have y given by

$$y = a_0 + \sum_{k=0}^{n-1} a_{k+1} x^{k+1} = \sum_{k=0}^{n} a_k x^k \,,$$

the value of the polynomial at x.

Problem 2-4: (Inversions)

Part (a): Given the list $< 2, 3, 8, 6, 1 >$ we find that the inversions in it are

$$(1,5)\,,(2,5)\,,(3,4)\,,(3,5)\,,(4,5)\,.$$

Recall that inversions are specified by the *indices* rather than the array values.

Part (b): The list with the most inversions is $< n, n-1, n-2, \cdots, 3, 2, 1 >$. This list has

$$\begin{aligned}(n-1)+(n-2)+\cdots+2+1 &= \sum_{k=1}^{n-1} k = \sum_{k=1}^{n} k - n \\ &= \frac{n(n+1)}{2} - n = \frac{n}{2}(n+1-2) \\ &= \frac{n(n-1)}{2} = \theta(n^2)\,.\end{aligned}$$

Part (c): Every inversion must be "undone" by a step in the insertion sort inner while loop and thus we expect that the running time of insertion sort to be proportional to the number of inversions.

Part (d): Following the hint given in the book we will attempt to implement a divide-and-conquer merge sort type algorithm to count the number of inversion in an input array. If we take the merge sort algorithm as a starting point then when after splitting the array A into two parts we see that the number of inversion in the original array A will be the number of inversions found in the left-half plus the number of inversions found in the right-half plus the number of inversions involving elements in the left and right halves.

In each of the recursive calls lets assume that our algorithm returns the number of inversions in the left and right subarrays. We then have to implement a merge like procedure that counts the number of inversions between the two left and right arrays. The big leap to understanding this problem is to notice that if the left and right subarrays are returned *sorted* (along with the number of inversions each subarray contained) we can easily count the number of "cross inversions" when we merge the two arrays.

Assume that we have two sorted arrays L (for "left") and R (for "right") and we will merge these two into an output sorted array as in merge sort. In that procedure if we are placing an element of L into the output array then this element is smaller than all other elements in L and is also smaller than all other elements in R. Thus this operation does not reveal

an inversion between elements in L and R. On the other hand, if we place an element from R into the output array then this element must be smaller than all the remaining unplaced elements in the L array.

If there are m remaining elements in L we have "found" m cross inversions. As we continually place elements from L and R onto the output array each time we place one from R we add m to the number of cross inversions found. At this point the pseudocode for this procedure can be written as

```
COUNT-INVERSIONS-N-SORT(A, p, r)
1  ▷ Returns the number of inversions and A sorted between the two indices p and r
2  if p ≥ r
3      then return 0
4  q ← floor((p + r)/2)
5  inversions ← COUNT-INVERSIONS-N-SORT(A, p, q)
6  inversions ← inversions + COUNT-INVERSIONS-N-SORT(A, q + 1, r)
7  inversions ← inversions + COUNT-INVERSIONS-N-MERGE(A, p, q, r)
8  return inversions
```

Next we need the following pseudocode (borrowing form the procedure MERGE above).

```
COUNT-INVERSIONS-N-MERGE(A, p, q, r)
 1  ▷ the number of elements in the left array
 2  n_1 ← q − p + 1
 3  ▷ the number of elements in the right array
 4  n_2 ← r − q
 5  create arrays L[1 . . . n_1] and R[1 · · · n_2]
 6  for i ← 1 to n_1
 7        do L[i] ← A[p + i − 1]
 8  for j ← 1 to n_2
 9        do R[j] ← A[q + j]
10  i ← 1 ▷ holds the index into L
11  j ← 1 ▷ holds the index into R
12  k ← 1 ▷ holds the index into A
13  inversions ← 0
14  while i ≤ n_1 and j ≤ n_2
15        do if L[i] ≤ R[j]
16              then A[k] ← L[i]
17                   i ← i + 1
18              else A[k] ← R[j]
19                   j ← j + 1
20                   ▷ Add in the inversion count
21                   inversions ← inversions +(n_1 − i + 1)
22           k ← k + 1
23  ▷ at this point one of L or R is now empty
24  while i ≤ n_1 ▷ never executed if the left array "runs out first"
25        do
26           A[k] ← L[i]
27           i ← i + 1
28           k ← k + 1
29  while j ≤ n_2 ▷ never executed if the right array "runs out first"
30        do
31           A[k] ← R[j]
32           j ← j + 1
33           k ← k + 1
34  return inversions
```

From the above code we see that the running time of this algorithm on an input of size n can be written as

$$T(n) = 2T\left(\frac{n}{2}\right) + \theta(n)\,.$$

This has the solution $T(n) = n\lg(n)$ as we were to show.

I should also mention that this algorithm, to report the number of inversions in an array with this complexity, is given as an example of using the "divide-and-conquer" algorithm design paradigm, and was presented in two lectures in Tim Roughgarden's Coursera algorithms

class[1]. One of the requirements of that class was to implement this algorithm. You can find source code for a `python` implementation of that algorithm linked to the web site for these notes.

[1] https://www.coursera.org/course/algo

Chapter 28 (Matrix Operations)

28.1 (Properties of Matrices)

Exercise 28.1-1 (elementary properties of transposes)

These two facts are a simple consequences of the definition of a transpose: the (i,j)th element in M^T is the (j,i)th element in M. For $A+B$ we have that the (i,j)th element in $(A+B)^T$ is the (j,i)th element in $A+B$, which is the sum of the (j,i)th elements in A and B individually, which is also the sum of the (i,j)th elements in A^T and B^T individually. So we have that the (i,j)th element in $(A+B)^T$ is the same as the sum of the (i,j)th element from A^T and B^T. Since A and B are symmetric we have that

$$(A+B)^T = A^T + B^T = A + B$$

and the matrix $A+B$ is symmetric. The proof for the difference of A and B is done in the same way.

Exercise 28.1-2 (transpose of a product)

To prove that $(AB)^T = B^T A^T$ we begin by looking at the components of the product AB. For simplicity assume A and B are n by n. Then the (i,j)th entry of AB is given by

$$(AB)_{i,j} = \sum_{k=1}^{n} A_{i,k} B_{k,j} \,.$$

Then the transpose of AB has components given by

$$((AB)^T)_{i,j} = (AB)_{j,i} = \sum_{k=1}^{n} A_{j,k} B_{k,i} \,.$$

Note that the components of A in the above can be expressed in term of A's transpose since $A_{j,k} = (A^T)_{k,j}$. The same can be said for B and when this substitution is done we have

$$((AB)^T)_{i,j} = (AB)_{j,i} = \sum_{k=1}^{n} (A^T)_{j,k} (B^T)_{i,k} = \sum_{k=1}^{n} (B^T)_{i,k} (A^T)_{j,k} \,.$$

Where in the first summation we have replace $A_{j,k}$ with $(A^T)_{k,j}$ (similarly for B) and in the second summation we just placed the term B^T before the term involving A^T. The above can be recognized as the (i,j)th element of the product $B^T A^T$ as expected.

Using the fact that $(AB)^T = B^T A^T$ (proven above), and that $(A^T)^T = A$, for the product $A^T A$ we have that

$$(A^T A)^T = (A)^T (A^T)^T = A^T A \,.$$

thus since $A^T A$ when transposed equals itself we have that it is a symmetric matrix as requested.

Exercise 28.1-3 (uniqueness of a matrix inverse)

If we assume (to reach a contradiction) that *both* B and C are inverses of A then we must have that

$$\begin{aligned} AB &= I \quad \text{and} \quad BA = I \\ AC &= I \quad \text{and} \quad CA = I\,, \end{aligned}$$

Multiplying the equation $AB = I$ on the left by C gives $CAB = C$. Using the fact that $CA = I$, the above simplifies to $B = C$, showing that the inverse of a matrix must be unique.

Exercise 28.1-4 (triangular matrices)

We will prove that the product of two lower triangular matrices is lower triangular by induction. We begin with $n = 2$ for which we have

$$\begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} m_{11} & 0 \\ m_{21} & m_{22} \end{bmatrix} = \begin{bmatrix} l_{11}m_{11} & 0 \\ l_{21}m_{11} + l_{22}m_{21} & l_{22}m_{22} \end{bmatrix}$$

which is lower triangular. Assume the product of two lower triangular matrices of size $\hat{n} \leq n$ is also lower triangular and consider two lower triangular matrices of size $n+1$. Performing a "bordered" block partitioning of the two lower triangular matrices we have that

$$L_{n+1} = \begin{bmatrix} L_n & 0 \\ l^T & l_{n+1,n+1} \end{bmatrix} \quad \text{and} \quad M_{n+1} = \begin{bmatrix} M_n & 0 \\ m^T & m_{n+1,n+1} \end{bmatrix}$$

where the single subscripts denote the order of the matrices. With bordered block decomposition of our two individual matrices we have a product given by

$$L_{n+1}M_{n+1} = \begin{bmatrix} L_nM_n & 0 \\ l^TM_n + l_{n+1,n+1}m^T & l_{n+1,n+1}m_{n_1,n+1} \end{bmatrix}.$$

Since by the induction hypotheses the product L_nM_n is lower triangular we see that our product $L_{n+1}M_{n+1}$ is lower triangular.

The fact that the determinant of a triangular matrix is equal to the product of the diagonal elements, can easily be proved by induction. Lets assume without loss of generality that our system is *lower* triangular (upper triangular systems are transposes of lower triangular systems) and let $n = 1$ then $|G| = g_{11}$ trivially. Now assume that for a triangular system of size $n \times n$ that the determinant is given by the product of its n diagonal elements and consider a matrix $\tilde{G}$ of size $(n+1) \times (n+1)$ partitioned into a leading matrix G_{11} of size $n \times n$.

$$G = \begin{bmatrix} G_{11} & 0 \\ h^T & g_{n+1,n+1} \end{bmatrix}.$$

Now by expanding the determinant of G along its last column we see that

$$|G| = g_{n+1,n+1}|G_{11}| = g_{n+1,n+1}\prod_{i=1}^{n} g_{ii} = \prod_{i=1}^{n+1} g_{ii}\,,$$

proving by induction that the determinant of a triangular matrix is equal to the product of its diagonal elements.

We will prove that the inverse of a lower triangular matrix L (if it exists) is lower triangular by induction. We assume that we are given an L that is non-singular and lower triangular. We want to prove that L^{-1} is lower triangular. We will do this by using induction on n the dimension of L. For $n = 1$ L is a scalar and L^{-1} is also a scalar. Trivially both are lower triangular. Now assume that if L is non-singular and lower triangular of size $n \times n$, then L^{-1} has the same property. Let L be a matrix of size $(n+1) \times (n+1)$ and partition L as follows

$$L = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix}.$$

Where L_{11} and L_{22} are both lower triangular matrices of sizes less than $n \times n$, so that we can apply the induction hypothesis. Let $M = L^{-1}$ and partition M con formally i.e.

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}.$$

We want to show that M_{12} must be zero. Now since $ML = I$ by multiplying the matrices above out we obtain

$$\begin{aligned} LM &= \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \\ &= \begin{bmatrix} L_{11}M_{11} & L_{11}M_{12} \\ L_{21}M_{11} + L_{22}M_{21} & L_{21}M_{12} + L_{22}M_{22} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \end{aligned}$$

Equating block components gives

$$\begin{aligned} L_{11}M_{11} &= I \\ L_{11}M_{12} &= 0 \\ L_{21}M_{11} + L_{22}M_{21} &= 0 \\ L_{21}M_{12} + L_{22}M_{22} &= I\,. \end{aligned}$$

By the induction hypothesis both L_{11} and L_{22} are invertible. Thus the equation $L_{11}M_{11} = I$ gives $M_{11} = L_{11}^{-1}$, and the equation $L_{11}M_{12} = 0$ gives $M_{12} = 0$. With these two conditions the equation $L_{21}M_{12} + L_{22}M_{22} = I$ becomes $L_{22}M_{22} = I$. Since L_{22} is invertible we compute that $M_{22} = L_{22}^{-1}$. As both L_{11} and L_{22} are lower triangular of size less than $n \times n$ by the induction hypothesis their inverse are lower triangular and we see that M itself is then lower triangular since

$$M = \begin{bmatrix} L_{11}^{-1} & 0 \\ M_{21} & L_{22}^{-1} \end{bmatrix}.$$

Thus by the principle of induction we have shown that the inverse of a lower triangular matrix is lower triangular.

Exercise 28.1-5 (permutation matrices)

From the definition of a permutation matrix (a matrix with only a single one in each row/column and all other elements zero) the product PA can be computed by recognizing

that each row of P when multiplied into A will select a single row of A and thus produces a permutation of the rows of A. In the same way the product AP will produce a permutation of the columns of A. If we have two permutation matrices P_1 and P_2, then P_1 acting on P_2 will result in a permutation of the rows of P_2. This result is a further permutation matrix, since it is the combined result of two permutations, that from P_2 and then that from P_1. If P is a permutation, then it represents a permutation of the rows of the identity matrix. The matrix $\hat{P}$ representing the permutation which reorders the rows of P back into the identity matrix would be the inverse of P and from this argument we see that P is invertible and has an inverse that is another permutation matrix. The fact that $P^{-1} = P^T$ can be recognized by considering the product of P with P^T. When row i of P is multiplied by any column of P^T not equal to i the result will be zero since the location of the one in each vector won't agree in the location of their index. However, when row i of P is multiplied by column i the result will be one. Thus we see by looking at the components of PP^T that $PP^T = I$ and P^T is the inverse of P.

Exercise 28.1-6 (Gauss transformations)

Lets assume (without loss of generality) that $j > i$, then we will let M be the elementary matrix (of type 1) that produces A' from A i.e. it adds row i to j and the sum replaces row j. Then since $AB = I$ multiplying this system by M on the left we obtain (since $MA = A'$) that $A'B = M$. From this we see that by multiplying this expression above by M^{-1} on the left we obtain $A'BM^{-1} = I$, showing that the inverse of A' is BM^{-1}. Now the matrix M is like the identity by with an additional one at position (j, i) rather than a zero there. Specifically we have

$$M = \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & \ddots & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & 1 & & & \ddots & \\ & & & & & & 1 \end{bmatrix}.$$

Where the one in the above matrix is at location (j, i). In this case the inverse of M is then easy to compute; it is the identity matrix with a *minus* one at position (j, i), specifically we have

$$M^{-1} = \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & \ddots & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & -1 & & & \ddots & \\ & & & & & & 1 \end{bmatrix}.$$

Now multiplying B by this matrix on the left will operate on the columns of B, thus $B' = BM^{-1}$ is the same matrix as B but with the jth and the negative of the ith column added

together and placed in column j. That is we are subtracting the ith column from the jth column and placing it in column j.

Exercise 28.1-7 (the realness of A^{-1})

Lets begin by assuming that every entry of A is real and then show that every entry of A^{-1} is real. The easiest way to see that all entries of A^{-1} must be real is to recall the adjoint theorem from linear algebra. The adjoint theorem gives the inverse of a matrix in terms of the adjoint of that matrix. Specifically, the adjoint theorem states that

$$A^{-1} = \frac{1}{\det(A)} C^T ,$$

where C is the matrix of *cofactors* of A. This cofactor matrix C is the matrix such that its (i,j) element is given by the cofactor of the element a_{ij} or

$$C_{i,j} = (-1)^{i+j} \det(A_{[ij]}) .$$

Where $A_{[ij]}$ is the ijth minor of the matrix A, i.e. the submatrix that is produced from A by deleting its ith row and its jth column. Because the determinants of the minors of A only involve additions and multiplications, if A has only real elements then all cofactors and determinants of A must be real. By the adjoint theorem above, A^{-1} can have only real elements. The other direction, where we argue that if A^{-1} has only real elements then A must have only real elements can be seen by applying the above arguments to the matrix A^{-1}.

Exercise 28.1-8 (symmetry of the inverse)

Let A be symmetric and invertible. Then by the definition of the inverse, A^{-1} satisfies $AA^{-1} = I$. Now taking the transpose of both sides and remembering that the transpose of a product is the product of the transposes but in the opposite order we have $(A^{-1})^T A^T = I^T$ which simplifies to $(A^{-1})^T A = I$, since both A and I are symmetric. By multiplying both sides on the left by A^{-1} we have that

$$(A^{-1})^T = A^{-1}$$

showing that A^{-1} is symmetric.

That the product BAB^T is symmetric is just an exercise in taking transposes. We have

$$(BAB^T)^T = (B^T)^T A^T B^T = BAB^T ,$$

and this product of matrices is symmetric.

Exercise 28.1-9 (full column rank)

Lets assume that A has full column rank and that $Ax = 0$. Note that $Ax = 0$ is equivalent to the statement that a linear combination of the columns of A sums to zero. Specifically if v_i represents the ith column of A then $Ax = 0$ is equivalent to

$$\sum_{i=1}^{n} x_i v_i = 0 \,.$$

Since A is full column rank its columns are linearly independent (this is the definition of full column rank). Because of this fact, the only way these columns can sum to zero is if $x = 0$ and we have proven one direction. Now lets assume that $Ax = 0$ implies that $x = 0$. This statement is equivalent to the fact that the columns of A are linearly independent which again implies that A is of full column rank.

Exercise 28.1-10 (rank inequalities)

To show this we will first show that

$$\text{rank}(AB) \leq \text{rank}(A) \,.$$

and then use this result to show that

$$\text{rank}(AB) \leq \text{rank}(B) \,.$$

When these two results are combined the rank of the product AB must be *less* than the smaller of the two rank(A) and rank(B) giving the requested inequality of

$$\text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B)) \,.$$

To show that rank$(AB) \leq$ rank(A), we first note that every vector in the column space of AB is in the column space of A. This is true since AB is the action of the matrix A on every column of the matrix B, and the action of a matrix (A) on a vector is a linear superposition of the columns of that matrix (A). Thus the number of linearly independent columns of the *product* matrix AB cannot be greater than that of the number of linear independent columns in the multiplying matrix (A) and we recognize the desired result of

$$\text{rank}(AB) \leq \text{rank}(A) \,.$$

To show the second inequality rank$(AB) \leq$ rank(B), we apply the expression proved above on the matrix $(AB)^T$. Since this matrix equals $B^T A^T$ we have that

$$\text{rank}((AB)^T) = \text{rank}(B^T A^T) \leq \text{rank}(B^T) = \text{rank}(B) \,.$$

But since rank$((AB)^T) =$ rank(AB) replacing the first term in the above gives

$$\text{rank}(AB) \leq \text{rank}(B) \,,$$

showing the second of the desired inequalities. If A or B is invertible, it must be *square* and its rank is equal to its number of columns (equivalently the number of rows). With out loss of generality assume that A is invertible. By the arguments above when we multiply B, the dimension of the column space of AB cannot change from the dimension of the column space of B. Thus we have that

$$\text{rank}(AB) = \text{rank}(B)\,.$$

The same argument applied to $(AB)^T$ (as outlined in the inequality proof above) can be used to show that if B is invertible that

$$\text{rank}(AB) = \text{rank}(A)\,.$$

Exercise 28.1-11 (the determinant of the Vandermonde matrix)

Following the hint given, we will multiply column i by $-x_0$ and add it to column $i+1$ for $i = n-1, n-2, \cdots, 1$. This action will not change the value of the determinant but will make it simpler to evaluate as we will soon see. We begin by first multiplying column $n-1$ by $-x_0$ and adding to column n. We find that

$$\begin{aligned}\det(V(x_0, x_1, x_2, \cdots, x_{n-3}, x_{n-2}, x_{n-1})) &= \begin{vmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-2} & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-2} & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-2} & x_2^{n-1} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-2} & x_{n-1}^{n-1} \end{vmatrix} \\ &= \begin{vmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{n-2} & 0 \\ 1 & x_1 & x_1^2 & \cdots & x_1^{n-2} & (x_1 - x_0)x_1^{n-2} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-2} & (x_2 - x_0)x_2^{n-2} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-2} & (x_{n-1} - x_0)x_{n-1}^{n-2} \end{vmatrix}.\end{aligned}$$

Where we see that this action has introduced a zero in the $(1, n)$ position. Continuing, we now multiply column $n-2$ by $-x_0$ and add it to column $n-1$. We find that the above determinant now becomes

$$\begin{vmatrix} 1 & x_0 & x_0^2 & \cdots & 0 & 0 \\ 1 & x_1 & x_1^2 & \cdots & (x_1 - x_0)x_1^{n-3} & (x_1 - x_0)x_1^{n-2} \\ 1 & x_2 & x_2^2 & \cdots & (x_2 - x_0)x_2^{n-3} & (x_2 - x_0)x_2^{n-2} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & (x_{n-1} - x_0)x_{n-1}^{n-3} & (x_{n-1} - x_0)x_{n-1}^{n-2} \end{vmatrix}.$$

Where we see that this action has introduced another zero this time at the $(1, n-1)$ position. Continuing in this manner we will obtain the following determinant where the first row has

only one non-zero element

$$\begin{vmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & x_1 - x_0 & (x_1 - x_0)x_1 & (x_1 - x_0)x_1^2 & \cdots & (x_1 - x_0)x_1^{n-3} & (x_1 - x_0)x_1^{n-2} \\ 1 & x_2 - x_0 & (x_2 - x_0)x_2 & (x_2 - x_0)x_2^2 & \cdots & (x_2 - x_0)x_2^{n-3} & (x_2 - x_0)x_2^{n-2} \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & x_{n-1} - x_0 & (x_{n-1} - x_0)x_{n-1} & (x_{n-1} - x_0)x_{n-1}^2 & \cdots & (x_{n-1} - x_0)x_{n-1}^{n-3} & (x_{n-1} - x_0)x_{n-1}^{n-2} \end{vmatrix}.$$

We can expand this determinant about the first row easily since there is only a single nonzero element in this row. After this reduction we see that we can factor $x_1 - x_0$ from the first row of our reduced matrix, factor $x_2 - x_0$ from the second row, $x_3 - x_0$ from the third row, and so on til we get to the $n-1$th row of our reduced matrix where we will factor $x_{n-1} - x_0$. This gives us

$$\det(V(x_0, x_1, x_2, \cdots, x_{n-2}, x_{n-1})) = \prod_{i=1}^{n-1}(x_i - x_0) \times \begin{vmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-3} & x_1^{n-2} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-3} & x_2^{n-2} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{n-3} & x_{n-1}^{n-2} \end{vmatrix}.$$

Note that the remaining determinant is of size $(n-1) \times (n-1)$ and is of exactly the same form as the previous determinant. Thus using the same manipulations performed earlier we see that

$$\begin{aligned} \det(V) &= \prod_{i=1}^{n-1}(x_i - x_0) \times \det(V(x_1, x_2, \cdots, x_{n-1})) \\ &= \prod_{i=1}^{n-1}(x_i - x_0) \times \prod_{j=2}^{n-1}(x_j - x_1) \times \det(V(x_2, x_3, \cdots, x_{n-1})) \\ &= \prod_{i=1}^{n-1}(x_i - x_0) \times \prod_{j=2}^{n-1}(x_j - x_1) \times \prod_{k=3}^{n-1}(x_k - x_2)\det(V(x_3, \cdots, x_{n-1})) \\ &= \prod_{i=1}^{n-1}(x_i - x_0) \times \prod_{j=2}^{n-1}(x_j - x_1) \times \prod_{k=3}^{n-1}(x_k - x_2) \cdots \prod_{l=n-2}^{n-1}(x_l - x_{n-3}) \times (x_{n-1} - x_{n-2}). \end{aligned}$$

As a bit of shorthand notation the above can be written as

$$\det(V(x_0, x_1, \cdots, x_{n-2}, x_{n-1})) = \prod_{0 \le j < k \le n-1}(x_k - x_j),$$

as claimed in the book.

28.2 (Strassen's algorithm for matrix multiplication)

Exercise 28.2-6 (multiplying complex numbers)

We are given a, b, c, and d and we desire to compute the complex product

$$(a + ib)(c + id) = (ac - bd) + i(ad + bc),$$

using only three real multiplications. To perform this computation define the some intermediate variables t_1, t_2, and t_3 in terms of our inputs as

$$\begin{aligned} t_1 &= ad \\ t_2 &= bc \\ t_3 &= (a+b)(c-d)\,, \end{aligned}$$

which involve only *three* real multiplications. Note that the product obtained by computing t_3 is algebraically equivalent to

$$t_3 = (a+b)(c-d) = ac - ad + bc - bd = (ac - bd) + (-ad + bc)\,,$$

where we have grouped specific terms to help direct the manipulations we will perform below. Note also that the sums $t_1 + t_2$ and $t_3 + t_1 - t_2$ are given by

$$\begin{aligned} t_1 + t_2 &= ad + bc \\ t_3 + t_1 - t_2 &= ac - bd\,. \end{aligned}$$

so that our full complex product can be written in terms of these sums as

$$(a+ib)(c+id) = (t_3 + t_1 - t_2) + i(t_1 + t_2)\,.$$

28.3 (Solving systems of linear equations)

Exercise 28.3-1 (an example of forward substitution)

Our problem is to solve

$$\begin{bmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ -6 & 5 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 14 \\ -7 \end{bmatrix}\,,$$

which upon performing forward substitution gives

$$\begin{aligned} x_1 &= 3 \\ x_2 &= 14 - 4x_1 = 14 - 12 = 2 \\ x_3 &= -7 + 6x_1 - 5x_2 = -7 + 18 - 10 = 1\,. \end{aligned}$$

28.5 (Symmetric PD matrices and LS approximations)

Exercise 28.5-1 (diagonal elements of positive-definite matrices)

Since A is positive definite we must have for all non-zero x's the condition $x^T A x > 0$. Let $x = e_i$, a vector of all zeros but with a one in the i-th spot. Then $x^T A x = e_i^T A e_i = a_{ii} > 0$, proving that the diagonal elements of a positive definite matrix must be positive.

Chapter 31 (Number-Theoretic Algorithms)

31.1 (Elementary number-theoretic notations)

Exercise 31.1-1

Following the hint, given a sequence of increasing primes $p_1, p_2, \cdots, p_k$ we can form another prime p by computing "the product plus one" or $p = p_1 p_2 \cdots p_k + 1$. Note that this new number p is prime since none of $p_1, p_2, \cdots, p_k$ divide it. If one did then it would also have to divide the number $p - p_1 p_2 \cdots p_k = 1$ which is impossible. Thus the newly formed number is prime and as we can repeat this procedure an infinite number of times there must be an infinite number of primes.

Exercise 31.1-2

If $a|b$ then we have that there exists an integer k_1 such that $b = k_1 a$. Since $b|c$ we have that there exists an integer k_2 such that $c = k_2 b$. Using the first of these two relationships into the second we have

$$c = k_2 b = k_2(k_1 a) = k_1 k_2 a\,,$$

showing that $a|c$.

Exercise 31.1-3

Assume that $\gcd(k, p) = d \neq 1$. Then $d|k$ and $d|p$. Since p is prime if $d|p$ then $d = 1$ or $d = p$. As we assumed that $d \neq 1$ we must have $d = p$. The fact that $d|k$ means that $|d| \leq |k|$ or $|p| \leq |k|$. This last fact is a contraction to the problem assumption that $k < p$. Thus the assumption that $d \neq 1$ must be false.

Exercise 31.1-4

Since $\gcd(a, n) = 1$ then by Theorem 31.2 there must exist integers x and y such that $1 = ax + ny$. If we multiply this equation by b we get

$$b = abx + nbx\,.$$

Note that the right-hand-side of the above is divisible by n since it is a linear combination of two things (the terms abx and nbx) both of which are divisible by n. We know that abx is divisible by n since ab is and nbx is divisible by n since n is a factor. Since the right-hand-side equals b this shows that b is divisible by n.

Exercise 31.1-5

Using Equation 4 we can write

$$\binom{p}{k} = \frac{p}{k}\binom{p-1}{k-1},$$

which since the right-hand-side is a multiple of p shows that p divides $\binom{p}{k}$ or $p|\binom{p}{k}$. Next note that using the binomial theorem we can write $(a+b)^p$ as

$$(a+b)^p = a^p + b^p + \sum_{k=1}^{p-1} \binom{p}{k} a^k b^{p-k}.$$

Since p divides $\binom{p}{k}$ it divides the summation on the right-hand-side of the above equation. Thus the remainder of $(a+b)^p$ and $a^p + b^p$ when dividing by p is the same or

$$(a+b)^p = a^p + b^p \pmod{p}.$$

Exercise 31.1-6

Now the definition of the "mod" operators is that

$$\begin{aligned} x \bmod b &= x - \left\lfloor \frac{x}{b} \right\rfloor b \\ x \bmod a &= x - \left\lfloor \frac{x}{a} \right\rfloor a. \end{aligned} \tag{1}$$

Since $a|b$ there exists an integer k such that $b = ka$. Thus Equation 1 becomes

$$x \bmod b = x - \left\lfloor \frac{x}{b} \right\rfloor ka. \tag{2}$$

Thus taking the "mod" a operator to both sides of Equation 2 we get

$$(x \bmod b) \bmod a = x \bmod a,$$

as we were to show. Next if we assume that $x \equiv y \pmod{b}$ then taking the "mod" a operator to both sides of this expression and using the just derived result we get

$$(x \bmod b) \bmod a = (y \bmod b) \bmod a \quad \text{or} \quad x \bmod a = y \bmod a,$$

as we were to show.

Exercise 31.1-8

Recall that Theorem 31.2 states that that $\gcd(a,b)$ is the smallest positive element of the set $\{ax + by : x, y \in \mathbb{Z}\}$. Since this set definition of the greatest common divisor is symmetric in a and b we see that

$$\gcd(a,b) = \gcd(b,a).$$

and because x and y are arbitrary integers that

$$\gcd(a,b) = \gcd(-a,b) = \gcd(|a|,|b|)\,.$$

We also have that $\gcd(a,0)$ is the smallest element in the set $\{ax : x \in \mathbb{Z}\}$ which would be when $x = \pm 1$ such that the product of ax is positive or $|a|$. Next we have that $\gcd(a,ka)$ is the smallest element of the set $\{ax + kay : x,y \in \mathbb{Z}\} = \{a(x+ky) : x,y \in \mathbb{Z}\}$. This smallest element would be when $x + ky = \pm 1$ in order that $a(x+ky) = |a|$.

Exercise 31.1-9

Let $d = \gcd(a, \gcd(b,c))$ i.e. the left-hand-side of the equation we wish to show. Then $d|a$ and $d|\gcd(b,c)$. This second statement $d|\gcd(b,c)$ means that $d|b$ and $d|c$. Since d divides a and b by the above this means that $d|\gcd(a,b)$. Thus d is a number that $\gcd(a,b)$ and c. Lets show that d is the largest number that divides both $\gcd(a,b)$ and c. Then if so we have that $d = \gcd(\gcd(a,b),c)$ and we have shown the desired equivalence. To show this assume that there is another integer d' such that $d' > d$ that divides $\gcd(a,b)$ and c. This means that $d'|a$ and $d'|b$, and $d'|c$ so that $d'|\gcd(b,c)$. Thus since d' divides a and $\gcd(b,c)$ it must be smaller than or equal to the greatest common divisor of a and $\gcd(b,c)$ or

$$d' \leq d = \gcd(a, \gcd(b,c))\,,$$

giving a contradiction to the initial assumption that $d' > d$. Thus means d must be the largest integer already and so $d = \gcd(\gcd(a,b),c)$.

31.2 (Greatest common divisor)

Notes on Euclid's algorithm

Recall that $a \bmod b = a - \left\lfloor \frac{a}{b} \right\rfloor b$ i.e. $a \bmod b$ is the remainder when a is divided by b. This is the common understanding when $a > b$. We now ask what does this mean if $a < b$? In that case we would have $\left\lfloor \frac{a}{b} \right\rfloor = 0$ since the fraction $\frac{a}{b} < 1$ and the above formula would give

$$a \bmod b = a \quad \text{when} \quad a < b\,. \tag{3}$$

To numerical examples of this are

$$5 \bmod 10 = 5$$
$$2 \bmod 5 = 2\,.$$

The point of these comments is to give some understanding to Euclid's algorithm for finding the greatest common divisor when the inputs are $\text{EUCLID}(a,b)$ where $a < b$ then in this case the first recursive call performed is

$$\text{EUCLID}(a,b) = \text{EUCLID}(b, a \bmod b) = \text{EUCLID}(b,a)\,.$$

From the above we see that now all recursive calls to EUCLID will have the first argument larger than the second argument. In `python`, the greatest common divisor is available in the `fractions` module i.e.

```
import fractions
print fractions.gcd(30,21) # the example from the book gives 3
```

At the time of writing the source code of this function shows that its in fact using Euclid's algorithm for its implementation.

Exercise 31.2-1

Let $d = \gcd(a, b)$ then d must have a prime factorization in terms of the same primes as a and b namely

$$d = p_1^{g_1} p_2^{g_2} \cdots p_r^{g_r} .$$

Since d is a divisor of a and b we must have $d|a$ and $d|b$ which means that

$$p_i^{g_i}|a \quad \text{and} \quad p_i^{g_i}|b\,,$$

for $i = 1, 2, \cdots, r$. This in tern means that

$$p_i^{g_i}|p_i^{e_i} \quad \text{and} \quad p_i^{g_i}|p_i^{f_i} .$$

Thus we have that

$$g_i \leq e_i \quad \text{and} \quad g_i \leq f_i\,.$$

For d to be as large as possible (since it is the *greatest* common divisor the powers g_i must be as large as possible such that the above two relations hold or

$$g_i = \min(e_i, f_i)\,,$$

as we were to show.

Exercise 31.2-2

See the output in Table 3 where we follow the output format given in this section of the notes where as we move down the table we are moving in increasing stack depth. From that output we see that $\gcd(899, 493) = 29$ and that $29 = 899(-6) + 493(11)$.

Exercise 31.2-3

One way to see that these two expressions are equal is to use one step of EUCLID's algorithm to evaluate both of them and see that after the first step both of these algorithms are

stack level	a	b	$\lfloor a/b \rfloor$	d	x	y
0	899	493	1	29	-6	11
1	493	406	1	29	5	-6
2	406	87	4	29	-1	5
3	87	58	1	29	1	-1
4	58	29	2	29	0	1
5	29	0	-	29	1	0

Table 3: The recursive calls resulting from executing EXTENDED-EUCLID(899,493)

computing the same thing. For example, EUCLID to compute $\gcd(a, n)$ would first call (assuming $n \neq 0$)

$$\text{EUCLID}(n, a \bmod n)\,.$$

The first step of EUCLID's algorithm on the second expression $\gcd(a + kn, n)$ would call

$$\text{EUCLID}(n, (a + kn) \bmod n)\,.$$

Since $(a + kn) \bmod n = a \bmod n$ we have that the two expressions will evaluate to the same number and are therefore the same. If $n = 0$ the we can see that both sides are equal.

Exercise 31.2-4

One way to do this would be the following in `python`

```
def iterative_euclid(a,b):
    while b != 0 :
        a, b = b, a % b # swap in place
    return a
```

Here we have assigned b to a and $a \bmod b$ to b in one step using `pythons` tuple assignment.

Exercise 31.2-5

For the Fibonacci sequence defined as $F_0 = 0$, $F_1 = 1$, and

$$F_k = F_{k-1} + F_{k-2} \quad \text{for} \quad k \geq 2\,,$$

Then it can be shown that $F_{k+1} \sim \frac{\phi^{k+1}}{\sqrt{5}}$ for large k where ϕ is the "golden ratio" defined as

$$\phi = \frac{1 + \sqrt{5}}{2} = 1.61803\,.$$

From Lame's theorem in the book we have less than k recursive calls in EUCLID(a,b) when $F_{k+1} > b$ or using the approximation above for F_{k+1} when

$$\frac{\phi^{k+1}}{\sqrt{5}} > b\,.$$

Solving the above for k gives

$$k > \log_\phi(b) + \log_\phi(\sqrt{5}) - 1\,.$$

Note that we can evaluate the constants in the above using

$$\log_\phi(\sqrt{5}) = \frac{\ln(\sqrt{5})}{\ln(\phi)} = 1.67\,,$$

Thus we conclude that $k > \log_\phi(b) + 0.67$. Thus we have shown that the invocation EUCLID(a,b) makes at most $1 + \log_\phi(b)$ recursive calls.

Exercise 31.2-7

One can show that the function returns the same answer independent of the order of the arguments by induction. It is easiest to see how to write

$$\gcd(a_0, a_1, a_2, \cdots, a_n) = a_0x_0 + a_1x_1 + a_2x_2 + \cdots + a_nx_n\,,$$

by looking at an example. Consider

$$\gcd(a_0, a_1, a_2, a_3) = \gcd(a_0, \gcd(a_1, a_2, a_3)) = \gcd(a_0, \gcd(a_1, \gcd(a_2, a_3)))\,.$$

We can now use Theorem 31.2 to write $\gcd(a_2, a_3)$ as $a_2x_2 + a_3x_3$ for two integers x_2 and x_3. This would give

$$\gcd(a_0, a_1, a_2, a_3) = \gcd(a_0, \gcd(a_1, a_2x_2 + a_3x_3))\,.$$

We can again use use Theorem 31.2 to write $\gcd(a_1, a_2x_2 + a_3x_3)$ as

$$x_1a_1 + y_1(a_2x_2 + a_3x_3) = x_1a_1 + y_1x_2a_2 + y_1x_3a_3\,.$$

for two integers x_1 and y_1. Thus we have shown that

$$\gcd(a_0, a_1, a_2, a_3) = \gcd(a_0, x_1a_1 + y_1x_2a_2 + y_1x_3a_3)\,.$$

One more application of Theorem 31.2 gives

$$\begin{aligned}\gcd(a_0, a_1, a_2, a_3) &= x_0a_0 + y_2(x_1a_1 + y_1x_2a_2 + y_1x_3a_3) \\ &= x_0a_0 + y_2x_1a_1 + y_1y_2x_2a_2 + y_1y_2x_3a_3\,.\end{aligned}$$

As each coefficient in front of a_i is the product of integers it itself is an integer. This gives the desired representation when $n = 3$. In general, we use the definition of the greatest common divisor for $n > 1$ to "nest" gcd function calls until we get a function call with only two arguments $\gcd(a_{n-1}, a_n)$. We can then use Theorem 31.2 to write this as $a_{n-1}x_{n-1} + a_nx_n$ for two integers x_{n-1} and x_n. We can then "unnest" the gcd function calls each time using Theorem 31.2 to get the desired result.

$+_4$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

Table 4: The group operation table for the $(Z_4, +_4)$ group

$*_5$	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

Table 5: The group operation table for the $(Z_5^*, *_5)$ group

Exercise 31.2-8 (the least common multiple)

We can first compute the greatest common divisor of the given n integers

$$d = \gcd(a_1, a_2, \ldots, a_n) .$$

This can be done using the recursive definition of the gcd for more than two arguments given in a previous problem. Then since d is a divisor of each of a_i it is a factor of each of them (by definition). Once we have this number then the least common multiple is given by

$$\operatorname{lcm}(a_1, a_2, \ldots, a_n) = d \left(\frac{a_1}{d}\right) \left(\frac{a_2}{d}\right) \cdots \left(\frac{a_n}{d}\right) .$$

In the right-hand-side of the above expression by grouping d with a given term in parenthesis we have that $d\left(\frac{a_i}{d}\right) = a_i$ and thus the above number is a multiple of a_i for each i.

Exercise 31.3-1

The group $(Z_4, +_4)$ has the elements $\{0, 1, 2, 3\}$ with the addition operation "modulo four". The group operation table for this group is given in Table 4. The group $(Z_5^*, *_5)$ is the multiplication group of order n. The elements of that group are the elements of Z_5 that are relatively prime to five i.e.

$$Z_5^* = \{[a]_5 \in Z_5 : \gcd(a, 5) = 1\} .$$

The elements in Z_5^* are then $\{1, 2, 3, 4\}$. The group operation table is given in Table 5. To show that these two groups are isomorphic to each other we are looking for a mapping $\Sigma(\cdot)$ from $Z_4^+ = \{0, 1, 2, 3\}$ to the set $Z_5^* = \{1, 2, 3, 4\}$ such that $\Sigma(a)\Sigma(b) = \Sigma(c) \bmod 5$. Its clear that $\Sigma(0) = 1$ (i.e. the addition identity should map to the multiplication identity). To get

	1	2	4	3
1	1	2	4	3
2	2	4	3	1
4	4	3	1	2
3	3	1	2	4

Table 6: The table that results when we take the group operation table for $(Z_4, +_4)$ and transform each element under the $\Sigma(\cdot)$ mapping.

the rest of the mapping for Σ lets try $\Sigma(1) = 2$ and see what we can conclude if that is true. We have

$$\begin{aligned}
\Sigma(0) &= 1 \\
\Sigma(1) &= 2 \\
\Sigma(2) &= \Sigma(1+1) = \Sigma(1)^2 \bmod 5 = 2^2 \bmod 5 = 4 \bmod 5 = 4 \\
\Sigma(3) &= \Sigma(1+1+1) = \Sigma(1)^3 \bmod 5 = 2^3 \bmod 5 = 8 \bmod 5 = 3 \\
\Sigma(4 \bmod 4) &= \Sigma(0) = 1\,.
\end{aligned}$$

Thus we have found a one-to-one mapping that takes the elements and operation from $(Z_4, +_4)$ to $(Z_5^*, *_5)$. If we then take the elements of Table 4 and use the above mapping on each element we get Table 6. Notice that this table is really just Table 5 but with some rows and columns interchanged.

Exercise 31.3-2

For this problem we want to prove that a nonempty closed subset of a finite group is a subgroup. To do that we need to show the following

- **Closure:** We are told that the subset is closed therefore we have the closure property needed for a group immediately.
- **Identity:** As the subset S' is nonempty it must have at least one element. Denote this this element as a. If $a = e$ then S' has the identity element and we are done. Even if $a \neq e$ then as $a^{|S'|} = e$ (by Corollary 31.19 from the book) then since S' is closed the subset must have the identity element as a member in this case as well.
- **Associativity:** The associativity of the elements of the subset S' follow from that of the associativity of the elements in the original group S.
- **Inverses:** Let $a \in S'$. Then as we know that $e \in S'$ we claim that the inverse of a must also be in S'. We can see that using the fact that $a^{|S'|} = e$ or $a^{|S'|-1}a = e$. This later expression means that $a^{|S'|-1}$ is an inverse of the element a. As S' is closed the element $a^{|S'|-1}$ must be a member of S' so the set S' has all its elements inverses.

Exercise 31.3-3

If p is prime and e a positive integer then from the definition of the function ϕ we have

$$\phi(p^e) = p^e \prod_{p'|p^e} \left(1 - \frac{1}{p'}\right) ,$$

where the product is over all primes p' that divide p^e. The only prime that does divide p^e is p itself so the above becomes

$$\phi(p^e) = p^e \left(1 - \frac{1}{p}\right) = p^{e-1}(p-1) .$$

Exercise 31.3-4

To show this we will use Theorem 31.6 from the text "if both $\gcd(a,p) = 1$ and $\gcd(b,p) = 1$ then $\gcd(ab,p) = 1$". To use this note that since $a \in Z_n^*$ and $x \in Z_n^*$ they are both relatively prime to n and thus $\gcd(a,n) = 1$ and $\gcd(x,n) = 1$. Then using Theorem 31.6 we have that the product ax is relatively prime to n and thus is an element of Z_n^*. Thus the given function f maps each elements of Z_n^* to another element of Z_n^* and results in a permutation of Z_n^*.

Exercise 31.3-5

Since the group Z_9 has the nine elements $\{0,1,2,3,4,5,6,7,8\}$ then by Lagrange's theorem the number of elements in any subgroup of it must be a divisor of nine. That means that the subgroups must have 1, 3, or 9 elements. The subgroup with only one element is the subgroup with only the identity or $\{0\}$. The subgroup with nine elements is the original group again. The subgroup with three elements can be obtained by using 3 as a generator and is the subgroup $\{0,3,6\}$.

Now the group Z_{13}^* is the set of elements that are relatively prime to 13. Since 13 is a prime number the set Z_{13}^* is then given by

$$Z_{13}^* = \{1,2,3,4,5,6,7,8,9,10,11,12\}$$

Since this set has twelve elements by Lagrange's theorem any subset of it must have 1, 2, 3, 4, 6, or 12 elements itself. The subset with only one element is the set $\{1\}$ and the subset with 12 elements is the original set Z_{13}^* again.

The other subsets can be generated by taking various elements of Z_{13}^* to be a generator and then generating the subset from that element. We do that in the `python` code

`chap_31_prob_3_5.py`. Running that code we find that all the subsets of Z_{13}^* are

$$
\begin{aligned}
&< 1 > = \{1\} \\
&< 2 > =< 6 >=< 7 >=< 11 >= \{1,2,3,4,5,6,7,8,9,10,11,12\} = Z_{13}^* \\
&< 3 > =< 9 >= \{1,3,9\} \\
&< 4 > =< 10 >= \{1,3,4,9,10,12\} \\
&< 5 > =< 8 >= \{1,5,8,12\} \\
&< 12 > = \{1,12\}\,.
\end{aligned}
$$

Exercise 31.4-1

WWX: Proof from here down.

For this exercise we will follow the steps in the pseudo-code Modular-Linear-Equation-Solver "by hand", to find the solutions (if they exist). To start we note that we have

$$
\begin{aligned}
a &= 35 \\
b &= 10 \\
n &= 50\,.
\end{aligned}
$$

Then we let

$$d = \gcd(a,n) = \gcd(35,50) = 5\,.$$

Now we know that we can write d as

$$d = \gcd(35,50) = 5 = 35x' + 50y'\,.$$

If $x' = -7$ then $y' = 5$. Next we ask if $d|b$ or $5|10$ which is yes. Thus we compute

$$x_0 = -7\left(\frac{10}{5}\right) = -14\,.$$

Then for $i = 0,1,2,3,4$ we compute

$$x_i = x_0 + i\left(\frac{n}{d}\right) = -14 + i\left(\frac{50}{5}\right) = -14 + 10i\,.$$

We find for these values of i that

$$-14\,, -4\,, 6\,, 16\,, 26\,.$$

Note that we could add $d = 5$ as many times as possible to make these numbers positive if desired. If we run the code in `modula-linear-equation-solver` we get

Exercise 31.4-2

Assume by way of contradiction that $x \neq y \bmod n$ but that

$$ax = ay \bmod n\,.$$

Then since $x \neq y \bmod n$ we have $x = y + z$ with n not a factor of z or $n \nmid z$ or $2 \bmod n \neq 0$. Now if $x = y + z$ and we multiply by a we get

$$ax = ay + az\,,$$

and we consider this modulo n we get

$$ax \bmod n = ay \bmod n + az \bmod n\,.$$

Now as $ax \bmod n = ay \bmod n$ we have $0 = az \bmod n$. But $a \neq 0$ and $\gcd(a, n) = 1$ therefore we must have $z \bmod n = 0$ but that is a contradiction to the statement above. Thus

$$ax = ay \bmod n \Rightarrow x = y\,,$$

when $\gcd(a, n) = 1$.

We want to show $ax = ay \bmod n$ when $x \neq y$ when $\gcd(a, n) > 1$. Let $z = 2$, $a = 2$, and $n = 4$. Then $x = y + z = y + 2$. And $2x = 2y \bmod 4$ is true with $x = 2$ and $y = 0$ and $x \neq y \bmod 4$.

Problem 31.1 (The binary gcd algorithm)

Part (a): The statement

$$\gcd(a, b) = 2\gcd\left(\frac{a}{2}, \frac{b}{2}\right)\,,$$

follows from Corollary 31.4 from the book which states that for all integers a' and b' and any nonnegative integer n that

$$\gcd(a'n, b'n) = n\gcd(a', b')\,.$$

When we take $n = 2$, $a' = \frac{a}{2}$, and $b' = \frac{b}{2}$ we have the desired result.

Part (b): For this part we assume that a is odd and b is even. Then from Theorem 31.2 in the book we have that $\gcd(a, b)$ is the smallest positive element of the set $\{ax + by\}$ for all x and y from Z. This is the same as the smallest positive element of the set

$$\left\{ax + \left(\frac{b}{2}\right)y'\right\}$$

with $y' = 2y$. The above is equal to $\gcd\left(a, \frac{b}{2}\right)$.

Part (c): We will first show that if $a > b$ then

$$\gcd(a, b) = \gcd(a - b, b)\,.$$

To show this, recall that the left-hand-side is the smallest positive integer in the set $\{ax+by\}$ while the right-hand-side is the smallest postive integer in the set $\{(a-b)x'+by'\} = \{ax'+b(y'-x')\}$ which are the same number.

For this part we assume that a and b are both odd and that $a > b$. In that case $a-b$ is the difference of two odd numbers and so is an even number. From Part (a) we then have that

$$\gcd(a,b) = \gcd(a-b,b) = \gcd\left(\frac{a-b}{2}, b\right) .$$

If $b > a$ the situation is the same

$$\gcd(a,b) = \gcd(b-a,a) = \gcd\left(\frac{b-a}{2}, a\right) .$$

We can then use the fact that $\gcd(a',b') = \gcd(-a',b')$ to reverse the sign of the $\frac{b-a}{2}$ expression.

Appendix C (Counting and Probability)

Appendix C.1 (Counting)

a lower bound on the binomial coefficients (book notes page 1097)

The proof of the lower bound on the binomial coefficients given in the book, requires that

$$\frac{n-1}{k-1} \geq \frac{n}{k},$$

for $1 \leq k \leq n$. We can show this expression is true, by starting with it and seeing if we can convert it using reversible transformations to another expression known to be true. If this can be done, since each transformation is reversible, the original expression must also be true. For this expression we can demonstrate it is true by using some simple manipulations. We have

$$\begin{aligned} \frac{n-1}{k-1} &\geq \frac{n}{k} \\ k(n-1) &\geq n(k-1) \\ kn-k &\geq nk-n \\ -k &\geq -n \\ n &\geq k \end{aligned}$$

Which is known to be true. Using the same manipulations we can show show that

$$\frac{n-2}{k-2} \geq \frac{n}{k}.$$

This inductive process can continue, subtracting one each time from the numerator and denominator. As a check we can verify the correctness of the final inequality which is given by

$$\frac{n-k+1}{1} \geq \frac{n}{k}.$$

Using the same approach as above

$$\begin{aligned} \frac{n-k+1}{1} &\geq \frac{n}{k} \\ kn-k^2+k &\geq n \\ n(k-1)-k(k-1) &\geq 0 \\ (k-1)(n-k) &\geq 0 \end{aligned}$$

and the last equation is true. Given this sequence of results we can derive the lower bounds on the binomial coefficients as

$$\begin{aligned}\binom{n}{k} &= \frac{n(n-1)\dots(n-k+1)}{k(k-1)\dots 1} \\ &= \left(\frac{n}{k}\right)\left(\frac{n-1}{k-1}\right)\cdots\left(\frac{n-k+1}{1}\right) \\ &\geq \left(\frac{n}{k}\right)\left(\frac{n}{k}\right)\cdots\left(\frac{n}{k}\right) \\ &= \left(\frac{n}{k}\right)^k\end{aligned}$$

a upper bound on the binomial coefficients (book notes page 1097)

We have from the definition of the binomial coefficients that

$$\begin{aligned}\binom{n}{k} &= \frac{n(n-1)\cdots(n-k+1)}{k(k-1)\cdots 2\,1} \\ &\leq \frac{n^k}{k!} \\ &\leq \frac{e^k n^k}{k^k} \\ &\leq \left(\frac{en}{k}\right)^k .\end{aligned}$$

Where in the above we have used the $k! \geq \left(\frac{k}{e}\right)^k$ (obtained from Stirling's approximation), in the form

$$\frac{1}{k!} \leq \left(\frac{e}{k}\right)^k$$

to simplify the expression $\frac{n^k}{k!}$ in the above.

the binomial coefficients and the entropy function (book notes page 1097)

If we assign $k = \lambda n$, then we have the following

$$\begin{aligned}
\left(\begin{array}{c} n \\ \lambda n \end{array} \right) &\leq \frac{n^n}{(\lambda n)^{\lambda n}(n-\lambda n)^{n-\lambda n}} \\
&= \frac{n^n}{(\lambda n)^{\lambda n}((1-\lambda)n)^{(1-\lambda)n}} \\
&= \left(\frac{n}{(\lambda n)^{\lambda}((1-\lambda)n)^{1-\lambda}} \right)^n \\
&= \left(\frac{1}{\lambda^{\lambda}(1-\lambda)^{1-\lambda}} \right)^n \\
&= \left(\left(\frac{1}{\lambda} \right)^{\lambda} \left(\frac{1}{1-\lambda} \right)^{1-\lambda} \right)^n \\
&\equiv 2^{nH(\lambda)} ,
\end{aligned}$$

where the introduction of the expression $H(\lambda)$ above requires that

$$nH(\lambda) = n \lg(\left(\frac{1}{\lambda} \right)^{\lambda} \left(\frac{1}{1-\lambda} \right)^{1-\lambda})$$

or

$$H(\lambda) = -\lambda \lg(\lambda) - (1-\lambda) \lg(1-\lambda) .$$

This is the (binary) entropy function. We note before continuing that since $\lambda = n/k$ the above bound can also be written as

$$\left(\begin{array}{c} n \\ k \end{array} \right) \leq 2^{nH(\frac{n}{k})} .$$

Exercise C.1-1 (counting substrings)

Assuming $k < n$, then the first substring occupies the locations $1, 2, \ldots, k$, the second substring occupies the locations $2, 3, \ldots, k+1$, and so on. The last possible substring would occupy the locations $n-k+1, n-k, \ldots, n$. Thus we have in total

$$n - k + 1$$

substrings of size k in a string of size n.

To calculate how many substrings of size k a string of size n has, we from the above formula that we have n size one substrings ($k = 1$), $n-1$ size two substrings ($k = 2$), $n-2$ size three substrings, etc. Continuing this pattern the total number of substrings is the sum of all these numbers given by

$$\sum_{k=1}^{n} (n-k+1) = \sum_{l=1}^{n} l = \frac{n(n+1)}{2} .$$

Exercise C.1-2 (counting Boolean functions)

Each Boolean function is defined by how it maps its inputs to outputs. For an n-input, 1-output Boolean function we have a total of 2^n possible inputs, to which we can assign a TRUE or a FALSE output. Thus for each possible unique input specification we have 2 possible output specifications. So we have for the number of possible inputs (counting the number of possible outputs for each input)

$$2 * 2 \cdots 2 * 2 \,.$$

With one factor of 2 for each of the possible input specification. Since there are 2^n possible input specifications, this gives a total of

$$2^{2^n} \,,$$

possible Boolean functions with n-input variables and 1-output variable.

For a n-input m-output we have 2^m possible choices for each possible output, so using the same logic as before we have

$$(2^m)^{2^n} = 2^{m2^n} \,,$$

possible n-input m-output Boolean functions.

Exercise C.1-3 (counting professors)

Let the number of such ways our professors can sit be denoted by N_t, with t for "table". Then for each one of these seatings we can generate all permutations of n objects in the following way. First consider a specific table ordering of professors say *abcd*. Then all the other equivalent table orderings are given by circularly shifting the given ordering i.e. producing *bcda*, *cdab*, *dabc*. This circular shifts produce n different. Since by performing this operation on each of the N_t table orderings we get all possible permutations of which there are $n!$, so in equation form we have that nN_t must equal $n!$, which when we solve for N_t gives

$$N_t = \frac{n!}{n} = (n-1)! \,.$$

Exercise C.1-4 (an even sum from distinct subsets of size three)

Our set is given by $S = 1, 2, \ldots, 100$ and we want three distinct numbers that sum to an even number. The total number of distinct size 3 subsets (independent of the order of the elements in the subset) drawn from S is given by $\begin{pmatrix} 100 \\ 3 \end{pmatrix}$. Half of these will sum to an even number and the other half will sum to an odd number. Thus the total number is given by

$$\frac{1}{2}\begin{pmatrix} 100 \\ 3 \end{pmatrix} = 80850 \,.$$

Another way to obtain this same number is to recognize that to have an even sum I must pick either three even numbers or one even number and two odd numbers. Since these two outcomes are mutually exclusive using the rule of sum the total number of ways to pick a even summable subset is the sum of the number of ways to select each of the previous sets. This number is

$$\left(\begin{array}{c} 50 \\ 3 \end{array} \right) + \left(\begin{array}{c} 50 \\ 1 \end{array} \right) \left(\begin{array}{c} 50 \\ 2 \end{array} \right) ,$$

where the first combination is the number of ways to select three even numbers and the second combination product is the number of ways to select a single odd number and then two even numbers. One can check that the sum above reduces to 80850 also.

Exercise C.1-5 (factoring a fraction from the binomial coefficients)

Using the definition of the binomial coefficient, we have the following

$$\left(\begin{array}{c} n \\ k \end{array} \right) = \frac{n!}{k!(n-k)!} = \frac{n(n-1)!}{k(k-1)!(n-1-(k-1))!} = \frac{n}{k} \left(\begin{array}{c} n-1 \\ k-1 \end{array} \right) . \tag{4}$$

Exercise C.1-6 (factoring another fraction from the binomial coefficients)

Using the definition of the binomial coefficient, we have the following

$$\left(\begin{array}{c} n \\ k \end{array} \right) = \frac{n!}{(n-k)!k!} = \frac{n(n-1)!}{(n-k)(n-k-1)!k!} = \frac{n}{n-k} \left(\begin{array}{c} n-1 \\ k \end{array} \right) .$$

Exercise C.1-7 (choosing k subsets from n by drawing subsets of size $k-1$)

Considering the group of n objects with one object specified as distinguished or "special". Then the number of ways to select k objects from n can be decomposed into two distinct occurrences. The times when this "special" object *is* selected in the subset of size k and the times when its *not*. When it is *not* selected in the subset of size k we are specifying our k subset elements from the $n-1$ remaining elements giving $\left(\begin{array}{c} n-1 \\ k \end{array} \right)$ total subsets in this case. When it *is* selected into the subset of size k we have to select $k-1$ other elements from the $n-1$ remaining elements, giving $\left(\begin{array}{c} n-1 \\ k-1 \end{array} \right)$ additional subsets in this case. Summing the counts from these two occurrences we have that factorization can be written as the following

$$\left(\begin{array}{c} n \\ k \end{array} \right) = \left(\begin{array}{c} n-1 \\ k \end{array} \right) + \left(\begin{array}{c} n-1 \\ k-1 \end{array} \right) .$$

Exercise C.1-8 (Pascal's triangle)

We have (evaluating our binomial coefficients as we place them in the table)

$$\begin{array}{ccccccccccc}
 & & & & 1 & & & & & & \\
 & & & 1 & & 1 & & & & & \\
 & & 1 & & 2 & & 1 & & & & \\
 & 1 & & 3 & & 3 & & 1 & & & \\
1 & & 4 & & 6 & & 4 & & 1 & & \\
 & 5 & & 10 & & 10 & & 5 & & 1 & \\
6 & & 15 & & 20 & & 15 & & 6 & & 1
\end{array}$$

Note that the top row only has one element $\begin{pmatrix} 0 \\ 0 \end{pmatrix} \equiv 1$. The second row has two elements $\begin{pmatrix} 1 \\ 0 \end{pmatrix} = 1$, and $\begin{pmatrix} 1 \\ 1 \end{pmatrix} = 1$. Subsequent rows are obtained by beginning and ending with a one and summing the values of the two binomial coefficients in the row above.

Exercise C.1-9 (sum of the integers up to n)

Expressing each side of this expression, we have that the left hand side is given by

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

while the right hand side is given by

$$\begin{pmatrix} n+1 \\ 2 \end{pmatrix} = \frac{n(n+1)}{2}$$

since the two sides are equal we have the proven identity.

Exercise C.1-10 (maximum of $\begin{pmatrix} n \\ k \end{pmatrix}$ as a function of k)

Because the binomial coefficients satisfy

$$\begin{pmatrix} n \\ k \end{pmatrix} = \begin{pmatrix} n \\ n-k \end{pmatrix} \quad \text{for} \quad 0 \leq k \leq n$$

as a function of k the values of the binomial coefficients repeat once $k \geq \frac{n}{2}$. A specific example of this can be seen in the Pascal triangle construction in Exercise C.1-8. If we can show that $\begin{pmatrix} n \\ k \end{pmatrix}$ is an increasing function of k, the k value that will maximize the binomial coefficient will be the one corresponding to $k \approx \frac{n}{2}$. If n is even this maximum will occur the value floor($\frac{n}{2}$) while if n is odd this maximum will be found at floor($\frac{n}{2}$) and ceiling($\frac{n}{2}$).

All one has to do is show that $\left(\begin{array}{c} n \\ k \end{array} \right)$ is an increasing function of k, which means proving the following chain of reasoning is reversible

$$\begin{aligned}
\left(\begin{array}{c} n \\ k+1 \end{array} \right) &> \left(\begin{array}{c} n \\ k \end{array} \right) \\
\frac{n!}{(k+1)!(n-k-1)!} &> \frac{n!}{(n-k)!k!} \\
\frac{(n-k)!}{(n-k-1)!} &> \frac{(k+1)!}{k!} \\
n-k &> k+1 \\
n &> 2k+1 \\
k &< \frac{n-1}{2}
\end{aligned}$$

which says that $\left(\begin{array}{c} n \\ k \end{array} \right)$ is increasing as a function of k as long as $k < \frac{n-1}{2}$, which equals the floor and ceiling expressions above if n is odd or even respectively.

Exercise C.1-11 (a product upper bounds on binomial coefficients)

We want to show that for any $n \geq 0$, $j \geq 0$, $k \geq 0$ and $j + k \leq n$ that

$$\left(\begin{array}{c} n \\ j+k \end{array} \right) \leq \left(\begin{array}{c} n \\ j \end{array} \right) \left(\begin{array}{c} n-j \\ k \end{array} \right) .$$

To show this using an algebraic proof, we will evaluate the left hand side of this expression, convert it into the right hand side multiplied by a correction factor and then show that the correction factor is less than one. To begin we have

$$\begin{aligned}
\left(\begin{array}{c} n \\ j+k \end{array} \right) &= \frac{n!}{(j+k)!(n-j-k)!} \\
&= \frac{n!}{j!(n-j)!} \frac{j!(n-j)!}{(j+k)!(n-j-k)!} \\
&= \left(\begin{array}{c} n \\ j \end{array} \right) \frac{(n-j)!}{k!(n-j-k)!} \frac{j!k!}{(j+k)!} \\
&= \left(\begin{array}{c} n \\ j \end{array} \right) \left(\begin{array}{c} n-j \\ k \end{array} \right) \frac{(j(j-1)(j-2)\cdots 2\,1)(k(k-1)\cdots 2\,1)}{(j+k)(j+k-1)(j+k-2)\cdots 21}
\end{aligned}$$

Note that the top and bottom of the factor multiplying the terms $\left(\begin{array}{c} n \\ j \end{array} \right) \left(\begin{array}{c} n-j \\ k \end{array} \right)$ both have $j + k$ elements in their product. With out loss of generality we will assume that $j \leq k$. The product above can be written as

$$\begin{aligned}
&\frac{(j(j-1)(j-2)\cdots 2\,1)(k(k-1)\cdots 2\,1)}{(j+k)(j+k-1)(j+k-2)\cdots 21} = \frac{j!k!}{(k+j)(k+j-1)\cdots(k+1)k!} \\
= \quad &\frac{j(j-1)(j-2)\cdots 21}{(j+k)(j+k-1)(j+k-2)\cdots(k+2)(k+1)} < 1 ,
\end{aligned}$$

since for every factor in the numerator, we can find a term in the denominator that is larger than it, for example the ratio of the first product in the numerator and denominator above satisfy

$$\frac{j}{j+k} < 1\,.$$

As the combined correction factor (with all these ratios) is less than one, we have shown the inequality we started with.

As a combinatorial proof, the number of ways to select $j+k$ items from a set of n will be smaller than if we first select j items from n, and for each of those sets select k items from $n-j$. That the former is a larger number can be seen by the following example where equality does not hold. Consider the case $n=4$, $k=3$, and $j=1$, then

$$\left(\begin{array}{c} 4 \\ 4 \end{array}\right) = 1 < \left(\begin{array}{c} 4 \\ 1 \end{array}\right)\left(\begin{array}{c} 3 \\ 3 \end{array}\right) = 4\,.$$

There the number of ways to select 4 items from 4 is only one which is smaller than if we are allowed to select 1 of the 4 items first (of which there are four ways) and then combine this with the number of ways to select three remaining elements (of which there is only one).

Exercise C.1-12 (a algebraic upper bound on the binomial coefficients)

We desire to prove by induction that

$$\left(\begin{array}{c} n \\ k \end{array}\right) \leq \frac{n^n}{k^k(n-k)^{n-k}}\,.$$

Assuming that $0^0 = 1$, we can show that $k=0$ satisfies this easily since it reduces to $1 \leq 1$. However, to begin with a more realistic case, to anchor our induction proof to, we start with $k=1$. This value gives $\left(\begin{array}{c} n \\ 1 \end{array}\right) = n$ for the left hand side of this expression, while the right hand-side evaluates to

$$\frac{n^n}{(n-1)^{n-1}}\,.$$

This can be simplified as follows

$$\begin{aligned} \frac{n^n}{(n-1)^{n-1}} &= n\left(\frac{n^{n-1}}{(n-1)^{n-1}}\right) \\ &= n\left(\frac{n}{n-1}\right)^{n-1} \\ &= n\left(\frac{1}{1-\frac{1}{n}}\right)^{n-1} \end{aligned}$$

Now since we assume that $n > 1$, the expression $1 - \frac{1}{n}$ is bounded by

$$0 < 1 - \frac{1}{n} < 1$$

and more specifically

$$1 < \frac{1}{1-\frac{1}{n}}\,.$$

Taking positive powers of this expression (i.e. to the $n-1$ power) can only increase its value and we have that

$$n < n\left(\frac{1}{1-\frac{1}{n}}\right)^{n-1}$$

which provides the anchoring point from which to begin mathematical induction. Having shown that this inequality is true for $k=1$ we next proceed to assume it is true for $\hat{k} < k$ and show that it is true for $\hat{k} < k+1$. To do this consider the following

$$\begin{aligned}
\left(\begin{array}{c} n \\ k+1 \end{array}\right) &= \frac{n!}{(n-k-1)!(k+1)!} \\
&= \frac{n!}{(n-k)!k!}\left(\frac{n-k}{k+1}\right) \\
&= \left(\begin{array}{c} n \\ k \end{array}\right)\left(\frac{n-k}{k+1}\right) \\
&\leq \frac{n^n}{k^k(n-k)^{n-k}}\left(\frac{n-k}{k+1}\right) = \frac{n^n}{k^k(k+1)(n-k)^{n-k-1}}\,.
\end{aligned}$$

Here on the last step we have used the induction hypothesis. Our induction proof will be finished if we can show that the last expression above is less than the following

$$\frac{n^n}{(k+1)^{k+1}(n-k-1)^{n-k-1}}\,.$$

Towards this end we will show this by *assuming* that it is true and then through reversible transformations reduce this expression to one that is known to be true. Since all transformation are reversible the original inequality must be true. So to begin we assume that

$$\begin{aligned}
\frac{n^n}{k^k(k+1)(n-k)^{n-k-1}} &\leq \frac{n^n}{(k+1)^{k+1}(n-k-1)^{n-k-1}} \\
\frac{(k+1)^k}{k^k} &\leq \frac{(n-k)^{n-k-1}}{(n-k-1)^{n-k-1}} \\
\left(1+\frac{1}{k}\right)^k &\leq \left(1+\frac{1}{n-k-1}\right)^{n-k-1}
\end{aligned}$$

From the above if we define the function $f(x)$ as

$$f(x) = \left(1+\frac{1}{x}\right)^x$$

The above expression becomes the following functional relationship, which if we can prove is true, will complete the inductive proof

$$f(k) \leq f(n-k-1)\,.$$

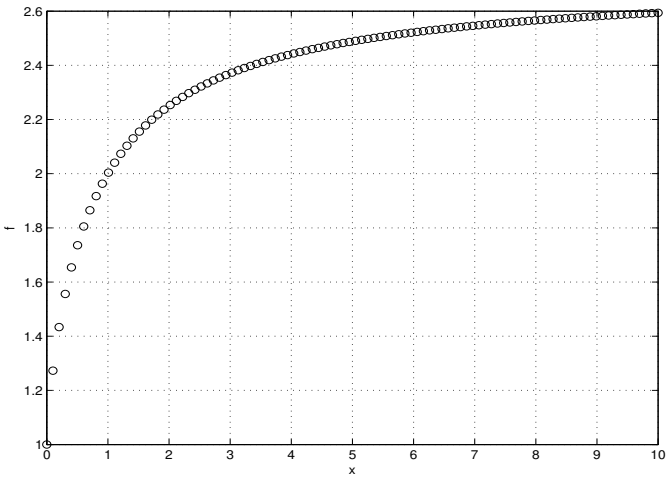

Figure 3: Graphical plot showing the monotonicity of $f(x)$ required for the inductive proof in Exercise C.1-12.

Further assuming that f has a monotone inverse (to be discussed below) then we can apply the inverse to this expression to obtain the region of validity for this inequality as

$$k \leq n - k - 1$$

or $k \leq \frac{n-1}{2}$. For all the statements above to be valid (and our induction proof to be complete) all one must do now is to show that $f(x)$ has an inverse or equivalently that $f(x)$ is monotonic over its domain. This could be done with derivatives but rather than that Figure 3 shows a plot of $f(x)$ showing its monotonic behavior.

Note also from the symmetry relationship possessed by the binomial coefficients, i.e,

$$\left(\begin{array}{c} n \\ k \end{array} \right) = \left(\begin{array}{c} n \\ n-k \end{array} \right)$$

we can extend our result above to all $0 \leq k \leq n$.

Exercise C.1-13 (Stirling's approximate for some binomial coefficients)

We desire to show that

$$\left(\begin{array}{c} 2n \\ n \end{array} \right) = \frac{2^{2n}}{\sqrt{\pi n}} \left(1 + O(1/n)\right)$$

using Stirling's approximation. From the definition of the binomial coefficients we know that

$$\left(\begin{array}{c} 2n \\ n \end{array} \right) = \frac{(2n)!}{n!\, n!} .$$

Now Stirling's approximation gives an estimate of $n!$. Its expression is

$$n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \theta(\frac{1}{n})\right) .$$

Therefore substituting $2n$ for n we have an approximation to $(2n)!$ given by

$$(2n)! = \sqrt{4\pi n}\left(\frac{2n}{e}\right)^{2n}\left(1+\theta(\frac{1}{n})\right).$$

We also need an expression for $(n!)^2$, which is given by

$$\begin{aligned}(n!)^2 &= 2\pi n\left(\frac{n}{e}\right)^{2n}\left(1+\theta(\frac{1}{n})\right)^2 \\ &= 2\pi n\left(\frac{n}{e}\right)^{2n}\left(1+\theta(\frac{1}{n})\right).\end{aligned}$$

Thus we have that

$$\begin{aligned}\left(\begin{array}{c}2n\\ n\end{array}\right) &= \frac{\sqrt{4\pi n}\left(\frac{2n}{e}\right)^{2n}\left(1+\theta(\frac{1}{n})\right)}{2\pi n\left(\frac{n}{e}\right)^{2n}\left(1+\theta(\frac{1}{n})\right)} \\ &= \frac{1}{\sqrt{\pi n}}\left(\frac{2n}{n}\right)^{2n}\frac{1+\theta(\frac{1}{n})}{1+\theta(\frac{1}{n})} \\ &= \frac{2^{2n}}{\sqrt{\pi n}}\left(1+\theta(\frac{1}{n})\right)\left(1+\theta(\frac{1}{n})\right) \\ &= \frac{2^{2n}}{\sqrt{\pi n}}\left(1+\theta(\frac{1}{n})\right).\end{aligned}$$

As was to be shown. In deriving this result we have made repeated use of the algebra of order symbols.

Exercise C.1-14 (the maximum of the entropy function $H(\lambda)$)

The entropy function is given by

$$H(\lambda) = -\lambda\lg(\lambda) - (1-\lambda)\lg(1-\lambda).$$

Remembering the definition of $\lg(\cdot)$ in terms of the natural logarithm $\ln(\cdot)$ as

$$\lg(x) = \frac{\ln(x)}{\ln(2)}$$

we have that the derivative of the $\lg(\cdot)$ function is given by

$$\frac{\lg(x)}{dx} = \frac{1}{x\ln(2)}.$$

Using this expression we can take the derivative of the binary entropy function giving

$$\begin{aligned}H'(\lambda) &= -\lg(\lambda) - \frac{\lambda}{\lambda\ln(2)} + \lg(1-\lambda) + \frac{(1-\lambda)}{(1-\lambda)\ln(2)} \\ &= -\lg(\lambda) + \lg(1-\lambda).\end{aligned}$$

When we set this equal to zero looking for an extrema we have that

$$-\lg(\lambda) + \lg(1-\lambda) = 0$$

which has as its solution $\lambda = \frac{1}{2}$. We can check that this point is truly a maximum and not a minimum by computing the second derivative of $H(\lambda)$. We have that

$$H''(\lambda) = -\frac{1}{\lambda \ln(2)} - \frac{1}{(1-\lambda)\ln(2)}\,,$$

which is negative when $\lambda = \frac{1}{2}$ implying a maximum. We also have that

$$H(\frac{1}{2}) = \frac{1}{2} + \frac{1}{2} = 1\,.$$

Exercise C.1-15 (moment summing of binomial coefficients)

Consider the binomial expansion learned in high school

$$(x+y)^n = \sum_{k=0}^{n} \left(\begin{array}{c} n \\ k \end{array} \right) x^k y^{n-k}\,.$$

Taking the derivative of this expression with respect to x gives the following

$$n(x+y)^{n-1} = \sum_{k=0}^{n} \left(\begin{array}{c} n \\ k \end{array} \right) k x^{k-1} y^{n-k}\,,$$

evaluating at $x = y = 1$, then gives

$$n2^{n-1} = \sum_{k=0}^{n} \left(\begin{array}{c} n \\ k \end{array} \right) k\,,$$

as was desired to be shown.

Appendix C.2 (Probability)

Exercise C.2-1 (Boole's inequality)

We begin by decomposing the countable union of sets A_i i.e.

$$A_1 \cup A_2 \cup A_3 \ldots\,,$$

into a countable union of disjoint sets C_j. Define these disjoint sets as

$$\begin{aligned}
C_1 &= A_1 \\
C_2 &= A_2 \backslash A_1 \\
C_3 &= A_3 \backslash (A_1 \cup A_2) \\
C_4 &= A_4 \backslash (A_1 \cup A_2 \cup A_3) \\
&\vdots \\
C_j &= A_j \backslash (A_1 \cup A_2 \cup A_3 \cup \cdots \cup A_{j-1})
\end{aligned}$$

Then by construction

$$A_1 \cup A_2 \cup A_3 \cdots = C_1 \cup C_2 \cup C_3 \cdots ,$$

and the C_j's are disjoint, so that we have

$$\Pr(A_1 \cup A_2 \cup A_3 \cup \cdots) = \Pr(C_1 \cup C_2 \cup C_3 \cup \cdots) = \sum_j \Pr(C_j) .$$

Since $\Pr(C_j) \leq \Pr(A_j)$, for each j, this sum is bounded above by

$$\sum_j \Pr(A_j) ,$$

and Boole's inequality is proven.

Exercise C.2-2 (more head for professor Rosencrantz)

For this problem we can explicitly enumerate our sample space and then count the number of occurrences where professor Rosencrantz obtains more heads than professor Guildenstern. Denoting the a triplet of outcomes from the one coin flip by Rosencrantz and the two coin flips by Guildenstern as (R, G_1, G_2), we see that the total sample space for this experiment is given by

$$\begin{array}{cccc} (H,H,H) & (H,H,T) & (H,T,H) & (H,T,T) \\ (T,H,H) & (T,H,T) & (T,T,H) & (T,T,T) \end{array}$$

From which the only outcome where professor Rosencrantz obtains more heads than professor Guildenstern is the sample (H, T, T). Since this occurs once from eights possible outcomes, Rosencrantz has a $1/8$ probability of winning.

Exercise C.2-3 (drawing three ordered cards)

There are $10 \cdot 9 \cdot 8 = 720$ possible ways to draw three cards from ten when the order of the drawn cards matters. To help motivate how to calculate then number of *sorted* three card draws possible, we'll focus how the number of ordered draws depends on the numerical value of the *first* card drawn. For instance, if a ten or a nine is drawn as a first card there are no possible ways to draw two other cards and have the total set ordered. If an eight is drawn as the first card, then it is possible to draw a nine for the second and a ten for the third producing the ordered set $(8, 9, 10)$. Thus if an eight is the first card drawn there is one possible sorted hand. If a seven is the first card drawn then we can have second and third draws consisting of $(8, 9)$, $(8, 10)$, or a $(9, 10)$ and have an ordered set of three cards. Once the seven is drawn we now are looking for the number of ways to draw two sorted cards from $\{8, 9, 10\}$. So if an seven is drawn we have three possible sorted hands. Continuing, if a six is drawn as the first card we have possible second and third draws of: $(7, 8)$, $(7, 9)$, $(7, 10)$, $(8, 9)$, $(8, 10)$, or $(9, 10)$ to produce an ordered set. Again we are looking for a way to draw two sorted cards from a list (this time $\{7, 8, 9, 10\}$).

After we draw the first card an important subcalculation is to compute the number of ways to draw *two* sorted cards from a list. Assuming we have n ordered items in our list, then we have n ways to draw the first element and $n-1$ ways to draw the second element giving $n(n-1)$ ways to draw two elements. In this pair of elements *one* ordering result in a correctly sorted list (the other will not). Thus half of the above are incorrectly ordered what remains are

$$N_2(n) = \frac{n(n-1)}{2}$$

The number of total ways to draw three sorted cards can now be found. If we first draw an eight we have two cards from which to draw the two remaining ordered cards in $N_2(2)$ ways. If we instead first drew a seven we have three cards from which to draw the two remaining ordered cards in $N_2(3)$ ways. If instead we first drew a six then we have four cards from which to draw two ordered cards in $N_2(4)$ ways. Continuing, if we first draw a one, we have nine cards from which to draw two ordered cards in $N_2(9)$ ways. Since each of these options is mutually exclusive the total number of ways to draw three ordered cards is given by

$$N_2(3) + N_2(4) + N_2(5) + \cdots + N_2(9) = \sum_{n=2}^{9} N_2(n) = \sum_{n=2}^{9} \frac{n(n-1)}{2} = 120\,.$$

Finally with this number we can calculate the probability of drawing three ordered cards as

$$\frac{120}{720} = \frac{1}{6}\,.$$

Exercise C.2-4 (simulating a biased coin)

Following the hint, since $a < b$, the quotient a/b is less than one. If we express this number in binary we will have an (possibly infinite) sequence of zeros and ones. The method to determine whether to return a biased "heads" (with probability a/b) or a biased "tails" (with probability $(b-a)/b$) is best explained with an example. To get a somewhat interesting binary number, lets assume an example where $a = 76$ and $b = 100$, then our ratio a/b is 0.76. In binary that has a representation given by (see the Mathematica file `exercise_C_2_4.nb`)

$$0.76 = 0.1100001010001111011_2$$

We then begin flipping our unbiased coin. To a "head" result we will associate a one and to a tail result we shall associate a zero. As long as the outcome of the flipped coin matches the sequence of ones and zeros in the binary expansion of a/b, we continue flipping. At the point where the flips of our unbiased coins diverges from the binary sequence of a/b we stop. If the divergence produces a sequence that is *less than* the ratio a/b we return this result by returning a biased "head" result. If the divergence produces a sequence that is *greater than* the ratio a/b we return this result by returning a biased "tail" result. Thus we have used our fair coin to determine where a sequence of flips "falls" relative to the ratio of a/b.

The expected number of coin flips to determine our biased result will be the expected number of flips required until the flips from the unbiased coin don't match the sequence of zeros and ones in the binary expansion of the ratio a/b. If we assume that a success is when our

unbiased coin flip *does not* match the digit in the binary expansion of a/b. The for each flip a success can occur with probability $p = 1/2$ and a failure can occur with probability $q = 1 - p = 1/2$, we have that the number of flips n needed before a "success" is a geometric random variable with parameter $p = 1/2$. This is that

$$P\{N = n\} = q^{n-1}p\,.$$

So that the expected number of flips required for a success is then the expectation of the geometric random variable and is given by

$$E[N] = \frac{1}{p} = 2\,,$$

which is certainly $O(1)$.

Exercise C.2-5 (closure of conditional probabilities)

Using the definition of conditional probability given in this section we have that

$$P\{A|B\} = \frac{P\{A \cap B\}}{P\{B\}}\,,$$

so that the requested sum is then given by

$$\begin{aligned} P\{A|B\} + P\{\bar{A}|B\} &= \frac{P\{A \cap B\}}{P\{B\}} + \frac{P\{\bar{A} \cap B\}}{P\{B\}} \\ &= \frac{P\{A \cap B\} + P\{\bar{A} \cap B\}}{P\{B\}} \end{aligned}$$

Now since the events $A \cap B$ and $\bar{A} \cap B$ are mutually exclusive the above equals

$$\frac{P\{(A \cap B) \cup (\bar{A} \cap B)\}}{P\{B\}} = \frac{P\{(A \cup \bar{A}) \cap B\}}{P\{B\}} = \frac{P\{B\}}{P\{B\}} = 1\,.$$

Exercise C.2-6 (conditioning on a chain of events)

This result follows for the two set case $P\{A \cap B\} = P\{A|B\}P\{B\}$ by grouping the sequence of A_i's in the appropriate manner. For example by grouping the intersection as

$$A_1 \cap A_2 \cap \cdots \cap A_{n-1} \cap A_n = (A_1 \cap A_2 \cap \cdots \cap A_{n-1}) \cap A_n$$

we can apply the two set result to obtain

$$P\{A_1 \cap A_2 \cap \cdots \cap A_{n-1} \cap A_n\} = P\{A_n|A_1 \cap A_2 \cap \cdots \cap A_{n-1}\}\, P\{A_1 \cap A_2 \cap \cdots \cap A_{n-1}\}\,.$$

Continuing now to peal A_{n-1} from the set $A_1 \cap A_2 \cap \cdots \cap A_{n-1}$ we have the second probability above equal to

$$P\{A_1 \cap A_2 \cap \cdots \cap A_{n-2} \cap A_{n-1}\} = P\{A_{n-1}|A_1 \cap A_2 \cap \cdots \cap A_{n-2}\}P\{A_1 \cap A_2 \cap \cdots \cap A_{n-2}\}\,.$$

Continuing to peal off terms from the back we eventually obtain the requested expression i.e.

$$\begin{aligned} P\{A_1 \cap A_2 \cap \cdots \cap A_{n-1} \cap A_n\} &= P\{A_n | A_1 \cap A_2 \cap \cdots \cap A_{n-1}\} \\ &\times P\{A_{n-1} | A_1 \cap A_2 \cap \cdots \cap A_{n-2}\} \\ &\times P\{A_{n-2} | A_1 \cap A_2 \cap \cdots \cap A_{n-3}\} \\ &\vdots \\ &\times P\{A_3 | A_1 \cap A_2\} \\ &\times P\{A_2 | A_1\} \\ &\times P\{A_1\} . \end{aligned}$$

Exercise C.2-7 (pairwise independence does not imply independence)

Note: As requested by the problem I was *not* able to find a set of events that are pairwise independent but such that no subset $k > 2$ of them are mutually independent. Instead I found a set of events that are all pairwise independent but that are not mutually independent. If you know of a solution to the former problem please contact me.

Consider the situation where we have n distinct people in a room. Let $A_{i,j}$ be the event that person i and j have the same birthday. We will show that any two of these events are pairwise independent but the totality of events $A_{i,j}$ are not mutually independent. That is we desire to show that the two events $A_{i,j}$ and $A_{r,s}$ are independent but the totality of all $\begin{pmatrix} n \\ 2 \end{pmatrix}$ events are not independent. Now we have that

$$P(A_{i,j}) = P(A_{r,s}) = \frac{1}{365} ,$$

since for the specification of either one persons birthday the probability that the other person will have that birthday is $1/365$. Now we have that

$$P(A_{i,j} \cap A_{r,s}) = P(A_{i,j} | A_{r,s}) P(A_{r,s}) = \left(\frac{1}{365}\right)\left(\frac{1}{365}\right) = \frac{1}{365^2} .$$

This is because $P(A_{i,j}|A_{r,s}) = P(A_{i,j})$ i.e. the fact that persons r and s have the same birthday has no effect on whether the persons i and j have the same birthday. This is true even if one of the people in the pairs (i, j) and (r, s) is the same. When we consider the intersection of *all* the sets $A_{i,j}$, the situation changes. This is because the event $\cap_{(i,j)} A_{i,j}$ (where the intersection is over all pairs (i, j)) is the event that *every* pair of people have the same birthday, i.e. that everyone considered has the same birthday. This will happen with probability

$$\left(\frac{1}{365}\right)^{n-1} ,$$

while if the events $A_{i,j}$ were independent the required probability would be

$$\prod_{(i,j)} P(A_{i,j}) = \left(\frac{1}{365}\right)^{\binom{n}{2}} = \left(\frac{1}{365}\right)^{\frac{n(n-1)}{2}} .$$

Since $\binom{n}{2} \neq n-1$, these two results are not equal and the totality of events $A_{i,j}$ are not independent.

Exercise C.2-8 (conditional but not independence)

Consider the situation where we have to coins. The first coin C_1 is biased and has probability p of landing heads when flipped where $p > 1/2$. The second coin C_2 is fair and has probability of landing heads $1/2$. For our experiment one of the two coins will be selected (uniformly and at random) and presented to two people who will each flip this coin. We will assume that the coin that is selected is not known. Let H_1 be the event that the first flip lands heads, and let H_2 be the event that the second flip lands heads. We can show that the events H_1 and H_2 are not independent by computing $P(H_1)$, $P(H_2)$, and $P(H_1, H_2)$ by conditioning on the coin selected. We have for $P(H_1)$

$$\begin{aligned} P(H_1) &= P(H_1|C_1)P(C_1) + P(H_1|C_2)P(C_2) \\ &= p\frac{1}{2} + \frac{1}{2}\frac{1}{2} \\ &= \frac{1}{2}(p + \frac{1}{2}) , \end{aligned}$$

Here C_1 is the event that we select the first (biased) coin while C_2 is the event we select the second (unbiased) coin. In the same way we find that

$$P(H_2) = \frac{1}{2}(p + \frac{1}{2}) ,$$

while $P(H_1, H_2)$ is given by

$$\begin{aligned} P(H_1, H_2) &= P(H_1, H_2|C_1)P(C_1) + P(H_1, H_2|C_2)P(C_2) \\ &= p^2\frac{1}{2} + \frac{1}{4}\frac{1}{2} \\ &= \frac{1}{2}(p^2 + \frac{1}{4}) . \end{aligned}$$

The events H_1 and H_2 will *only* be independent if $P(H_1)P(H_2) = P(H_1, H_2)$ or

$$\frac{1}{4}\left(p + \frac{1}{2}\right)^2 = \frac{1}{2}(p^2 + \frac{1}{4}) .$$

or simplifying this result equality only holds if $p = 1/2$ which we are assuming is not true. Now let event E denote the event that we are *told* that the coin selected and flipped by both

parties is the fair one, i.e. that event C_2 happens. Then we have

$$\begin{aligned} P(H_1|E) &= \frac{1}{2} \quad \text{and} \quad P(H_2|E) = \frac{1}{2} \\ P(H_1, H_2|E) &= \frac{1}{4}. \end{aligned}$$

Since in this case we do have $P(H_1, H_2|E) = P(H_1|E)P(H_2|E)$ we have the required conditional independence. Intuitively, this result makes sense because once we *know* that the coin flipped is fair we expect the result of each flip to be independent.

Exercise C.2-9 (the Monte-Hall problem)

Once we have selected a curtain we are two possible situations we might find ourself in. The first situation A, is where we have selected the curtain with the prize behind it or situation B where we have *not* selected the curtain with the prize behind it. The initial probability of event A is $1/3$ and that of event B is $2/3$. We will calculate our probability of winning under the two choices of actions. The first is that we choose *not* to switch curtains after the emcee opens a curtain that does not cover the prize. The probability that we win, assuming the "no change" strategy, can be computed by conditioning on the events A and B above as

$$P(W) = P(W|A)P(A) + P(W|B)P(B).$$

Under event A and the fact that we don't switch curtains we have $P(W|A) = 1$ and $P(W|B) = 0$, so we see that $P(W) = 1/3$ as would be expected. If we now assume that we follow the second strategy where by we *switch* curtains after the emcee revels a curtain behind which the prize does not sit. In that case $P(W|A) = 0$ and $P(W|B) = 1$, since in this strategy we switch curtains. The the total probability we will is given by

$$P(W) = 0 + 1 \cdot \frac{2}{3} = \frac{2}{3},$$

since this is greater than the probability we would win under the strategy were we *don't switch* this is the action we should take.

Exercise C.2-10 (a prison delima)

I will argue that X has obtained some information from the guard. Before asking his question the probability of event X (X is set free) is $P(X) = 1/3$. If prisoner X is told that Y (or Z in fact) is to be executed, then to determine what this implies about the event X we need to compute $P(X|\neg Y)$. Where X, Y, and Z are the events that prisoner X, Y, or Z is to be set free respectively. Now from Bayes' rule

$$P(X|\neg Y) = \frac{P(\neg Y|X)P(X)}{P(\neg Y)}.$$

We have that $P(\neg Y)$ is given by

$$P(\neg Y) = P(\neg Y|X)P(X) + P(\neg Y|Y)P(Y) + P(\neg Y|Z)P(Z) = \frac{1}{3} + 0 + \frac{1}{3} = \frac{2}{3}\,.$$

So the above probability then becomes

$$P(X|\neg Y) = \frac{1(1/3)}{2/3} = \frac{1}{2} > \frac{1}{3}\,.$$

Thus the probability that prisoner X will be set free has increased and prisoner X has learned from his question.

proof of the cumulative summation formula for $E[X]$ (book notes page 1109)

We have from the definition of the expectation that

$$E[X] = \sum_{i=0}^{\infty} iP\{X = i\}\,.$$

expressing this in terms of the the complement of the cumulative distribution function $P\{X \geq i\}$ gives $E[X]$ equal to

$$\sum_{i=0}^{\infty} i(P\{X \geq i\} - P\{X \geq i+1\})\,.$$

Using the theory of difference equations we write the difference in probabilities above in terms of the Δ operator defined on a discrete function $f(i)$ as

$$\Delta g(i) \equiv g(i+1) - g(i)\,,$$

giving

$$E[X] = -\sum_{i=0}^{\infty} i\Delta_i P\{X \geq i\}\,.$$

No using the discrete version of integration by parts (demonstrated here for two discrete functions f and g) we have

$$\sum_{i=0}^{\infty} f(i)\Delta g(i) = f(i)g(i)|_{i=1}^{\infty} - \sum_{i=1}^{\infty} \Delta f(i)g(i)\,,$$

gives for $E[X]$ the following

$$\begin{aligned} E[X] &= -iP\{X \geq i\}|_{i=1}^{\infty} + \sum_{i=1}^{\infty} P\{X \geq i\} \\ &= \sum_{i=1}^{\infty} P\{X \geq i\}\,. \end{aligned}$$

	1	2	3	4	5	6
1	(2,1)	(3,2)	(4,3)	(5,4)	(6,5)	(7,6)
2	(2,2)	(4,2)	(5,3)	(6,4)	(7,5)	(8,6)
3	(4,3)	(5,3)	(6,3)	(7,4)	(8,5)	(9,6)
4	(5,4)	(6,4)	(7,4)	(8,4)	(9,5)	(10,6)
5	(6,5)	(7,5)	(8,5)	(9,5)	(10,5)	(11,6)
6	(7,6)	(8,6)	(9,6)	(10,6)	(11,6)	(12,6)

Table 7: The possible values for the sum (the first number) and the maximum (the second number) observed when two die are rolled.

which is the desired sum. To see this another way we can explicitly write out the summation given above and cancel terms as in

$$\begin{aligned} E[X] &= \sum_{i=0}^{\infty} i(P\{X \geq i\} - P\{X \geq i+1\}) \\ &= 0 + P\{X \geq 1\} - P\{X \geq 2\} \\ &+ 2P\{X \geq 2\} - 2P\{X \geq 3\} \\ &+ 3P\{X \geq 3\} - 3P\{X \geq 4\} + \ldots \\ &= P\{X \geq 1\} + P\{X \geq 2\} + P\{X \geq 3\} + \ldots , \end{aligned}$$

verifying what was claimed above.

Appendix C.3 (Discrete random variables)

Exercise C.3-1 (expectation of the sum and maximum of two die)

We will begin by computing all possible sums and maximum that can be obtained when we roll two die. These numbers are computed in table 7, where the row corresponds to the first die and the column corresponds to the second die. Since each roll pairing has a probability of $1/36$ of happening we see that the expectation of the sum S is given by

$$\begin{aligned} E[S] &= 2\left(\frac{1}{36}\right) + 3\left(\frac{2}{36}\right) + 4\left(\frac{3}{36}\right) + 5\left(\frac{4}{36}\right) + 6\left(\frac{5}{36}\right) + 7\left(\frac{6}{36}\right) \\ &+ 8\left(\frac{5}{36}\right) + 9\left(\frac{4}{36}\right) + 10\left(\frac{3}{36}\right) + 11\left(\frac{2}{36}\right) + 12\left(\frac{1}{36}\right) \\ &= 6.8056\,. \end{aligned}$$

While the expectation of the maximum M is given by a similar expression

$$\begin{aligned} E[M] &= 1\left(\frac{1}{36}\right) + 2\left(\frac{3}{36}\right) + 3\left(\frac{5}{36}\right) + 4\left(\frac{7}{36}\right) + 5\left(\frac{9}{36}\right) + 6\left(\frac{11}{36}\right) \\ &= 4.4722\,. \end{aligned}$$

Exercise C.3-2 (expectation of the index of the max and min of a random array)

Since the array elements are assumed random the maximum can be at any of the n indices with probability $1/n$. Thus if we define X to be the random variable representing the location of the maximum of our array we see that the expectation of X is given by

$$\begin{aligned} E[X] &= 1\left(\frac{1}{n}\right) + 2\left(\frac{1}{n}\right) + \cdots + n\left(\frac{1}{n}\right) \\ &= \frac{1}{n}\sum_{k=1}^{n} k = \frac{1}{n}\left(\frac{n(n+1)}{2}\right) \\ &= \frac{n+1}{2}\,. \end{aligned}$$

The same is true for the expected location of the minimum.

Exercise C.3-3 (expected cost of playing a carnival game)

Define X to be the random variable denoting the payback from one play of the carnival game. Then the payback depends on the possible outcomes after the player has guessed a number. Let E_0 be the event that the player's number does not appear on any die, E_1 the event that the player's number appears on only one of the three die, E_2 the event that the player's number appears on only two of the die, and finally E_3 the event that the player's number appears on all three of the die. The the expected payback is given by

$$E[X] = -P(E_0) + P(E_1) + 2P(E_2) + 3P(E_3)\,.$$

We now compute the probability of each of the events above. We have

$$\begin{aligned} P(E_0) &= \left(\frac{5}{6}\right)^3 = 0.5787 \\ P(E_1) &= 3\left(\frac{1}{6}\right)\left(\frac{5}{6}\right)^2 = 0.3472 \\ P(E_2) &= 3\frac{1}{6}\cdot\frac{1}{6}\cdot\frac{5}{6} = 0.0694 \\ P(E_3) &= \frac{1}{6^3} = 0.0046\,. \end{aligned}$$

Where the first equation expresses the fact that each individual die will not match the player's selected die with probability of $5/6$, so the three die will not match the given die with probability $(5/6)^3$. The other probabilities are similar. Using these to compute the expected payoff using the above formula gives

$$E[X] = -\frac{17}{216} = -0.0787\,.$$

To verify that these numbers are correct in the Matlab file `exercise_C_3_3.m`, a Monte-Carlo simulation is developed verifying these values. This function can be run with the command `exercise_C_3_3(100000)`.

Exercise C.3-4 (bounds on the expectation of a maximum)

We have for $E[\max(X,Y)]$ computed using the joint density of X and Y that

$$\begin{aligned} E[\max(X,Y)] &= \sum_x \sum_y \max(x,y) P\{X=x, Y=y\} \\ &\leq \sum_x \sum_y (x+y) P\{X=x, Y=y\}\,, \end{aligned}$$

since X and Y are non-negative. Then using the linearity of the above we have

$$\begin{aligned} E[\max(X,Y)] &\leq \sum_x \sum_y x P\{X=x, Y=y\} + \sum_x \sum_y y P\{X=x, Y=y\} \\ &= \sum_x x P\{X=x\} + \sum_y y P\{Y=y\} \\ &= E[X] + E[Y]\,, \end{aligned}$$

which is what we were to show.

Exercise C.3-5 (functions of independent variables are independent)

By the independence of X and Y we know that

$$P\{X=x, Y=y\} = P\{X=x\}P\{Y=y\}\,.$$

But by definition of the random variable X has a realization equal to x then the random variable $f(X)$ will have a realization equal to $f(x)$. The same statement hold for the random variable $g(Y)$ which will have a realization of $g(y)$. Thus the above expression is equal to (almost notationally)

$$P\{f(X)=f(x), g(Y)=g(y)\} = P\{f(X)=f(x)\}P\{g(Y)=g(y)\}$$

Defining the random variable F by $F=f(X)$ (an instance of this random variable f) and the random variable $G=g(Y)$ (similarly and instance of this random variable g) the above shows that

$$P\{F=f, G=g\} = P\{F=f\}P\{G=g\}$$

or

$$P\{f(X)=f, g(Y)=g\} = P\{f(X)=f\}P\{g(Y)=g\}\,,$$

showing that $f(X)$ and $g(Y)$ are independent as requested.

Exercise C.3-6 (Markov's inequality)

To prove that

$$P\{X \geq t\} \leq \frac{E[X]}{t}\,,$$

is equivalent to proving that

$$E[X] \geq tP\{X \geq t\}\,.$$

To prove this first consider the expression for $E[X]$ broken at t i.e.

$$\begin{aligned} E[X] &= \sum_x xP\{X = x\} \\ &= \sum_{x<t} xP\{X = x\} + \sum_{x\geq t} xP\{X = x\}\,. \end{aligned}$$

But since X is non-negative we can drop the expression $\sum_{x<t} xP\{X = x\}$ from the above and obtain a lower bound. Specifically we have

$$\begin{aligned} E[X] &\geq \sum_{x\geq t} xP\{X = x\} \\ &\geq t\sum_{x\geq t} P\{X = x\} \\ &= tP\{X \geq t\}\,, \end{aligned}$$

or the desired result.

Exercise C.3-7 (if $X(s) \geq X'(s)$ then $P\{X \geq t\} \geq P\{X' \geq t\}$)

Lets begin by defining two sets A_t and B_t as follows

$$\begin{aligned} A_t &= \{s \in S : X(s) \geq t\} \\ B_t &= \{s \in S : X'(s) \geq t\}\ . \end{aligned}$$

Then lets consider an element $\tilde{s} \in B_t$. Since $\tilde{s}$ is in B_t we know that $X'(\tilde{s}) > t$. From the assumption on X and X' we know that $X(\tilde{s}) \geq X'(\tilde{s}) \geq t$, and $\tilde{s}$ must also be in A_t. Thus the set B_t is a subset of the set A_t. We therefore have that

$$P\{X \geq t\} = \sum_{s\in A_t} P\{s\} \geq \sum_{s\in B_t} P\{s\} = P\{X' \geq t\}\,,$$

and the desired result is obtained.

Exercise C.3-8 ($E[X^2] > E[X]^2$)

From the calculation of the variance of the random variable X we have that

$$\mathrm{Var}(X) = E[(X - E[X])^2] > 0\,,$$

since the square in the expectation is always a positive number. Expanding this square we find that

$$E[X^2] - E[X]^2 > 0\,.$$

Which shows on solving for $E[X^2]$ that

$$E[X^2] > E[X]^2 \,,$$

or that the expectation of the square of the random variable is larger than the square of the expectation. This make sense because if X where to take on both positive and negative values in computing $E[X]^2$ the expectation would be decreased by X taking on instances of both signs. The expression $E[X^2]$ would have no such difficulty since X^2 is aways positive regardless of the sign of X.

Exercise C.3-9 (the variance for binary random variables)

Since X takes on only value in $\{0, 1\}$, lets assume that

$$\begin{aligned} P\{X = 0\} &= 1 - p \\ P\{X = 1\} &= p \,. \end{aligned}$$

Then the expectation of X is given by $E[X] = 0(1-p) + 1p = p$. Further the expectation of X^2 is given by $E[X^2] = 0(1-p) + 1p = p$. Thus the variance of X is given by

$$\begin{aligned} \mathrm{Var}(X) &= E[X^2] - E[X]^2 \\ &= p - p^2 = p(1-p) \\ &= E[X](1 - E[X]) \\ &= E[X]\, E[1 - X] \,. \end{aligned}$$

Where the transformation $1 - E[X] = E[1 - X]$ is possible because of the linearity of the expectation.

Exercise C.3-10 (the variance for aX)

From the suggested equation we have that

$$\begin{aligned} \mathrm{Var}(aX) &= E[(aX)^2] - E[aX]^2 \\ &= E[a^2X^2] - a^2E[X]^2 \\ &= a^2(E[X^2] - E[X]^2) \\ &= a^2\mathrm{Var}(X) \,, \end{aligned}$$

by using the linearity property of the expectation.

Appendix C.4 (The geometric and binomial distributions)

an example with the geometric distribution (book notes page 1112)

All the possible outcomes for the sum on two die are given in table 7. There one sees that the sum of a 7 occurs along the right facing diagonal (elements $(6,1), (5,5), (4,3), (3,4), (2,5), (1,6)$),

while the sum of 11 occurs for the elements $(6,5),(5,6)$. The the probability we roll a 7 or an 11 is given by

$$p = \frac{6}{36} + \frac{2}{36} = \frac{8}{36} = \frac{2}{9}\,.$$

if we take this as the probability of success then the remaining results from the book follow.

Exercise C.4-1 (the closure property for the geometric distribution)

The geometric distribution gives the probability our first success occurs at trail numbered k when each trial has p as a probability of success. It has a distribution function given by

$$P\{X = k\} = p(1-p)^{k-1} = pq^{k-1}\,.$$

Where we have defined $q = 1 - p$. If we sum this probability distribution for all possible location of the first success we obtain

$$\begin{aligned}
\sum_{k=1}^{\infty} p(1-p)^{k-1} &= \frac{p}{q}\sum_{k=1}^{\infty} q^k \\
&= \frac{p}{q}\left(\sum_{k=0}^{\infty} q^k - 1\right) \\
&= \frac{p}{q}\left(\frac{1}{1-q} - 1\right) \\
&= \frac{p}{q}\left(\frac{1}{p} - 1\right) \\
&= \frac{p}{q}\left(\frac{1-p}{p}\right) = 1\,,
\end{aligned}$$

as we were to show.

Exercise C.4-2 (average number of times to obtain three heads and tails)

Let one trial consist of flipping six coins. Then we consider a success when we have obtained three head and three tails. Thus since there are 2^6 possible outcomes of six flips when the flips are ordered there are then $\binom{6}{3}$ the probability of "success" (p) for this experiment is given by

$$p = \frac{\binom{6}{3}}{2^6} = \frac{20}{64} = 0.3125\,.$$

Since the number of trials needed to obtain a success is a geometric distribution we have that the average number of times we must perform this experiment is given by the average for a geometric distribution with probability of success p or

$$\frac{1}{p} = \frac{1}{0.3125} = 3.2\,.$$

Exercise C.4-3 (an equality with the binomial distribution)

Using the definition of $b(k;n,p)$ we have that

$$b(k;n,p) = \left(\begin{array}{c} n \\ k \end{array} \right) p^k (1-p)^{n-k} .$$

Since $\left(\begin{array}{c} n \\ k \end{array} \right) = \left(\begin{array}{c} n \\ n-k \end{array} \right)$ the above can be written as

$$b(k;n,p) = \left(\begin{array}{c} n \\ n-k \end{array} \right) (1-p)^{n-k} p^k .$$

Defining $q = 1-p$ we see the right hand side of the above is equal to

$$\left(\begin{array}{c} n \\ n-k \end{array} \right) q^{n-k} (1-q)^k = b(n-k;n,q) ,$$

and the desired result is shown.

Exercise C.4-4 (the maximum of the binomial coefficient)

From the discussion in the text the binomial coefficient has a maximum at the integer k that lies in the range $np - q < k < (n+1)p$. To evaluate the approximate maximum of the binomial coefficient we will evaluate it at $k = np$ which is certainly between the two limits $np-q$ and $(n+1)p$. Our binomial coefficient evaluated at its approximate maximum $k = np$ is given by

$$\begin{aligned} b(np;n,p) &= \left(\begin{array}{c} n \\ np \end{array} \right) p^{np} (1-p)^{n-np} \\ &= \frac{n!}{(n-np)!(np)!} p^{np} (1-p)^{nq} \\ &= \frac{n!}{(nq)!(np)!} p^{np} q^{nq} . \end{aligned}$$

where we have introduced $q = 1-p$ into the above to simplify the notation. Using Stirling's approximation for $n!$ given by

$$n! = \sqrt{2\pi n} \left(\frac{n}{e} \right)^n \left(1 + \theta(\frac{1}{n}) \right) ,$$

we can simplify the factorial expression above (and dropping the order symbols θ) we have

$$\begin{aligned} \frac{n!}{(nq)!(np)!} &\approx \sqrt{2\pi n} \left(\frac{n}{e} \right)^n \times \frac{1}{\sqrt{2\pi nq}} \left(\frac{e}{nq} \right)^{nq} \times \frac{1}{\sqrt{2\pi np}} \left(\frac{e}{np} \right)^{np} \\ &= \frac{1}{\sqrt{2\pi}} \left(\frac{n}{(nq)(np)} \right)^{1/2} \frac{n^n}{(nq)^{nq}(np)^{np}} \\ &= \frac{1}{\sqrt{2\pi}} \left(\frac{1}{nqp} \right)^{1/2} \left(\frac{1}{q^q p^p} \right)^n . \end{aligned}$$

upon multiplying the above by $p^{np}q^{nq}$ we find that

$$b(np;n,p) \approx \left(\frac{1}{2\pi npq}\right)^{1/2},$$

as we were to show.

Exercise C.4-5 (the limiting probability of no successes)

The probability of no successes in a sequence of n Bernoulli trials with (probability of success $p = 1/n$) is given by $b(0;n,1/n)$. This is equal to

$$\begin{aligned} b(0;n,1/n) &= \binom{n}{0}\left(\frac{1}{n}\right)^0\left(1-\frac{1}{n}\right)^n \\ &= \left(1-\frac{1}{n}\right)^n \end{aligned}$$

Remembering a famous limit from calculus

$$\lim_{x\to+\infty}\left(1+\frac{x}{n}\right)^n = e^x,$$

we see that for large n the probability $b(0;n,1/n)$ is approximately e^{-1}. The probability of at least one success is given by $b(1;n,1/n)$. This is equal to

$$\begin{aligned} b(1;n,1/n) &= \binom{n}{1}\left(\frac{1}{n}\right)^1\left(1-\frac{1}{n}\right)^{n-1} \\ &= n\left(\frac{1}{n}\right)\left(1-\frac{1}{n}\right)^{n-1} \\ &= \left(1-\frac{1}{n}\right)^{n-1} \\ &= \frac{1}{(1-\frac{1}{n})}\left(1-\frac{1}{n}\right)^n \\ &\to \left(1-\frac{1}{n}\right)^n \quad \text{as} \quad n\to+\infty. \end{aligned}$$

Using the limiting argument above we have this probability equal to e^{-1} as n goes to infinity.

Exercise C.4-6 (the probability of obtaining the same number of heads)

For a total of $2n$ flips we have $2^{2n} = 4^n$ possible sequences of heads and tails in $2n$ flips. Let the first n of these correspond to the flips of professor Rosencrantz and the remaining n corresponds to those of professor Guildenstern. Following the hint in the book, we will call a success for professor Rosencrantz when his flip lands *heads* and a success for professor

Guildenstern when his flip lands *tails*. Then both professors will obtain the *same* number of heads if say professor Rosencrantz has k successes while professor Guildenstern has $n-k$ successes. Thus the number of sequences (from our 4^n) that have the same number of heads for both professors is an equivalent problem to selecting $k+(n-k)=n$ total locations from among $2n$ possible and declaring these to be the locations of the successes for both professors. This means that if a success is placed before the $n+1$th flip it is considered to be a success for professor Rosencrantz and denotes a head (the remaining flips for Rosencrantz are then all tails). If a success falls after the nth flip it is considered to be a success for professor Guildenstern and is considered to be a tail (the remaining flips for Guildenstern are considered to be all heads). The number of sequences with n total successes is given by $\left(\begin{array}{c} 2n \\ n \end{array} \right)$ and so the probability of obtaining the same number of heads is given by

$$\frac{\left(\begin{array}{c} 2n \\ n \end{array} \right)}{4^n} ,$$

as claimed. Using the result from Exercise C.1-13 which derives the approximate

$$\left(\begin{array}{c} 2n \\ n \end{array} \right) = \frac{4^n}{\sqrt{\pi n}} \left(1 + O(\frac{1}{n}) \right) .$$

we can simplify the probability of obtaining the same number of heads and find that is is approximately equal to

$$\frac{1}{\sqrt{\pi n}} \left(1 + O(\frac{1}{n}) \right) .$$

Another way to approach this problem is to explicitly sum the probability that professor Rosencrantz has k successes while professor Guildenstern has $n-k$ successes. The probability that both these events happen (since they are independent) is given by the product of the appropriate binomial distribution i.e.

$$b(k;n,1/2)b(n-k;n,1/2) .$$

So the total probability of that the two professors get the same number of heads is given by the sum of that probability for all possible k's ($k=0$ to $k=n$) or

$$\sum_{k=0}^{n} b(k;n,1/2)b(n-k;n,1/2) .$$

This expression simplifies in the obvious ways

$$\begin{aligned} \sum_{k=0}^{n} b(k;n,1/2)b(n-k;n,1/2) &= \sum_{k=0}^{n} b(k;n,1/2)^2 \\ &= \sum_{k=0}^{n} \left(\begin{array}{c} n \\ k \end{array} \right)^2 \left(\frac{1}{2} \right)^{2n} \\ &= \frac{1}{4^n} \sum_{k=0}^{n} \left(\begin{array}{c} n \\ k \end{array} \right)^2 . \end{aligned}$$

In the above we have used the symmetry of the binomial distribution with respect to k and $n-k$, i.e. $b(k;n,1/2) = b(n-k;n,1/2)$, which a special case of the result shown in Exercise C.4-3 above. Equating these two results we have a nice combinatorial identity

$$\sum_{k=0}^{n} \left(\begin{array}{c} n \\ k \end{array} \right)^2 = \left(\begin{array}{c} 2n \\ n \end{array} \right) ,$$

as claimed.

Exercise C.4-7 (the entropy bound on the binomial distribution)

From the binomial bound derived in Appendix C.1 (also see the notes above that go with that section) we have that

$$\left(\begin{array}{c} n \\ k \end{array} \right) \leq 2^{nH(\frac{n}{k})} .$$

Using this expression we can immediately bound $b(k;n,1/2)$ as

$$\begin{aligned} b(k;n,1/2) &= \left(\begin{array}{c} n \\ k \end{array} \right) \left(\frac{1}{2}\right)^k \left(\frac{1}{2}\right)^{n-k} \\ &= \left(\begin{array}{c} n \\ k \end{array} \right) \left(\frac{1}{2}\right)^n \\ &\leq 2^{nH(\frac{n}{k})} \left(\frac{1}{2}\right)^n \\ &= 2^{nH(\frac{n}{k})-n} , \end{aligned}$$

which is the desired upper bound on $b(k;n,1/2)$.

Exercise C.4-8 (sums of Bernoulli random variables with different values of p)

Since X is a random variable representing the total number of success in n Bernoulli trials (with each trial having p_i as its probability of success) we can explicitly express X as the sum of n indicator random variables I_i as

$$X = \sum_{i=1}^{n} I_i .$$

Here where I_i is *one* if the ith Bernoulli trial is a success and zero otherwise. Then since the definition of $P\{X < k\}$ means the sum of the probability that X takes the values $0, 1, \ldots, k-1$ or

$$P\{X < k\} = \sum_{i=0}^{k-1} P\{X = i\} ,$$

the probability we are attempting to bound is given in terms of "equality" probabilities i.e. expressions like $P\{X = i\}$. Now for the random variable X to be *exactly* i means that only

i of the Bernoulli trials were a success (no more and no less) and the remaining trials were a failure. Thus if we select a subset of size i from the n total Bernoulli random variables, the above probability is expressed as the sum over all such subsets of size i i.e.

$$P\{X = i\} = \sum p_{l_1} p_{l_2} \cdots p_{l_i} (1 - p_{l_{i+1}})(1 - p_{l_{i+2}}) \cdots (1 - p_{l_n}) \,. \tag{5}$$

Here the sum is over all possible subsets of size i from n and the indices l_i select which of the i Bernoulli indicator variables from n were successful. Now since $p_i \leq p$ it follows that $1 - p_i \geq 1 - p$ and thus using these for each factor in the product we have that the term in the sum above is bounded as

$$p_{l_1} p_{l_2} \cdots p_{l_i} (1 - p_{l_{i+1}})(1 - p_{l_{i+2}}) \cdots (1 - p_{l_n}) \leq p^i (1 - p)^{n-i} \,, \tag{6}$$

Since there are $\left(\begin{array}{c} n \\ i \end{array} \right)$ total terms in the sum above we see that

$$P\{X = i\} \leq \left(\begin{array}{c} n \\ i \end{array} \right) p^i (1 - p)^{n-i} = b(i; n, p) \,.$$

Thus using this we have just shown that

$$P\{X < k\} = \sum_{i=0}^{k-1} P\{X = i\} \leq \sum_{i=0}^{k-1} b(i; n, p) \,,$$

which is the desired result.

Exercise C.4-9 (a lower bound on our success)

We can work this exercise in much the same way as the previous exercise C.4-8. Here we can still write $P\{X = i\}$ in the form as given by Equation 5. Rather than use a single value of p such that $p \geq p_i$ for all i we now have several p'_i where $p'_i \geq p_i$ and thus Equation 6 in this case becomes

$$p_{l_1} p_{l_2} \cdots p_{l_i} (1 - p_{l_{i+1}})(1 - p_{l_{i+2}}) \cdots (1 - p_{l_n}) \leq p'_{l_1} p'_{l_2} \cdots p'_{l_i} (1 - p'_{l_{i+1}})(1 - p'_{l_{i+2}}) \cdots (1 - p'_{l_n}) \,.$$

Thus

$$\begin{aligned} P\{X = i\} &= \sum p_{l_1} p_{l_2} \cdots p_{l_i} (1 - p_{l_{i+1}})(1 - p_{l_{i+2}}) \cdots (1 - p_{l_n}) \\ &\leq \sum p'_{l_1} p'_{l_2} \cdots p'_{l_i} (1 - p'_{l_{i+1}})(1 - p'_{l_{i+2}}) \cdots (1 - p'_{l_n}) = P\{X' = i\} \,. \end{aligned}$$

The cumulative probabilities now follow. We have

$$P\{X' \geq k\} = \sum_{i=k}^{n} P\{X' = i\} \geq \sum_{i=k}^{n} P\{X = i\} = P\{X \geq k\} \,,$$

which is the desired result.

Appendix C.5 (The tails of the binomial distribution)

Exercise C.5-1 (flip n tails or obtain fewer than n heads)

We know that the probability of obtaining n tails when flipping a fair coin is given by

$$P_{n \text{ tails in } n \text{ flips}} = \left(\frac{1}{2}\right)^n .$$

This is to be compared to the probability of flipping fewer than n heads when we flip our fair coin $4n$ times or

$$P_{\text{less than } n \text{ heads in } 4n \text{ flips}} = \sum_{i=0}^{n-1} b(i; 4n, 1/2) .$$

When I first saw this problem I thought it would have a definitive answer (i.e. the first expression is always less than the second expression). Unfortunately after working in for a bit I think the answer depends on the value of n. After making an attempt to solve it in various ways I decided to compute each of the above numerically and see which one is larger. In the `R` code `exercise_C_5_1.R` we compute the two probabilities above and denote which one is larger. I find

```
   n P_n_tails_in_n P_less_than_n_heads_in_4n first_gt_second
1 15   3.051758e-05              2.111852e-05            TRUE
2 16   1.525879e-05              1.218229e-05            TRUE
3 17   7.629395e-06              7.037078e-06            TRUE
4 18   3.814697e-06              4.070015e-06           FALSE
5 19   1.907349e-06              2.356621e-06           FALSE
6 20   9.536743e-07              1.365935e-06           FALSE
```

Based on the above I should attempt to prove

$$P_{n \text{ tails in } n \text{ flips}} > P_{\text{less than } n \text{ heads in } 4n \text{ flips}} ,$$

when $n \leq 17$ and

$$P_{n \text{ tails in } n \text{ flips}} < P_{\text{less than } n \text{ heads in } 4n \text{ flips}} ,$$

when $n \geq 18$. This means that we need to show

$$\left(\frac{1}{2}\right)^n < \sum_{i=0}^{n-1} b(i; 4n, 1/2) .$$

for $n \geq 18$. While this seemed simple enough I was unable to find a way to prove this (even for large n). If anyone sees a way to do this please contact me.

Exercise C.5-2 (bounds on the right tail of the binomial distribution)

We begin by proving Corollary C.6 which claims that given a sequence of n Bernoulli trials each with probability p of success that

$$P\{X > k\} = \sum_{i=k+1}^{n} b(i; n, p) < \frac{(n-k)p}{k-np} b(k; n, p) \,.$$

To prove this lets first consider the ratio $b(i+1; n, p)/b(i; n, p)$ which is given by

$$\begin{aligned} \frac{b(i+1; n, p)}{b(i; n, p)} &= \frac{\binom{n}{i+1} p^{i+1}(1-p)^{n-i-1}}{\binom{n}{i} p^i (1-p)^{n-i}} \\ &= \left(\frac{n-i}{i+1}\right) \frac{p}{1-p} \,. \end{aligned}$$

Thus (since we assume that i is taken between k and n i.e. that $k < i < n$) we have

$$\frac{b(i+1; n, p)}{b(i; n, p)} \leq \left(\frac{n-k}{k+1}\right) \left(\frac{p}{1-p}\right) \,.$$

Defining x to be the above expression we see that $x < 1$ through the following arguments

$$\begin{aligned} x &< \left(\frac{n-np}{np+1}\right) \left(\frac{p}{1-p}\right) \\ &= \frac{n(1-p)p}{(np+1)(1-p)} = \frac{np}{np+1} \\ &< \frac{np}{np+1} < \frac{np}{np} = 1 \,. \end{aligned}$$

Now using the ratio above and the definition of x we have that $b(i+1; n, p)$ is bounded by $b(i; n, p)$ as

$$b(i+1; n, p) < x b(i; n, p) \,.$$

In the same way, $b(i+2; n, p)$ is bounded by $b(i; n, p)$ as

$$b(i+2; n, p) < x b(i+1; n, p) < x^2 b(i; n, p) \,.$$

Continuing this iteration scheme for $i+3, i+4, \cdots$ we see that the probability $X > k$ (the right tail probability) is bounded above by

$$\sum_{i=k+1}^{n} b(i; n, p) < x b(k; n, p) + x^2 b(k; n, p) + \cdots + x^n b(k; n, p) \,.$$

This can be further manipulated (by summing to infinity) as follows

$$\begin{aligned} \sum_{i=k+1}^{n} b(i; n, p) &< b(k; n, p) \sum_{i=1}^{n} x^i \\ &< b(k; n, p) \sum_{i=1}^{\infty} x^i \\ &= b(k; n, p) \left(\frac{x}{1-x}\right) \,. \end{aligned}$$

Remembering the definition of x we can compute that

$$\frac{x}{1-x} = \frac{(n-k)p}{1-p+k-np}\,,$$

which is itself less than $\frac{(n-k)p}{k-np}$ so we can finally conclude that

$$\sum_{i=k+1}^{\infty} b(i;n,p) \leq \frac{(n-k)p}{k-np} b(k;n,p)\,,$$

as we were asked to show.

We next prove Corollary C.7 which is the statement that given a sequence of n Bernoulli trials each with probability p of being a success and a specific number of success k in the right tail of our distribution ($\frac{np+n}{2} < k < n$) that the probability of obtaining more than k success is less than one half the probability of obtaining more than $k-1$ success. This is really a statement that as we require having more and more successes from our n Bernoulli trials the probabilities decrease geometrically (with rate 1/2). We will begin by showing that the coefficient of $b(k;n,p)$ in Corollary C.6 is less than one. We have since $\frac{np+n}{2} < k < n$ that

$$\left(\frac{n-k}{k-np}\right)p < \left(\frac{n-\frac{np+n}{2}}{\frac{np+n}{2}-np}\right)p = p < 1\,.$$

Thus from Corollary C.6 we have that

$$P\{X > k\} = \sum_{i=k+1}^{n} b(i;n,p) < b(k;n,p)\,,$$

or equivalently that

$$\frac{b(k;n,p)}{\sum_{i=k+1}^{n} b(i;n,p)} > 1\,.$$

Now consider the ratio of $P\{X > k\}$ to $P\{X > k-1\}$

$$\begin{aligned}
\frac{P\{X > k\}}{P\{X > k-1\}} &= \frac{\sum_{i=k+1}^{n} b(i;n,p)}{\sum_{i=k}^{n} b(i;n,p)} \\
&= \frac{\sum_{i=k+1}^{n} b(i;n,p)}{b(k;n,p) + \sum_{i=k+1}^{n} b(i;n,p)} \\
&= \frac{1}{1 + \frac{b(k;n,p)}{\sum_{i=k+1}^{n} b(i;n,p)}} \\
&< \frac{1}{1+1} = \frac{1}{2}\,,
\end{aligned}$$

and the desired result is proven.

Exercise C.5-3 (upper bounds on a geometric sum)

Let $a > 0$ be given, and define $p = \frac{a}{a+1} < 1$, then

$$q = 1 - p = 1 - \frac{a}{a+1} = \frac{1}{a+1}\,.$$

Now consider the pq product found in the binomial coefficient $b(i;n,p) = \left(\begin{array}{c} n \\ i \end{array} \right) p^i q^{n-i}$, i.e. $p^i q^{n-i}$. With the definition given above for p and q we have

$$p^i q^{n-i} = \left(\frac{a}{a+1}\right)^i \left(\frac{1}{a+1}\right)^{n-i} = \frac{a^i}{(a+1)^n} .$$

From this we see that $a^i = (a+1)^n p^i q^{n-i}$ and thus our desired sum is given by

$$\sum_{i=0}^{k-1} \left(\begin{array}{c} n \\ i \end{array} \right) a^i = (a+1)^n \sum_{i=0}^{k-1} \left(\begin{array}{c} n \\ i \end{array} \right) p^i q^{n-i} .$$

Which can be seen as the left tail of the binomial distribution for which we have developed bounds for in the text. Specifically if X is a binomial random variable (with parameters (n,p)) then using the bounds that

$$P\{X < k\} = \sum_{i=0}^{k-1} b(i;n,p) < \frac{kq}{np-k} b(k;n,p) ,$$

the above becomes

$$\begin{aligned} \sum_{i=0}^{k-1} b(i;n,\frac{a}{a+1}) &\leq \frac{k\left(\frac{1}{a+1}\right)}{n\left(\frac{a}{a+1}\right)-k} b(k;n,\frac{a}{a+1}) \\ &= \frac{k}{na-k(a+1)} b(k;n,\frac{a}{a+1}) . \end{aligned}$$

Which gives in summary (including the factor $(a+1)^n$) then that

$$\sum_{i=0}^{k-1} \left(\begin{array}{c} n \\ i \end{array} \right) a^i \leq (a+1)^n \frac{k}{na-k(a+1)} b(k;n,\frac{a}{a+1}) ,$$

as we were requested to show.

Exercise C.5-4 (an upper bound on a geometric like sum)

Consider the sum we are asked to study

$$\sum_{i=0}^{k-1} p^i q^{n-i} .$$

Since $1 \leq \left(\begin{array}{c} n \\ i \end{array} \right)$ for $i = 0, 1, 2, \cdots, n$ the above sum is less than or equal to

$$\sum_{i=0}^{k-1} \left(\begin{array}{c} n \\ i \end{array} \right) p^i q^{n-i} = \sum_{i=0}^{k-1} b(i;n,p) ,$$

which by Theorem C.4 is bounded above by

$$\left(\frac{kq}{np-k}\right) b(k;n,p)\,.$$

From Lemma C.1 we can further bound the individual binomial coefficient $b(k;n,p)$ as

$$b(k;n,p) \leq \left(\frac{np}{k}\right)^k \left(\frac{nq}{n-k}\right)^{n-k}\,.$$

Thus incorporating each of these bounds we have that

$$\sum_{i=0}^{k-1} p^i q^{n-i} < \left(\frac{kq}{np-k}\right)\left(\frac{np}{k}\right)^k \left(\frac{nq}{n-k}\right)^{n-k}\,,$$

as we were requested to show.

Exercise C.5-5 (bounds on the number of failures)

We begin by defining the random variable Y to be the number of *failures* in n Bernoulli trials where the probability of success for each trial is p_i. Then in terms of X (the number of successes in these n Bernoulli trials) we have that $Y = n - X$, since each success is *not* a failure and vice versa. This expression simply states that if you know the number of success (failures) in a sequence of n Bernoulli trials it is easy to calculate the number of failures (successes). Taking the expectation of this expression and remembering that we defined μ as $\mu = E[X]$, we see that $E[Y] = n - E[X] = n - \mu$. Now applying Theorem C.8 to the random variable Y we have that

$$P\{Y - E[Y] \geq r\} \leq \left(\frac{E[Y]e}{r}\right)^r\,.$$

Writing the above expression in terms of X and μ we have

$$P\{(n-X) - (n-\mu) \geq r\} \leq \left(\frac{(n-\mu)e}{r}\right)^r\,.$$

or

$$P\{\mu - X \geq r\} \leq \left(\frac{(n-\mu)e}{r}\right)^r\,,$$

the desired result.

To show the second identity, we will apply Corollary C.9 to the random variable Y, but in this case the probability of success for each Bernoulli trial is a constant p. Thus the probability of failure in each Bernoulli trial is also a constant $q = 1 - p$. We thus obtain (since $E[Y] = nq$)

$$P\{Y - nq \geq r\} \leq \left(\frac{nqe}{r}\right)^r\,.$$

Again using the fact that $Y = n - X$ to write the above expression in terms of X we find

$$P\{n - X - nq \geq r\} \leq \left(\frac{nqe}{r}\right)^r ,$$

or

$$P\{n(1-q) - X \geq r\} \leq \left(\frac{nqe}{r}\right)^r ,$$

or

$$P\{np - X \geq r\} \leq \left(\frac{nqe}{r}\right)^r ,$$

the desired result.

Exercise C.5-6 (an exponential bound on the right tail)

Note: I was not able to prove the inequality

$$p_i e^{\alpha q_i} + q_i e^{-\alpha p_i} \leq e^{\alpha^2/2} ,$$

as suggested in the text. If anyone has such a proof please contact me.

Using the notation in the text we have that (by using Markov's inequality)

$$P\{X - \mu \geq r\} \leq E[e^{\alpha(X-\mu)}]e^{-\alpha r} .$$

Since X is the number of success in n independent Bernoulli trials $X = \sum_{i=1}^n X_i$ the expectation of the exponential can be written as the product of the individual indicator random variables X_i as

$$E[e^{\alpha(X-\mu)}] = \prod_{i=1}^n E[e^{\alpha(X_i - p_i)}] .$$

The explicit expectation of each term in the product is easily computed using the definition of expectation and gives

$$E[e^{\alpha(X_i - p_i)}] = p_i e^{\alpha q_i} + q_i e^{-\alpha p_i} .$$

If we now assume that $p_i e^{\alpha q_i} + q_i e^{-\alpha p_i} \leq e^{\alpha^2/2}$, then the product expectation can be written as

$$E[e^{\alpha(X-\mu)}] \leq \prod_{i=1}^n e^{\alpha^2/2} = e^{n\alpha^2/2} .$$

so that we can bound $P\{X - \mu \geq r\}$ as

$$P\{X - \mu \geq r\} \leq e^{n\alpha^2/2} e^{-\alpha r} = e^{-(\alpha r - n\alpha^2/2)} .$$

To compute the tightest possible upper bound on this probability we can minimize the right hand side of this expression with respect to α. This is equivalent to maximizing the expression $\alpha r - n\alpha^2/2$ with respect to α. Taking the α derivative of this expression and setting it equal to zero gives

$$r - n\alpha = 0 ,$$

which has $\alpha = r/n$ as its solution. The second derivative of $\alpha r - n\alpha^2/2$ with respect to α begin negative imply that this value of α corresponds to a maximum. The value of this quadratic at this maximal α is given by

$$\frac{r}{n}r - \frac{n}{2}\frac{r^2}{n^2} = \frac{r^2}{2n}\,.$$

This result intern implies that we have an upper bound on $P\{X - \mu \geq r\}$ given by

$$P\{X - \mu \geq r\} \leq e^{-\frac{r^2}{2n}}\,,$$

as we were required to show.

Exercise C.5-7 (minimizing the upper bound in Markov's inequality)

The equation C.45 is given by

$$P\{X - \mu \geq r\} \leq \exp(\mu e^\alpha - \alpha r)\,.$$

If we consider the right-hand side of this expression to be an function of α, we can find the α that minimizes this function by taking the first derivative and setting it equal to zero. We find that the first derivative is given by

$$\frac{d}{d\alpha}\exp(\mu e^\alpha - \alpha r) = \exp(\mu e^\alpha - \alpha r)\,(\mu e^\alpha - r)\,.$$

Which when we set this equal to zero and solve for α gives

$$\alpha = \ln(\frac{r}{\mu})\,.$$

We can check that this point is indeed a minimum of our right-hand side by computing the sign of the second derivative at that point. The second derivative is given by

$$\begin{aligned}\frac{d^2}{d\alpha^2}\exp(\mu e^\alpha - \alpha r) &= \exp(\mu e^\alpha - \alpha r)(\mu e^\alpha - r)^2 + \exp(\mu e^\alpha - \alpha r)(\mu e^\alpha) \\ &= \exp(\mu e^\alpha - \alpha r)(\mu^2 e^{2\alpha} - 2\mu r e^\alpha + r^2 + \mu e^\alpha) \\ &= \exp(\mu e^\alpha - \alpha r)(\mu^2 e^{2\alpha} - (2r-1)\mu e^\alpha + r^2)\,.\end{aligned}$$

Evaluating this expression at $\alpha = \ln(r/\mu)$ (and ignoring the exponential factor which does not affect the sign of the second derivative) gives

$$\mu^2\left(\frac{r^2}{\mu^2}\right) - (2r-1)r + r^2 = r^2 - 2r^2 + r + r^2 = r > 0\,.$$

Proving that for $\alpha = \ln(r/\mu)$ the expression $\exp(\mu e^\alpha - \alpha r)$ is a minimum.

Problem C-1 (Balls and bins)

Part (a): We have b choices for the location of the first ball, b choices for the location of the second ball, b choices for the location of the third ball and so on down to b choices for the n-th ball. Since the balls are distinguishable each of these placements represent different configurations. Thus the total number of possible configurations in this case is b^n.

Part (b): Following the hint, since we have a total of n distinguishable balls and b bins we can imagine the requested problem as equivalent to that of counting the number of configurations of n balls and $b-1$ sticks. The $b-1$ sticks represents the internal bin edges and the linear ordering of the balls between sticks represents the ball ordering within the bin. Now to count this number we note that we can order $n+b-1$ distinguishable objects in $(n+b-1)!$ ways, but since the $b-1$ sticks are indistinguishable we need to divide by the number of unique orderings of $b-1$ objects giving a total count of

$$\frac{(n+b-1)!}{(b-1)!}\,.$$

Part (c): If the balls in each bin are in fact identical in the same manner in which we needed to divide $(n+b-1)!$ by the number of unique orderings of distinguishable internal sticks $(b-1)!$ we need to divide the result above by the number of unique orderings of these n balls or $n!$. This gives

$$\frac{(n+b-1)!}{(b-1)!n!} = \left(\begin{array}{c} n+b-1 \\ n \end{array} \right) .$$

Part (d): We assume in this part that we have more bins than balls or $b > n$. If we assume for the moment that the balls are *not* identical then we have b choices for the location of the first ball, $b-1$ choices for the location of the second ball, $b-2$ choices for the location of the third ball and so on down to $b-n+1$ choices for the n-th ball. This gives

$$b(b-1)(b-2)\cdots(b-n+2)(b-n+1) = \frac{b!}{(b-n)!}\,,$$

Ways to place n unique balls. Since the balls are in fact identical this number over counts the total possible by $n!$. So we have

$$\frac{b!}{(b-n)!n!} = \left(\begin{array}{c} b \\ n \end{array} \right) ,$$

placements of identical balls where no more than one ball can go in any given bin.

Part (e): We assume in this part that we have more balls than bins or $n > b$. Consider all of our n indistinguishable balls lined up in a row and note that our problem is equivalent to that of counting the number of ways we can select $b-1$ "spaces" (which will represent the $b-1$ internal bin boundaries) from all $n-1$ possible spaces in our linear ordering of the balls. The number of ways to select $b-1$ things from a set of $n-1$ is given by

$$\left(\begin{array}{c} n-1 \\ b-1 \end{array} \right) ,$$

or the claimed number.

References

[1] W. G. Kelley and A. C. Peterson. *Difference Equations. An Introduction with Applications*. Academic Press, New York, 1991.