NYU Tandon Bridge

Homework 7

Kamel Bassel M Gazzaz

02/26/2021

Question 3

a) Exercise 8.2.2 section b.

The goal is show that $n^3 + 3n^2 + 4 = \Theta(n^3)$

Step 1: Determine big O

Using the exaggerate and simplify method:

1. Exaggerate $3n^2$ to $3n^3$

$n \geq 1$	$for\ all\ n \geq 1$
$n^2 \geq n$	Multiply both sides by n. Since n is positive it doesn't affect the order of the inequality
$n^3 \geq n^2$	Multiply both sides by n. Since n is positive it doesn't affect the order of the inequality
$3n^3 \geq 3n^2$	Multiply both sides by 3

2. Exaggerate 4 to $4n^3$

$n \geq 1$	$for\ all\ n \geq 1$
$4n \geq 4$	Multiply both sides by 4. Since n is positive it doesn't affect the order of the inequality
$64n^3 \geq 64$	Cube both sides. Since n is positive and cube is an increasing function it doesn't affect the order of the inequality
$4n^3 \geq 4$	Divide both sides by 16. Since n is positive it doesn't affect the order of the inequality.

3. n^3 stays the same

$n \geq n$	For all n
$n^2 \geq n^2$	Square both sides
$n^3 \geq n^3$	Multiply both sides by n. Since n is positive it doesn't affect the order of the inequality

Combining all three inequalities we get:

$n^3 + 3n^3 + 4n^3 \geq n^3 + 3n^2 + 4$	$for\ all\ n \geq 1$
$8n^3 \geq n^3 + 3n^2 + 4$	Sum everything up
$8 \cdot (n^3) \geq f(n)$	Replace the value of f(n) by its function name
$8\ g(n) \geq f(n)$	Replace n^3 with $g(n)$
$C_2 = 8$	By definition, we know $f(n) = O(g(n))$ if $c_2 \cdot g(n) \geq f(n)\ for\ all\ n \geq n_0$

Therefore, $f(n) = O(n^3)$ $for\ c_2 = 8\ and\ n_0 = 1$

Step 2: Determine Omega

Let's find the value of n for which the lower order terms $3n^2 + 4 \geq 0$:

$n \geq 0$	$for\ all\ n \geq 0$.
$n^2 \geq 0$	Square both sides (note, n^2 would be nonnegative even for $n < 0$ but that scope does not concern us).
$3n^2 \geq 0$	Divide both sides by 3. Since n^2 is positive it doesn't affect the order of the inequality.
$3n^2 + 4 \geq 4$	Add 4 to both sides.
$3n^2 + 4 \geq 0$	Since $3n^2 + 4 \geq 4$ and $4 \geq 0$, we can safely say $3n^2 + 4 \geq 0$.

As such, we showed that $3n^2 + 4 \geq 0$ for all $n \geq 0$. We also have that:

$n \geq n$	*for all n*
$n^2 \geq n^2$	Square both sides
$n^3 \geq n^3$	Multiply both sides by n. Since n is positive it doesn't affect the inequality

Combining those two inequalities we get:

$n^3 + 3n^2 + 4 \geq n^3$	*for all* $n \geq 0$
$f(n) \geq 1 \cdot (n^3)$	Replace the value of f(n) by its function name
$f(n) \geq 1 \cdot g(n)$	Replace n^3 with $g(n)$
$C_1 = 1$	By definition, we know that $f(n) = \Omega(g(n))$ if $c_1 \cdot g(n) \leq f(n)$ *for all* $n \geq n_0$

Therefore, $f(n) = \Omega(n^3)$ *for* $c_1 = 1$ *and* $n_0 = 0$

Step 3: Determine theta

We know that:

- f(n) is lower bound by $\Omega(n^3)$ with $c_1 = 1$ and $n_0 = 0$
- f(n) is upper bound by $O(n^3)$ with $c_2 = 8$ and $n_0 = 1$

By definition, $f(n) = \Theta(n^3)$ if there are positive constants c_1 *and* c_2 *and* n_0 such that

$c_1 (n^3) \leq f(n) \leq c_2(n^3)$ for all $n \geq n_0$.

If we use:

- $c_1 = 1$
- $c_2 = 8$
- $n_0 = 1$

Combining our results we get $n^3 \leq f(n) \leq 8n^3$ for all $n \geq 1$

Therefore, $n^3 + 3n^2 + 4 = \Theta(n^3)$

b) Exercise 8.3.5

a.

Step 1: Experiment with some values:

Consider the case:

- n = 6
- P = 0
    - $a_1 = -10$
    - $a_2 = 4$
    - $a_3 = 7$
    - $a_4 = -2$
    - $a_5 = 9$
    - $a_6 = -1$

I - First iteration of the outer while Loop:

1. First iteration of the first inner while loop:
   We have: i = 1 j = 6 && $a_i < P$
   So i++

2. Second (potential) iteration of the first inner while loop:
   We have: i = 1 j = 6
   But $a_i > P$
   So break out of first while

a. First iteration of the second inner while loop:
   We have i = 2 j = 6
   But $a_j < P$
   Break out of second while

   Swap a_2 and a_6 , $a_2 = -1$ and $a_6 = 4$

II - Second iteration of the outer while loop:

1. First iteration of the first inner while loop:
   We have i = 2 and j = 6 and $a_i < P$
   So i++

2. Second (potential) iteration of the first inner while loop:
   We have i = 3 j = 6:
   But $a_i > p$
   Break out of first while

a. First iteration of the second inner inner while loop

   We have i = 3 j = 6 and $a_j \geq P$
   So j--

b. Second iteration of the second inner inner while loop

   We have i = 3 and j = 5 and $a_j \geq P$
   So j --

c. Third (potential) iteration of the second inner while loop
   We have I = 3 j = 4:
   But $a_j < P$
   Break out of second while

   Swap a_3 and a_4. Now $a_3 = -2$ and $a_4 = 7$

III - Third iteration of the outer while loop

1. First iteration of the first inner while loop:
   We have i = 3 j = 4 and $a_i < P$
   So i++

2. Second (potential) iteration of the first inner while loop:
   We have i = 4 j = 4 and $a_i > p$
   Break out of first while

   Since i = j, algorithm doesn't enter second while loop

   Since i = j, don't swap a_i and a_j

   End of algorithm.

   We end up with $a_1 = -10$, $a_2 = -1$, $a_3 = -2$, $a_4 = 7$, $a_5 = 9$, $a_6 = 4$

Step 2: Algorithm Description in English

The mystery algorithm takes a sequence of numbers a_1 through a_n and swaps elements in the sequence such that we end up with:

- All elements that are less than a defined number P are to the left (beginning of the sequence).
- All elements whose value is greater than or equal to P are to the right (end of sequence).

The position that separates the numbers lower than P and higher than P depends on how many numbers are less than P. The algorithm also does not sort elements that are on one side or the other of P.

b.

This question can be interpreted in 2 ways:

1) Does the number of times each line (i := i + 1) and (j := j - 1) is executed (independently of the other) depend on the actual values of the numbers in the sequence or just the length of the sequence?
2) Does the number of times the line (i := i + 1) and (j := j - 1) are executed (in total) depend on the actual values of the numbers in the sequence or just the length of the sequence?

Case 1: Answer if we're considering each line independently:

The number of executions depends on both the length of the sequence n and the values of the numbers a_i through a_n.

1. We know that the first while loop is entered if $i < j$ and $a_i < P$. Therefore, to maximize the number of times "i: i+= 1" runs, we can:
    - Increase the length of the sequence n (since j takes the value of n).This will make the total number of iterations of i to reach j larger.
    - Make all values a_1 through a_n less than P. So long as the number a_k is less than P, then the first while loop will keep iterating until i reaches j. It takes n-1 iterations to do so.

2. Conversely, to minimize the number of times "i: i+= 1" runs, we can:
    - Decrease the length of n (since j takes on the value of n). This will make the total number of iterations of i to reach j smaller.
    - Make all values a_1 through a_n greater than or equal to P. If a_k is never less than P, then the first while loop is never entered and therefore iterates 0 times.

3. We know that the second while loop is entered if i < j and $a_j \geq P$. Therefore, to maximize the number of times "j: j+= 1" runs:
    - Increase the length of the sequence n. Since j takes on the value of n and iterates through the sequence backwards, j will start at a higher number and have more potential items to loop through.
    - Make all values a_1 through $a_n \geq P$. So long as the number a_j is greater than or equal to P, then the second while loop will keep iterating and decrementing j until j reaches i. It takes n-1 iterations to do so.

4. Conversely, to minimize the number of times "j: j+= 1" is executed:
    - Decrease the length of n. Since j takes on the value of n and iterates through the sequence backwards, j will start at a lower number and have less potential items to loop through.
    - Make all values a_1 through $a_n < P$. If all the numbers a_j are less than P, then the second while loop is never entered and therefore iterates 0 times.

<u>Case 2: Answer if we're considering both lines together:</u>

If we are considering how many times both lines run together, then the answer does not depend as much on the actual values of a_i through a_n since what maximizes the iterations of i++ will minimize j-- and vice versa. The sum of the lines i++ and j-- will always run n-1 times.

1. Example: Already sorted around the pivot P

Consider P = 0 and the sequence: -3 -2 -1 -1 7 7 7 7

First iteration of outer while loop :

i++ will execute 4 times then break out.

- -3 **-i++->** -2 -1 -1 7 7 7 7
- -3 -2 **-i++->** -1 -1 7 7 7 7
- -3 -2 -1 **-i++->** -1 7 7 7 7
- -3 -2 -1 -1 **-i++->** 7 7 7 7

j-- will execute 3 times then break out:

- -3 -2 -1 -1 7 7 7 <-j-- 7
- -3 -2 -1 -1 7 7 <- j-- 7 7
- -3 -2 -1 -1 7 <- j-- 7 7 7

i = j, so end of program

In total, i++ and j-- ran n-1 = 7 times.

2. Example: Unsorted around P

Consider our first case with P = 0 and a_1 = -10 a_2 = 4 a_3 = 7 a_4= -2 a_5 = 9 a_6 = -1

First iteration of outer while:

i++ will execute 1 time:

- -10 -i++-> 4 7 -2 9 -1

j-- will execute 0 times:

- -10 4 7 -2 9 -1

Swap values a_2 and a_6.

Second iteration of outer while:

i++ will execute 1 time:

- -10 -1 -i++-> 7 -2 9 4

j-- will execute 2 times:

- -10 4 7 -2 <-j-- 9 <-j-- 4

Swap values a_3 and a_4.

Third iteration of outer while:

i++ will execute 1 time:

- -10 -1 -2 i++-> 7 9 4

j-- will execute 0 times:

- -10 4 -2 7 9 4

Swap values a_3 and a_4.

i = j so end of program.

In total, i++ and j-- ran n-1 = 5 times.

3. Example: Oppositely sorted around P

Consider P = 0 and the sequence: 7 6 5 -4 -3 -2

First iteration of outer while loop :

i++ will execute 0 times.

- 7 6 5 -4 -3 -2

j-- will execute 0 times.

- 7 6 5 -4 -3 -2

Swap a_1 and a_6.

Second iteration of outer while loop :

i++ will execute 1 time.

- -2 **--i++->** 6 5 -4 -3 7

j-- will execute 1 time.

- -2 6 5 -4 -3 **<-j--** 7

Swap a_2 and a_5.

Third iteration of outer while loop :

i++ will execute 1 time.

- -2 -3 **--i++->** 5 -4 6 7

j-- will execute 1 time.

- -2 6 5 -4 **<-j--** 6 7

Swap a_3 and a_4.

Fourth iteration of outer while loop :

i++ will execute 1 time.

- -2 -3 -4 **--i++->** 5 6 7

j-- will execute 0 times.

- -2 6 -4 5 6 7

i = j, so end of program

In total, i++ and j-- ran n-1 = 5 times.

One edge case that might be an issue is whether varying the number of negatives and positives around P might make a difference. Let's show that it doesn't.

4. Example: Uneven count of numbers around P

Consider P = 0 and the sequence: -1 5 -4 -3 -2

First iteration of outer while loop :

i++ will execute 1 time.

- -1 **i++->** 5 -4 -3 -2

j-- will execute 0 times.

- -1 5 -4 -3 -2

Swap a_2 and a_5.

Second iteration of outer while loop :

i++ will execute 2 times.

- -1 -2 **i++->** -4 -3 5
- -1 -2 **i++->** -4 **i++->** -3 5

j-- will execute 1 time.

- -1 -2 -4 -3 **<-j--** 5

i = j, so end of program

In total, i++ and j-- ran n-1 = 4 times.

The last edge case that might be an issue is if all values were less/greater than P, that might make a difference. Let's show that it doesn't.

5. Example: All values to one side of P

Consider P = 0 and the sequence: 6 5 4 3 2

First iteration of outer while loop :

i++ will execute 0 times.

- 6 5 4 3 2

j-- will execute 4 times.

- 6 5 4 3 **<-j--** 2

- 6 5 4 <-j-- 3 2
- 6 5 <-j-- 4 3 2
- 6 <-j-- 5 4 3 2

i = j, so end of program

In total, i++ and j-- ran n-1 = 4 times.

In all cases, when we consider the total number of executions of the two statements together, the sum of the i++ and j-- statements are executed n-1 times regardless of the values in the sequence.

c.

The swap operation is executed every time i < j.

Since i increments and j decrements depending on the values of the numbers in the sequence, i and j may converge at different rates depending on the values of a_k in the sequence.

So, the number of times swap is executed also depends on the values of the numbers a_k in the sequence, not just the length of the sequence.

To maximize the number of times swap runs, we can:

- (Increase the length of the sequence n)
- Make the first half the values a_i through $a_n \geq P$ and the second half of them a < P. In this case, i will increment only once at a time and j will decrement only once at a time so there can be up to n/2 swaps. This is because each swap affects 2 numbers and once a number is swapped once it will no longer be swapped.

To minimize the number of times swap runs, we can:

- (Decrease the length of n)
- Option 1: Make all values a_1 through a_n greater than or equal to P. In this case, i will not increment and j will decrement all the way until it is equal to i and as a result the swap operation will never run.
- Option 2: Make all values a_1 through a_n less than P. In this case, i will increment all the way until it is equal to j and as a result the swap operation will never run.
- Option 3: Input a sequence that already has all numbers that are less than P in the beginning and the numbers greater than or equal to P at the end. In this case i will increment until the first number that is greater than P and and j will decrement until that same number. Since i and j meet at that position, the swap operation will never run.

d.

Previous findings:

- In part b, we determined that the number of times that, individually, i++ (or similarly j--) can run a minimum of 0 and maximum of n-1 times.
- In part c, we determined that the number of times swap runs varies depending on the values of the numbers in the sequence and can run 0 to n/2 times.
- Therefore, both the number of increments of i and the number swap operations are a function of the values in the sequence. So let's find a relationship between the number of times i increments and the number of swap operations.

Relationship:

- When i runs 0 times or n-1 times, the swap operation never executes.
- When i runs 1 time or n-2 times, the swap operation is executed 1 time.
- When i runs 2 times or n-3 times, the swap operation is executed 2 times.
- …
- When i runs n/2 times, the swap operation is executed n/2 times.

Therefore, there is a linear relationship between the number of times i++ runs and the number of swap operations that are executed (positive linear until n/2 then negative linear).

In part b, we also determined that the combination of i increments and j decrements is always n-1. Since we are doing worst case analysis, we must consider the "worst case" in which the most number of operations run. For the mystery algorithm, this occurs when i executes n/2 times (j also executes n/2 - 1 times) and swap executes n/2 times.

So we can take the runtime to be n/2 + n/2 - 1 + n/2 in this case. So 3/2n - 1

(This isn't required) To find an asymptotic lower bound Omega, we can find a function that has a constant real number C_1 and constant integer n_0 for which $f(n) = \frac{3}{2}n - 1$ is always greater than or equal to.

Step		Commentary
1	$n \geq 2$	For $n_0 = 2$
3	$3n \geq 2n + 2$	Add 2n to both sides.
4	$3n - 2 \geq 2n$	Add 2 to both sides
5	$\frac{3}{2}n - 1 \geq n$	Divide both sides by 2. Since n is positive it doesn't affect the order of the inequality.
6	$f(n) \geq 1 \cdot n$	Replace the value with f(n)
7	$f(n) \geq 1 \cdot g(n)$	By definition, we know that $f(n) = \Omega(g(n))$ if $c_1 \cdot g(n) \leq f(n)$ *for all* $n \geq n_0$ Therefore, for $C_1 = 1$ and $n_0 = 2$, $f(n) = \Omega(n)$

Therefore, the algorithm is asymptotically lower bounded by n. In other words, $\Omega(n)$. In worst case analysis, we consider the worst case scenario even when looking for the lower bound.

e.

Let's find the matching asymptotic upper bound. We said that the complexity of the algorithm can be (roughly) modeled by $\frac{3}{2}n - 1$.

Step		Commentary
1	$n \geq 0$	
2	$n \geq -2$	Since $n \geq 0$, we can confidently say that $n \geq -2$
3	$\frac{1}{2}n \geq -1$	Divide both sides by 2
4	$\frac{1}{2}n + \frac{3}{2}n \geq -1 + \frac{3}{2}n$	Add 1.5n to both sides
5	$2n \geq \frac{3}{2}n - 1$	Sum everything up
6	$2 \cdot g(n) \geq f(n)$	Replace the values with *f*(n) and *g*(n). By definition, we know $f(n) = O(g(n))$ if $c_2 \cdot g(n) \geq f(n)$ *for all* $n \geq n_0$ Therefore, if we select $C_1 = 2$ and $n_0 = 2$, we get $f(n) = O(n)$

Therefore, considering the worst case scenario, the complexity of the algorithm is asymptotically upper bounded by n. In other words, $O(n)$.

Question 4

a) Exercise 5.1.1

b.

The goal is to find the number of strings of length 7, 8, or 9. Characters can be special characters, digits, or letters.

We know that the strings can have length 7, 8, or 9.

As such,

$$\#\ passwords\ =\ |strings\ of\ length\ 7|\ +\ |strings\ of\ length\ 8|\ +\ |strings\ of\ length\ 9|$$

We know that there are no restrictions on reusing characters. So the strings are a sequence of the total characters options.

$$\#\ passwords\ =\ (character\ options)^7\ +\ (character\ options)^8\ +\ (character\ options)^9$$

We know that characters can be digits, letters, special characters and that:

- There are 26 letters
- There are 10 digits
- There are 4 special characters

As such,

$$character\ options\ =\ \#\ letters\ +\ \#\ digits\ +\ \#\ special\ characters$$

$$character\ options\ =\ 26\ +\ 10\ +\ 4$$

$$character\ options\ =\ 4$$

Hence, we can plug the value for the character options into the formula to get the number of passwords:

$$\#\ passwords\ =\ (character\ options)^7\ +\ (character\ options)^8\ +\ (character\ options)^9$$

$$\#\ passwords\ =\ 40^7\ +\ 40^8\ +\ 40^9$$

Therefore, the number of passwords is $40^7 + 40^8 + 40^9$

c.

The goal is to find the number of strings of length 7, 8, or 9. Characters can be special characters, digits, or letters. The first character cannot be a letter.

We know that the strings can have length 7, 8, or 9.

As such, $\#\ passwords\ =\ |strings\ of\ length\ 7|\ +\ |strings\ of\ length\ 8|\ +\ |strings\ of\ length\ 9|$

We know that we are not allowed to use a letter for the first character. So the first letter of each string will be different than the following characters.

$$\#\ passwords\ =\ (first\ character)\cdot(character\ options)^6\ +\ (first\ character)\cdot(character\ options)^7$$

$$+\ (first\ character)\cdot(character\ options)^9$$

We know that characters can be digits, letters, special characters and that:

- There are 26 letters
- There are 10 digits
- There are 4 special characters

As such, $character\ options\ =\ \#\ letters\ +\ \#\ digits\ +\ \#\ special\ characters$

$$character\ options\ =\ 26\ +\ 10\ +\ 4$$

$$character\ options\ =\ 4$$

We know that the first letter cannot be a letter. As such, it must be a digit or a special character:

$$first\ letter\ =\ \#\ digits\ +\ \#\ special\ characters$$

$$first\ letter\ =\ 10\ +\ 4$$

$$first\ letter\ =\ 14$$

Hence, we can plug the values into the formula to get the number of passwords:

$$\#\ passwords\ =\ (first\ character)\cdot(character\ options)^{6}\ +\ (first\ character)\cdot(character\ options)^{7}$$

$$+\ (first\ character)\cdot(character\ options)^{9}$$

$$\#\ passwords\ =\ 14\cdot 40^{6}\ +\ 14\cdot 40^{7}\ +\ 14\cdot 40^{8}$$

$$\#\ passwords\ =\ 14\cdot(40^{6}\ +\ 40^{7}\ +\ 40^{8})$$

Therefore, the number of passwords is $14\cdot(40^{6}\ +\ 40^{7}\ +\ 40^{8})$

b) Exercise 5.3.2

a.

The goal is to determine the number of possible strings of length 10 in which characters come from the set {a, b, c} and in which no two consecutive characters are the same.

We know that:

- The first character can be any of the 3 items in the set {a, b, c}.
- Every character after that must be one of the two other characters that are different than the one that precedes it.

As such,

$$\#\ strings\ =\ 3\cdot 2\cdot 2\cdot 2\cdot 2\cdot 2\cdot 2\cdot 2\cdot 2\cdot 2$$

$$\#\ strings\ =\ 3\cdot 2^{9}$$

$$\#\ strings\ =\ 1536$$

Therefore, the number of strings over the set {a, b, c} that have length 10 in which no two consecutive characters are the same is $3\cdot 2^{9}\ =\ 1536$.

c) Exercise 5.3.3

b. How many license plate numbers are possible if no digit appears more than once?

We know that:

- A license plate number in this state can be any string of the form: Digit-Letter-Letter-Letter-Letter-Digit-Digit
- There are 10 possible digits
- There are 26 possible uppercase letters

Since we can not reuse any digit or letter more than once, the number of choices for digits decrements by 1 following every digit selection.

We get:

$$\#\ plates = (\#\ digits) \cdot (\#\ letters) \cdot (\#\ letters) \cdot (\#\ letters) \cdot (\#\ letters) \cdot (\#\ digits - 1) \cdot (\#\ digits - 2)$$

$$\#\ plates = 10 \cdot 26 \cdot 26 \cdot 26 \cdot 26 \cdot (10 - 1) \cdot (10 - 2)$$

$$\#\ plates = 10 \cdot 26 \cdot 26 \cdot 26 \cdot 26 \cdot 9 \cdot 8$$

$$\#\ plates = 10 \cdot 26 \cdot 26 \cdot 26 \cdot 26 \cdot 9 \cdot 8$$

$$\#\ plates = 10 \cdot 9 \cdot 8 \cdot 26^4$$

$$\#\ plates = P(10,\ 3) \cdot 26^4$$

$$\#\ plates = 329\ 022\ 720$$

Therefore, the number of license plates is $P(10,\ 3) \cdot 26^4$

c. How many license plates are possible if no digit or letter appears more than once?

We know that:

- A license plate number in this state can be any string of the form:
  Digit-Letter-Letter-Letter-Letter-Digit-Digit
- There are 10 possible digits
- There are 26 possible uppercase letters

Since we can not reuse any digit or letter more than once, the number of choices for digits decrements by 1 following every digit selection and the number of choices for letters decrements by 1 following every letter selection.

We get:

$$\#\ plates = (\#\ digits) \cdot (\#\ letters) \cdot (\#\ letters - 1) \cdot (\#\ letters - 2) \cdot (\#\ letters - 3) \cdot (\#\ digits - 1) \cdot (\#\ digits - 2)$$

$$\#\ plates = 10 \cdot 26 \cdot (26 - 1) \cdot (26 - 2) \cdot (26 - 3) \cdot (10 - 1) \cdot (10 - 2)$$

$$\#\ plates = 10 \cdot 26 \cdot 25 \cdot 24 \cdot 23 \cdot 9 \cdot 8$$

$$\#\ plates = 10 \cdot 9 \cdot 8 \cdot 26 \cdot 25 \cdot 24 \cdot 23$$

$$\#\ plates = 10 \cdot 9 \cdot 8 \cdot 26 \cdot 25 \cdot 24 \cdot 23$$

$$\#\ plates = P(10,\ 3) \cdot P(26,\ 4)$$

$$\#\ plates = 720 \cdot 358,800$$

$$\#\ plates = 258,336,000$$

Therefore, the number of license plates is $P(10,\ 3) \cdot P(26,\ 4)$

d) Exercise 5.2.3

a.

We have that:

- B^9 is the set of binary strings with 9 bits.
- E_{10} is the set of binary strings with 10 bits that have an even number of 1's.

Let's define the function (note: E_9 is the subset of B^9 where the number of 1's is even):

$$f : B^9 \rightarrow E_{10} \begin{cases} f(s) = s0 \text{ if } s \in E_9 \\ f(s) = s1 \text{ otherwise} \end{cases}$$

Let's show that this function is a bijection between B^9 and E_{10} :

Step 1: Explain that f is one-to-one:

We know that:

- Every string *s* in B^9 is unique (has a different sequence of characters).
- Given that every string *s* is unique, any input *s* concatenated with a character *a* will produce a unique output *sa*.

As such, given that our function *f* takes an input string *s,* all of which are unique and appends a 0 or a 1 depending on the number of 1's in the string s, the function *f* produces a unique output *s0* or *s1* for every input.

Therefore, *f* is one-to-one.

- Second Proof (from Ian OH):

Consider $f(x) = a$ and $f(y) = b$.
Since a and b are 10 bit strings, if a = b then each bit in a is equal to the bit in the equivalent position in b.
As such, the first nine bits in a must be equal to the first 9 bits in b, let's call it s.
Since f takes a 9 bit input k and appends a 0 or 1 to the end, the first 9 bits of the output are the input k. Therefore, x = s and y = s.

We conclude that x must be equal to y if a = b, so f is one-to-one.

<u>Step 2: Explain that f is onto:</u>

We know that:

- B^9 encompasses all possibilities for 9 bit strings. So we have all possible character combinations for the first 9 bits accounted for.
- Depending on the number of 1's in the string s, the function f produces a unique output $s0$ or $s1$ for every input. Since the number of possible binary characters are $\{0,1\}$, f accounts for both possibilities for the tenth digit.
    - The only caveat is that the function f does not output $s1$ for all inputs that have an even number of 1's or $s0$ for all inputs that have an odd number of 1's. This is okay given that both of those string possibilities would have an odd number of 1's and therefore not be in the target E_{10} (even if they are members of B^{10}).

Therefore, given that our function f can produce all possible values for E_{10}, f is onto.

Therefore, given that f is one-to-one and onto, it is a bijection between B^9 *and* E_{10}.

b.

Since f is a bijection between $B^9 \text{ and } E_{10}$, we get:

$$\left|E_{10}\right| = \left|B^9\right|$$

$$\left|E_{10}\right| = \{0,1\}^9$$

$$\left|E_{10}\right| = 2^9$$

$$\left|E_{10}\right| = 512$$

Therefore, $\left|E_{10}\right|$ *is* 512.

Question 5

a) Exercise 5.4.2

a.

We know that:

- All phone numbers are 7-digits long
- All phone numbers start with either 824 or 825.

So:

$$\#\ phones\ =\ (options\ for\ digit\ 1)\ *\ (options\ for\ digit\ 2)\ *\ (options\ for\ digit\ 3)\ *$$

$$(options\ for\ digit\ 4)\ *\ (options\ for\ digit\ 5)\ *\ (options\ for\ digit\ 6)\ *\ (options\ for\ digit\ 7)$$

Since all phone numbers start with 824 or 825, this implies that:

- The first digit is always 8. There is 1 option.
- The second digit is always 2. There is 1 option.
- The third digit can be 4 or 5. There are 2 options.
- Digits 4-7 can be any digit. There are 10 options.

Plugging these values into the formula gives:

$$\#\ phones\ =\ 1 \cdot 1 \cdot 2 \cdot 10^4$$

$$\#\ phones\ =\ 2 \cdot 10^4$$

$$\#\ phones\ =\ 20,000$$

Therefore, there are 20,000 possible phone numbers for the university.

b.

This problem can be interpreted in two ways:

1) How many different phone numbers are there in which the last four digits are all different from each other? (This is what I think it means)
2) How many different phone numbers are there in which the last four digits are each different from all the digits that preceded them (including 824 and 825)?

Solving the first interpretation of the problem:

We know that:

- All phone numbers are 7-digits long
- All phone numbers start with either 824 or 825.
- The last four digits are all different.

So: $\#\ phones\ =\ (options\ for\ digit\ 1)\ *\ (options\ for\ digit\ 2)\ *\ (options\ for\ digit\ 3)\ *$

$(options\ for\ digit\ 4)\ *\ (options\ for\ digit\ 5)\ *\ (options\ for\ digit\ 6)\ *\ (options\ for\ digit\ 7)$

Since all phone numbers start with 824 or 825, this implies that:

- The first digit is always 8. There is 1 option.
- The second digit is always 2. There is 1 option.
- The third digit can be 4 or 5. There are 2 options.
- The fourth digit can be any digit. There are 10 options.
- The fifth digit can be any digit except digit 4. There are 9 options.
- The sixth digit can be any digit except digits 4, 5. There are 8 options.
- The seventh digit can be any digit except digits 4, 5, 6. There are 7 options.

Plugging these values into the formula gives:

$$\#\ phones\ =\ 1 \cdot 1 \cdot 2 \cdot 10\ \cdot 9 \cdot 8 \cdot 7$$

$$\#\ phones\ =\ 2 \cdot 10 \cdot 9 \cdot 8 \cdot 7$$

$$\#\ phones\ =\ 2 \cdot P(10,\ 4)$$

$$\#\ phones\ =\ 10{,}080$$

Therefore, there are 10,080 possible phone numbers for the university.

Solving the second interpretation of the problem:

We know that:

- All phone numbers are 7-digits long
- All phone numbers start with either 824 or 825.
- The last four digits are all different than all preceding digits.

So: $\#\ phones\ =\ (options\ for\ digit\ 1)\ *\ (options\ for\ digit\ 2)\ *\ (options\ for\ digit\ 3)\ *$

$(options\ for\ digit\ 4)\ *\ (options\ for\ digit\ 5)\ *\ (options\ for\ digit\ 6)\ *\ (options\ for\ digit\ 7)$

Since all phone numbers start with 824 or 825, this implies that:

- The first digit is always 8. There is 1 option.
- The second digit is always 2. There is 1 option.
- The third digit can be 4 or 5. There are 2 options.
- The fourth digit can be any digit except the digits in position 1,2,3. There are 7 options.
- The fifth digit can be any digit except the digits in position 1,2,3,4. There are 6 options.
- The sixth digit can be any digit except the digits in position 1,2,3,4,5. There are 5 options.
- The seventh digit can be any digit except the digits in 1,2,3,4,5,6. There are 4 options.

Plugging these values into the formula gives:

$$\#\ phones\ =\ 1 \cdot 1 \cdot 2 \cdot 7 \cdot 6 \cdot 5 \cdot 4$$

$$\#\ phones\ =\ 2 \cdot 7 \cdot 6 \cdot 5 \cdot 4$$

$$\#\ phones\ =\ 2 \cdot P(7,\ 4)$$

$$\#\ phones\ =\ 1680$$

Therefore, there are 1680 possible phone numbers for the university.

b) Exercise 5.5.3

a.

For ten bit strings with no restrictions, each of the digits can be 0 or 1.

Therefore, the number of bit strings is 2 x 2 x 2 … 2 (10 times) = 2^{10} = 1024

b.

For ten bit strings that start with 001:

- The first digit must be 0
- The second digit must be 0
- The third digit must be 1
- All other digits can be 0 or 1.

Therefore, the number of bit strings is 1 x 1 x 1 x 2 … 2 (7 times) = 2^7 = 128

c.

For ten bit strings that start with 001:	For ten bit strings that start with 10:
• The first digit must be 0 • The second digit must be 0 • The third digit must be 1 • All other digits can be 0 or 1.	• The first digit must be 1 • The second digit must be 0 • All other digits can be 0 or 1.

Combining the two, for ten bit strings that start with 001 or 10:

strings = (10 bit strings that start with 001) + (10 bit strings that start with 10)

strings = (1 x 1 x 1 x 2 x 2 x 2 x 2 x 2 x 2 x 2) + (1 x 1 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2)

strings = $2^7 + 2^8$

strings = 384

d.

For ten bit strings in which the first two bits are the same as the last two bits:

- The first 8 digits can be 0 or 1. So 2 options
- The 9th digit must be the same as the first digit. So 1 option.
- The 9th digit must be the same as the second digit. So 1 option.

Therefore, the number of bit strings is 2 … 2 (8 times) x 1 x 1 $= 2^8 = 256$

e.

For ten bit strings that have exactly six 0's:

- Any 4 digits can be 1.
- Any 6 digits can be 0.

So $\#\ strings\ = C(10,4)$

$$\#\ strings\ = \frac{10!}{4!6!}$$

$$\#\ strings\ = \frac{10 \cdot 9 \cdot 8 \cdot 7}{4 \cdot 3 \cdot 2 \cdot 1}$$

$$\#\ strings\ = 210$$

Therefore, the number of 10 bit strings that have exactly six 0's is 210.

f.

For ten bit strings that have exactly six 0's and the first bit is 1:

- The first digit must be 1.
- Any 3 other digits can be 1.
- Any 6 other digits can be 0.

$$\#\ strings\ =\ 1 \cdot C\,(9,3)$$

$$\#\ strings\ = \frac{9!}{3!6!}$$

$$\#\ strings\ = \frac{9 \cdot 8 \cdot 7}{3 \cdot 2 \cdot 1}$$

$$\#\ strings\ = 84$$

Therefore, the number of 10 bit strings that have exactly six 0's and start with 1 is 84.

g.

For ten bit strings that have exactly one 1 in the first half and exactly three 1's in the second half:

- First half is 5-bits long:
  - Any one digit in the first half can be 1.
  - The other 4 digits in the first half must be 0.
- Second half is 5-bits long:
  - Any 3 digits in the second half can be 1.
  - The other 2 digits must be 0.

So $\#\ strings\ =\ C(5,1) \cdot C\,(5,3)$

$$\#\ strings\ = \frac{5!}{1!4!} \cdot \frac{5!}{3!2!}$$

$$\#\ strings\ = 5 \cdot \frac{5 \cdot 4}{2 \cdot 1}$$

$$\#\ strings\ = 50$$

Therefore, the number of ten bit strings that have exactly one 1 in the first half and exactly three 1's in the second half is 50.

c) Exercise 5.5.5

a.

We need to select:

- 10 boys from a set of 30 boys
- 10 girls from a set of 35 girls

Step 1: Select any 10 out of 30 boys

$$\#\ boys\ =\ C(30,\ 10)$$

$$\#\ boys\ =\ \frac{30!}{10!20!}$$

$$\#\ boys\ =\ 30{,}045{,}015$$

Step 2: Select any 10 out of 35 girls

$$\#\ girls\ =\ C(35,\ 10)$$

$$\#\ girls\ =\ \frac{35!}{10!25!}$$

$$\#\ girls\ =\ 183{,}579{,}396$$

Step 3: Combine the two with the product rule

$$\#\ chorus\ =\ \#\ boys \cdot \#\ girls$$

$$\#\ chorus\ =\ C(30,\ 10) \cdot C(35,\ 10)$$

Therefore, there are $C(30,\ 10) \cdot C(35,\ 10)$ ways to select the chorus.

d) Exercise 5.5.8

c. How many five-card hands are made entirely of hearts and diamonds?

We know that:

- There are 52 cards in a deck
- There are 4 suits: diamonds. spades, hearts, clubs.

As such, each suit has 52/4 = 13 cards.

So the total number of hearts and spades is 13 * 2 = 26 cards.

To select a five card hand from the 26 possibilities, we can do it in C(26, 5) ways.

So $\#\ heartSpadeHands\ =\ C(26,5)$

$\#\ heartSpadeHands\ =\ 65,780$

Therefore, $C(26,5)$ five-card hands are made entirely of hearts and diamonds

d. How many five-card hands have four cards of the same rank?

We know that:

- There are 13 ranks.
- There are four cards for each rank.

Step 1: Choose a rank

To select 1 rank from the 13 options, we can do it C(13, 1) ways.

Step 2: Choose four cards from that rank

To select 4 cards from the 4 options, we can do it C(4, 4) ways.

Step 3: Choose the remaining card

We already selected 4 cards, so the deck has 52 - 4 = 48 cards left.

To select 1 card from the 48 remaining options, we can do it C(48, 1) ways.

$$\#fourOfAKindHands = C(13,1) \cdot C(4,4) \cdot C(48,1)$$

$$\#fourOfAKindHands = 13 \cdot 48$$

$$\#fourOfAKindHands = 624$$

Therefore, 624 five-card hands have four cards of the same rank.

e. How many five-card hands contain a full house?

A full house contains:

- 2 cards of the same rank.
- 3 cards of the same (but different than the other 2) rank.

Step 1: Choose a rank for the three of a kind

To select 1 rank from the 13 options, we can do it C(13, 1) ways.

Step 2: Choose three cards from that rank

To select 3 cards from the 4 options, we can do it C(4, 3) ways.

Step 3: Choose a rank for the pair

We already selected one rank for the three of a kind, so there are 13 - 1 = 12 ranks left.

To select 1 rank from the 12 options, we can do it C(12, 1) ways.

Step 2: Choose two cards from that rank

To select 2 cards from the 4 options, we can do it C(4, 2) ways.

So $\#fullHouses = C(13,1) \cdot C(4,3) \cdot C(12,1) \cdot C(4,2)$

$\#fullHouses = 13 \cdot 4 \cdot 12 \cdot 6$

$\#fullHouses = 3744$

Therefore, there are $C(13,1) \cdot C(4,3) \cdot C(12,1) \cdot C(4,2) = 3744$ five-card hands that contain a full house.

f. How many five-card hands do not have any two cards of the same rank?

We know that:

- There are 13 total ranks.
- There are 4 cards in a rank.

Step 1: Select 5 ranks

To select 5 ranks from the 13 options, we can do it C(13, 5) ways.

Step 2: Select a card from that rank

To select 1 card from the 4 card options within the rank, we can do it C(4, 1) ways for each card in the hand. Since there are 5 cards in the hand with 5 different ranks, we do this 5 times:

$$C(4,1) \cdot C(4,1) \cdot C(4,1) \cdot C(4,1) \cdot C(4,1)$$

Step 3: Determine the total number of possibilities

$$\#\ noSameCardsHands\ =\ C(13,5) \cdot C(4,1) \cdot C(4,1) \cdot C(4,1) \cdot C(4,1) \cdot C(4,1)$$

$$\#\ noSameCardsHands\ =\ C(13,5) \cdot 4 \cdot 4 \cdot 4 \cdot 4 \cdot 4$$

$$\#\ noSameCardsHands\ = C(13,5) \cdot 4^5$$

$$\#\ noSameCardsHands\ = 1,317,888$$

Therefore, there are $C(13,5) \cdot 4^5$ five-card hands that do not have any two cards of the same rank.

e) Exercise 5.6.6

a.

To select a 10 person committee with an equal number of Demonstrators and Repudiators, there must be 5 Demonstrators and 5 Repudiators on the committee.

Step 1: Select 5 Demonstrators

To select 5 Demonstrators from the 44 senate members belonging to that party, we can do this C(44, 5) ways.

Step 1: Select 5 Repudiators

To select 5 Repudiators from the 56 senate members belonging to that party, we can do this C(56, 5) ways.

Step 3: Combine with the product rule

$$\#\ senateCommittees\ =\ \#demSelections \cdot \#RepSelections$$

$$\#\ senateCommittees\ =\ C(44,5) \cdot C(56,5)$$

Therefore, there are $C(44,5) \cdot C(56,5)$ ways of selecting the committee.

b.

Step 1: Select Demonstrator speaker and vice speaker

To select a speaker and a vice speaker from the 44 senate members belonging to the Demonstrators party, we can do this P(44, 2) ways since the order of who is speaker and vice speaker matters.

Step 1: Select 5 Repudiators

To select a speaker and a vice speaker from the 56 senate members belonging to the Repudiators party, we can do this P(56, 2) ways since the order of who is speaker and vice speaker matters.

Step 3: Combine with the product rule

$$\#\ senateCommittees\ =\ \#demSelections \cdot \#RepSelections$$

$$\#\ senateCommittees\ =\ P(44,\ 2) \cdot P(56,\ 2)$$

Therefore, there are $P(44,\ 2) \cdot P(56,\ 2)$ ways of selecting the two speakers and vice-speakers from the parties.

Question 6

a) Exercise 5.7.2

a. How many 5-card hands have at least one club?

We can count by complement: The number of 5-card hands that have at least one club is equal to the total number of hands minus the number of 5-card hands that do not have any clubs.

Let's name the hands that have at least one club C and the total number of hands T. We get:

$$|C| = |T| - |\overline{C}|$$

Step 1: Determine the total number of hands

Since there are 52 cards in the deck, the total number of 5-card hands is C(52,5)

Step 2: Determine the number of hands that do not have a club

There are 4 suits with 13 cards each in the deck.

Since we are not considering clubs. There are 13 * 3 = 39 possible cards to choose from,

So the number of 5-card hands that do not have any clubs is C(39,5).

Step 3: Determine the number of hands with at least one club

$$|C| = |T| - |\overline{C}|$$

$$|C| = C(52,5) - C(39,5)$$

$$|C| = 2,023,203$$

Therefore, the number of hands with at least one club is equal to $C(52,5) - C(39,5)$.

b. How many 5-card hands have at least two cards with the same rank?

We can count by complement: The number of 5-card hands that have at least two cards with the same rank is equal to the total number of hands minus the number of 5-card hands that do not have any clubs.

Let's name the hands that have at least two cards of the same rank S and the total number of hands T. We get:

$$|S| = |T| - |\bar{S}|$$

Step 1: Determine the total number of hands

Since there are 52 cards in the deck, the total number of 5-card hands is C(52,5)

Step 2: Determine the number of hands that do not have any two cards with the same rank

To select 5 ranks from the 13 options, we can do it C(13, 5) ways.

To select 1 card from the 4 card options within the rank, we can do it C(4, 1) ways for each card in the hand. Since there are 5 cards in the hand with 5 different ranks, we do this 5 times:

$$|\bar{C}| = C(13,5) \cdot C(4,1) \cdot C(4,1) \cdot C(4,1) \cdot C(4,1) \cdot C(4,1)$$

$$|\bar{C}| = C(13,5) \cdot 4^5$$

Step 3: Determine the number of hands with at least one club

$$|C| = |T| - |\bar{C}|$$

$$|C| = C(52,5) - \left(C(13,5) \cdot 4^5\right)$$

$$|C| = 1,281,072$$

Therefore, the number of hands with at least one club is equal to $C(52,5) - \left(C(13,5) \cdot 4^5\right)$.

b) Exercise 5.8.4

a. **How many ways are there to distribute 20 comic books to 5 kids if there are no restrictions on how many go to each kid?**

Since there are no restrictions on how many books go to each kid. Every book can go to each of the 5 kids.

So first book = 5 possibilities

Second book = 5 possibilities

.
.
.

Twentieth book = 5 possibilities

In total, there are a total of 5^{20} ways to distribute the comic books.

b. **How many ways are there to distribute the comic books if they are divided evenly so that 4 go to each kid?**

Since we are dividing the 20 books to 20 = 4 + 4 + 4 + 4 + 4.

We can count permutations with repetition:

$$\#\ possibilities\ =\ \frac{Total\ number\ of\ books!}{books\ to\ kid\ 1!\ books\ to\ kid\ 2!\ books\ to\ kid\ 3!\ books\ to\ kid\ 4!\ books\ to\ kid\ 5!}$$

$$\#\ possibilities\ =\ \frac{20!}{4!4!4!4!4!}$$

$$\#\ possibilities\ =\ \frac{20!}{(4!)^5}$$

Therefore, the total number of possibilities is $\frac{20!}{(4!)^5}$

Question 7

a)

For a function to be one-to-one, the target must at least be the same size as the domain so that there are enough elements in the target that the elements in the domain can map to.

In this case:

- Domain: 5 elements
- Target: 4 elements

Since domain > target, there is no one-to-one function that maps elements from the domain to the target.

Therefore, the number of one-to-one functions whose domain has 5 elements and target has 4 elements is 0.

b)

Following the framework of Zybooks:

A function is defined in terms of its output for every possible element in the domain.

Step 1: Start with the first element in the domain d_1 and select the value for $f(d_1)$.

There are 5 possible choices because $f(d_1)$ can be any element from the target set with 5 elements.

Step 2: After $f(d_1)$ has been chosen, select the value for $f(d_2)$.

There are 5 - 1 = 4 possible choices for $f(d_2)$ because the output can be any element from the target set except the element that was chosen to be $f(d_1)$.

Step 3: Repeat for all other elements in the domain.

Select the value of f(x) for each $x \in \{t_1, t_2, t_3, t_4, t_5\}$ in such a way that there are no repetitions.

- d_3: 5-2 = 3 options
- d_4: 5-3 = 2 options
- d_5: 5-4 = 1 option

In total, we get: $\#possibilities = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$

$$\#possibilities = 5!$$

Therefore the number of one-to-one functions whose domain and target sets both have 5 elements is 5!

c)

Following the framework of Zybooks:

A function is defined in terms of its output for every possible element in the domain.

Step 1: Start with the first element in the domain d_1 and select the value for $f(d_1)$.

There are 6 possible choices because $f(d_1)$ can be any element from the target set with 6 elements.

Step 2: After $f(d_1)$ has been chosen, select the value for $f(d_2)$.

There are 6 - 1 = 5 possible choices for $f(d_2)$ because the output can be any element from the target set except the element that was chosen to be $f(d_1)$.

Step 3: Repeat for all other elements in the domain.

Select the value of f(x) for each $x \in \{t_1, t_2, t_3, t_4, t_5\}$ in such a way that there are no repetitions.

- d_3: 6-2 = 4 options
- d_4: 6-3 = 3 options
- d_5: 6-4 = 2 options

In total, we get: $\#possibilities = 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2$

$$\#possibilities = P(6,5)$$

Therefore the number of one-to-one functions whose domain has 5 elements and target set has 6 elements is P(6,5)

d)

Following the framework of Zybooks:

A function is defined in terms of its output for every possible element in the domain.

Step 1: Start with the first element in the domain d_1 and select the value for $f(d_1)$.

There are 7 possible choices because $f(d_1)$ can be any element from the target set with 7 elements.

Step 2: After $f(d_1)$ has been chosen, select the value for $f(d_2)$.

There are 7 - 1 = 6 possible choices for $f(d_2)$ because the output can be any element from the target set except the element that was chosen to be $f(d_1)$.

Step 3: Repeat for all other elements in the domain.

Select the value of f(x) for each $x \in \{t_1, t_2, t_3, t_4, t_5\}$ in such a way that there are no repetitions.

- d_3: 7-2 = 5 options
- d_4: 7-3 = 4 options
- d_5: 7-4 = 3 options

In total, we get: $\#possibilities = 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3$

$$\#possibilities = P(7,5)$$

Therefore the number of one-to-one functions whose domain has 5 elements and target set has 7 elements is P(7,5)