ECE345 HW1

Homework Group #85
Yanni Alan Alevras | yanni.alevras@mail.utoronto.ca | 1009330706
Evan Banerjee | evan.banerjee@mail.utoronto.ca | 1009682309
Gautam | gautam.makheja@mail.utoronto.ca | 1008788974

Total Pages: 16

Friday 20$^{\text{th}}$ September, 2024

1 Question 1, Graphs and Combinations, Permutations

a.

Theorem. Give a combinatorial argument to prove that

$$\sum_{k=0}^{n} \binom{n}{k} 2^k = 3^n$$

Proof. Combinatorial Argument

Imagine you have a bucket with three balls in it, where each ball is either red, green, or blue.

let's say each time you pick one of the balls out of the bucket, you make a dash on a piece of paper using a marker with the same color as the color of the ball. Let's also say that you put the ball back after making each dash. Assume that you write the dashes in a straight line so that after repeating this process n times, you end up with a well defined sequence of n dashes, where each dash is either red, green, or blue.

since each dash was one of three possible choices, there are 3^n possible sequences of colored dashes that you can draw with this process.

Each colored dash has 2 properties, its color, and its index in the sequence.
For example, if n = 5, one possible order could be RGGRB. in this case, you could represent this unique order as

(G, 3), (R, 1), (G, 2), (R, 4), (B, 5)

where, in each tuple, the letter is the color and the number is the index of that color in the sequence.

Clearly the index of each colored dash is going to be a positive integer between 1 and n

Because each unique sequence of n colored dashes can be represented by a unique, unordered list of color-index tuples, any procedure that can generate this list of tuples is mathematically identical to the process of drawing the sequences by choosing colored balls.

Now imagine you had a bucket with n numbered tokens in it. Each token has number written on it ranging from 1 to n.

One possible way to create an unordered list of color-index tuples (and as a result, a way to create the original sequence of colored dashes) would be to first decide how many red tuples you wanted ahead of time, create those tuples by randomly selecting indices from the bucket, and then randomly choose between green and blue as you made the remaining tuples

Again, assuming n = 5, you could start out by saying that you will have exactly 2 red tuples. To do this, you could randomly choose 2 tokens from the bucket without replacement.

Whatever numbers you pulled out of the bucket, you would create red tuples using those numbers as your indices. For example, if the numbers you pulled out of the bucket were 1 and 4, then you would have the tuples (R, 1) and (R, 4). This exactly defines the red dashes in the sequence presented earlier in this proof.

However, there's no reason why the tokens pulled from the bucket had to be 1 and 4. Since you're choosing 2 tokens from a set of 5 tokens without replacement, there are $\binom{5}{2}$ possible combinations 2 red tuples.

After forming these red tuples, there are now 3 indices left. Each of these indices could be paired with either the color green, or the color blue.

You could choose to keep pulling numbers out of the bucket, assigning each index token that you pull out to either green or blue based on the toss of a coin

However, another equally valid approach would be to just pour out the remaining tokens, line them up in ascending order, and then flip a coin as you look at each token one by one

Heads means the index makes a tuple with green, tails means it makes a tuple with blue.

If the remaining indices after making the red tuples were 2, 3, and 5, and if the three coin tosses had given the sequence H, H, T, then we would end up with the tuples

(G, 2), (G, 3), (B, 5)

Which represents the blue and green dashes in the example shown earlier in this proof.

Because there are 2 possible outcomes for each of the 3 remaining tokens (Heads or Tails), the number of possible combinations of green and blue tuples is 2^3.

Since there are $\binom{5}{2}$ combinations of red tuples, and 2^3 combinations of green and blue tuples, there are $\binom{5}{2} * 2^3$ possible combinations of 5 index-color tuples where 2 have the color red

This means that there are $\binom{5}{2} * 2^3$ possible sequences of 5 colored dashes (from the colors red, green, and blue) where two of those dashes are red

However, if we go back to the tuple generation, there was no reason why we had to have exactly two red dashes. We could have had no red dashes, or five red dashes, or any number in between

Let's define k as the number of red tuples (where $k \in [0, 5]$). for each k, there are $\binom{5}{k}$ possible combinations of k red tuples, and since there are 5-k remaining indices, we'll end up with 2^{5-k} unique combinations or green and blue tuples. In other words, there are $\binom{5}{k} * 2^{5-k}$

Every unique combination of tuples will have some amount of red tuples, with the exact amount being between 0 and 5 inclusive.

As a result, if we were to sum up all the possible combinations of tuples as the number of red tuples (k) goes from 0 to 5, then we would end up with every possible combination of 5 tuples formed from the colors red, green, and blue.

This sum would look like:

$(\binom{5}{0} * 2^5) + (\binom{5}{1} * 2^4) + (\binom{5}{2} * 2^3) + (\binom{5}{3} * 2^2) + (\binom{5}{4} * 2^1) + (\binom{5}{5} * 2^0)$

Which is the same as:

$$\sum_{k=0}^{5} \binom{5}{k} 2^{5-k}$$

Due to the equivalency of unique tuple combinations and unique RGB color sequences, this sum is also equal to the number of unique sequences of 5 colored dashes.

However, there's no reason why we had to choose 5 as our number of tuples. This process works just as well for making n unique color-index tuples using red, green, and blue.

If we choose to make n unique color-index tuples, and we choose to make k red tuples, then there are $\binom{n}{k}$ unique combinations of red tuples, and 2^{n-k} unique combinations of green and blue tuples.

this means there are $\binom{n}{k} * 2^{5-k}$ unique combinations of n tuples where k tuples are red, and

$$\sum_{k=0}^{n} \binom{n}{k} 2^{n-k}$$

unique combinations of n color-index tuples as a whole

Again, due to the equivalency between tuple combinations and RGB sequences, this formula also describes the number of unique sequences formed from red, green, and blue dashes.

And with a little bit of algebra, it's easy to see that:

$$\begin{aligned}\binom{n}{k} &= \frac{n!}{k!(n-k)!} \\ &= \frac{n!}{(n-k)!(k)!} \\ &= \frac{n!}{(n-k)!(n-n+k)!} \\ &= \frac{n!}{(n-k)!(n-(n-k))!} \\ &= \binom{n}{n-k}\end{aligned}$$

Which makes intuitive sense because the number of ways to choose k elements to use out of a set of n (without replacement) is the same as the number of ways to choose n - k elements not to use out of a set of n (without replacement)

Using this fact, we can rewrite

$$\sum_{k=0}^{n} \binom{n}{k} 2^{n-k}$$

as

$$\sum_{k=0}^{n} \binom{n}{n-k} 2^{n-k}$$

We can also easily make the substitution $l = n - k$. When k is 0, $l = n$, and when k is n, $l = 0$. This lets us rewrite the formula as

$$\sum_{l=n}^{0} \binom{n}{l} 2^{l}$$

But it doesn't really matter the order in which we evaluate a sum. After all, $a + b + c = c + b + a$, so we can say:

$$\sum_{l=n}^{0} f(l) = \sum_{l=0}^{n} f(l)$$

which allows us to rewrite our expression for the number of unique color-index tuples as:

$$\sum_{l=0}^{n} \binom{n}{l} 2^l$$

which is the same as

$$\sum_{k=0}^{n} \binom{n}{k} 2^k$$

Since this is the number of unique combinations of color-index tuples for n tuples, it is also the number of unique sequences of n dashes where each dash is either red, green, or blue.

Finally, the number of unique RGB sequences of n dashes is 3^n (as previously proven) which means that:

$$\sum_{k=0}^{n} \binom{n}{k} 2^k = 3^n$$

□

b. Time complexity: $\mathcal{O}(n)$

Proof. ($\Leftarrow$) Assume by Contradiction,

($\Rightarrow$) □

2 Question 2 Example Pseudocode

```
1: function GALE-SHAPLEY(E, S)
2:     initialize all employers in E and students in S to unmatched
3:     while an unmatched employer with at least one student on its preference list remains do
4:         choose such an employer e ∈ E
5:         make offer to next student s ∈ S on e's preference list
6:         if s is unmatched then
7:             Match e with s                                  ▷ s accepts e's offer
8:         else if s prefers e to their current employer e′ then
9:             Unmatch s and e′                                ▷ s rejects e′
10:            Match e with s                                  ▷ s accepts e's offer
11:        end if
12:        cross s off e's preference list
13:    end while
14:    report the set of matched pairs as the final matching
15: end function
```

3 Question 3, Asymptotics

Sort the following 20 functions from asymptotically smallest to asymptotically largest, indicating ties if there are any. You must provide proofs that *justify your answers* to receive full credit! All logarithms are base 2 unless otherwise stated.

$$
\begin{array}{llllll}
n^{4.5} - (n-1)^{4.5}, & n \lg \lg n, & n\sqrt{\frac{n}{2}}, & n^{2n}, & \lg(\lg^* n), \\
n^{(\lg \lg n)/(\lg n)}, & \lg^{(9001)} n, & \lg^* 2^n, & \lg(n!), & (\lg n)^{\lg n}, \\
\sum_{i=2}^{n} \frac{2}{i^2-1}, & e^{2n}, & n^{\lg \lg n}, & n^{1337}, & \lg^*(n/2), \\
(1 + \frac{1}{787898})^{787898n}, & n!, & \pi, & 2^{\lg^* n}, & (\lg n)^{\lg^* n}
\end{array}
$$

To solve this problem, I will find a tight bound, or an upper and lower bound for each function listed.

For each bound I find, I will place the corresponding function in an "Asymptotic Array" that corresponds to that bound.

If I find that the bound for the function being analyzed is the same as the bound for a function that I analyzed previously, then I will place the function in the same Asymptotic Array as the previously analyzed function.

Any two functions in the same Asymptotic Array have the same asymptotic behavior.

All asymptotic arrays will be stored in a larger array for convenince.

Once all functions are sorted, I will sort the Arrays from smallest to largest.

I will be using these properties as provided in textbook section 3.3 (c is a constant, k is a constant greater than 1):

$c << \lg^* n << (\lg n)^c << n^c << k^n$

I would also like to begin by proving $k^n << n^n$

Proof. substitute: $n = k^x$ where x is a positive variable

this means $x = \log_k n$

and as n $\to \infty$:

$$\log_k n \to \infty, \quad 1 - \log_k n \to -\infty, \quad n(1 - \log_k n) \to -\infty$$

the proof is as follows:

$$\lim_{n\to\infty}\frac{k^n}{n^n}=\lim_{x\to\infty}\frac{k^{k^x}}{(k^x)^{k^x}}=\lim_{x\to\infty}\frac{k^{k^x}}{k^{xk^x}}=\lim_{x\to\infty}\frac{1}{k^{(xk^x-k^x)}}=$$
$$\lim_{x\to\infty}\frac{1}{k^{k^x(x-1)}}=\lim_{x\to\infty}\frac{1}{k^{-k^x(1-x)}}=\lim_{x\to\infty}k^{k^x(1-x)}=\lim_{n\to\infty}k^{n(1-\log_k n)}=$$
$$\lim_{n\to\infty}k^{n(1-\log_k n)}=\lim_{m\to-\infty}k^m=0$$

so $k^n << n^n$ □

The final string of inequalities is therefore:

$c << \lg^* n << (\lg n)^c << n^c << k^n << n^n$

Let's first examine $n^{4.5}-(n-1)^{4.5}$

This is a specific case of the general function $n^k-(n-1)^k$ where k is a nonzero constant.

By the mean value theorem, if f(x) is continuous and continuously differentiable over an interval, then $f(x)-f(x-1)=1*f'(x-\epsilon)$ where $0<\epsilon<1$

And we know from calculus that:

$f(n)=n^k \implies f'(x)=kn^{k-1}$

Therefore:

$n^k-(n-1)^k=k(n-\epsilon)^{k-1}$

We can also show that if a is any constant:

$$\lim_{n\to\infty}\frac{n^k}{(n-a)^k}=\lim_{n\to\infty}(\frac{n}{(n-a)})^k=(\lim_{n\to\infty}\frac{n}{(n-a)})^k=(\lim_{n\to\infty}\frac{\frac{d}{dn}n}{\frac{d}{dn}(n-a)})^k=(1)^k=1$$

which means that if f(n) is a polynomial function, f(n-a) and f(n) are asymptotically equivalent.

putting this all together $(0<\epsilon<1)$:

$$n^k-(n-1)^k=k(n-\epsilon)^{k-1}=4.5\Theta(n^{k-1})=\Theta(n^{k-1})$$

Our example is just the case when k = 4.5 so

$n^{4.5}-(n-1)^{4.5}=\Theta(n^{3.5})$

so we have our first category:

$\{\Theta(n^{3.5}) : [n^{4.5} - (n-1)^{4.5}]\}$

Now let's look at $n \lg \lg n$

This function is equal to it's asymptotic (θ) form, which I will henceforth refer to as being in its "most simplified form". Since we can't make it simpler, we just make a new category:

$$\{\Theta(n^{3.5}) : [n^{4.5} - (n-1)^{4.5}], \quad \Theta(n \lg \lg n) : [n \lg \lg n]\}$$

Now let's look at $n\sqrt{\frac{n}{2}}$

$$n\sqrt{\frac{n}{2}} = \frac{1}{\sqrt{2}} n * n^{0.5} = \frac{1}{\sqrt{2}} n^{1.5} = \Theta(n^{1.5})$$

$$\left\{\Theta(n^{3.5}) : [n^{4.5} - (n-1)^{4.5}], \quad \Theta(n \lg \lg n) : [n \lg \lg n], \quad \Theta(n^{1.5}) : [n\sqrt{\frac{n}{2}}], \right\}$$

Now let's look at n^{2n}

This is already in its most simplified form

$$\left\{\begin{array}{l} \Theta(n^{3.5}) : [n^{4.5} - (n-1)^{4.5}], \qquad \Theta(n \lg \lg n) : [n \lg \lg n], \quad \Theta(n^{1.5}) : [n\sqrt{\frac{n}{2}}], \\ \Theta(n^{2n}) : [n^{2n}], \end{array}\right\}$$

Now let's look at $\lg(\lg^* n)$

This is already in its most simplified form

$$\left\{\begin{array}{l} \Theta(n^{3.5}) : [n^{4.5} - (n-1)^{4.5}], \qquad \Theta(n \lg \lg n) : [n \lg \lg n], \quad \Theta(n^{1.5}) : [n\sqrt{\frac{n}{2}}], \\ \Theta(n^{2n}) : [n^{2n}], \qquad\qquad\qquad \Theta(\lg(\lg^* n)) : [\lg(\lg^* n)], \end{array}\right\}$$

The rule I'm using to determine whether a function is already in its big theta form is this:

"If a function is formed entirely of subfunctions from different asymptotic bounds, and those functions do not have the capacity to cancel each other, then the function is in its most simplified asymptotic form."

For example, e^{2n} is in its most simplified form because it's formed of a polynomial function (2n) nested inside an exponential function (e^n). polynomial functions and exponential functions don't have the capacity to cancel when nested, so this must be fully simplified.

As another example of a fully simplified function: $nlgn$. It's formed from the product of a polynomial function and a logarithmic function, and these functions cannot cancel in a product, so it must already be simplified.

However, I cannot assume a function such as $\lg 5^n$ is in its most simplified form, because logarithmic functions and exponential functions have the capacity to cancel, out when nested together, so further simplification must occur

To save time, I'm going to add all the functions that are already equivalent to their big theta forms:

$$\left\{\begin{array}{lll} \Theta(n^{3.5}) : [n^{4.5} - (n-1)^{4.5}], & \Theta(n \lg\lg n) : [n \lg\lg n], & \Theta(n^{1.5}) : [n\sqrt{\frac{n}{2}}], \\ \Theta(n^{2n}) : [n^{2n}], & \Theta(\lg(\lg^* n)) : [\lg(\lg^* n)], & \Theta(\lg^{(9001)} n) : [\lg^{(9001)} n)], \\ \Theta(1) : [\pi], & \Theta(e^{2n}) : [e^{2n}], & \\ \Theta(n^{1337}) : [n^{1337}], & \Theta(2^{\lg^* n}) : [2^{\lg^* n}], & \Theta((\lg n)^{\lg^* n}) : [(\lg n)^{\lg^* n}], \end{array}\right\}$$

before further analysis, let's order this array from smallest to largest asymptotically when reading from left to right, and top to bottom

since the smallest asymptotic growth is just a constant value, $\Theta(1)$ comes first:

$$\left\{\Theta(1) : [\pi] \quad \right\}$$

then, the next growth rate in terms of asymptotic size after $\Theta(1)$ is $\Theta(\lg^* n)$. We have two relevant Theta functions here:

$\Theta(\lg(\lg^* n))$
$\Theta(2^{\lg^* n})$

Since $\lg^* n$ just acts as the input into another function, so as long as $\lg^* n$ increases without bound:

$\lg(\lg^* n) << \lg^* n$ and $2^{\lg^* n} >> \lg^* n$

as a result, the ordering will look like this:

$$\{\Theta(1) : [\pi], \quad \Theta(\lg(\lg^* n)) : [\lg(\lg^* n)], \quad \Theta(2^{\lg^* n}) : [2^{\lg^* n})], \quad \}$$

The next smallest function asymptotically is going to be $\lg^{(9001)} n$ this is because every other simplified function is at least $\Omega(\lg n)$, since $\lg f(n) << f(n)$ when f(n) increases without bound, $\lg^{(9001)} n << lg^{(9000)} n << lg^{(8999)} n << ... << \lg(\lg n) << \lg n$

now you may be wondering how we know that $\lg^{(9001)} n >> 2^{\lg^* n}$.

We can prove this by setting $n = {}^k 2 = 2^{2^{2^{\cdot^{\cdot^{\cdot^2}}}}}$ with k 2s in total on the tower of 2.

Then $\lg^* n = k$ (this is basically the definition of $\lg^*$)

We can also prove that ${}^k 2 >> 2^n$

if we set $k_0 = 9001$

The next smallest function asymptotically is going to be either $n \lg \lg n$ or $(\lg n)^{\lg^* n}$

The reasoning is fairly simple. Every other function on this list has some component that makes it at least polynomial time

$$\left\{\begin{array}{lll} \Theta(n^{3.5}) : [n^{4.5} - (n-1)^{4.5}], & \Theta(n \lg \lg n) : [n \lg \lg n], & \Theta(n^{1.5}) : [n\sqrt{\frac{n}{2}}], \\ \Theta(n^{2n}) : [n^{2n}], & \Theta(\lg(\lg^* n)) : [\lg(\lg^* n)], & \Theta(\lg^{(9001)} n) : [\lg^{(9001)} n)], \\ \Theta(1) : [\pi], & \Theta(e^{2n}) : [e^{2n}], & \\ \Theta(n^{1337}) : [n^{1337}], & \Theta(2^{\lg^* n}) : [2^{\lg^* n}], & \Theta((\lg n)^{\lg^* n}) : [(\lg n)^{\lg^* n}], \end{array}\right\}$$

$$\begin{array}{llllll} n^{4.5} - (n-1)^{4.5}, & n \lg \lg n, & n\sqrt{\frac{n}{2}}, & n^{2n}, & \lg(\lg^* n), \\ n^{(\lg \lg n)/(\lg n)}, & \lg^{(9001)} n, & \lg^* 2^n, & \lg(n!), & (\lg n)^{\lg n}, \\ \sum_{i=2}^{n} \frac{2}{i^2 - 1}, & e^{2n}, & n^{\lg \lg n}, & n^{1337}, & \lg^*(n/2), \\ (1 + \frac{1}{787898})^{787898n}, & n!, & \pi, & 2^{\lg^* n}, & (\lg n)^{\lg^* n} \end{array}$$

$\Theta(\lg^* 2^n)$

let's assume $n = {}^k 2 = 2^{2^{2^{\cdot^{\cdot^{\cdot^2}}}}}$ with k 2s in total on the tower of 2

then $\lg^* n = k$ (this is basically the definition of $\lg^*$)

As a result, if $\lg^* n$ = k, then

$\lg^* 2^n = \lg^* 2^{k2} = \lg^* 2^{2^{2^{2^{\cdot^{\cdot^{\cdot^2}}}}}}$ with (k + 1) 2s

which evaluates to just k+1, or $\lg^* n + 1$.

since for any value of n, $\lg^* 2^n$ is only $\lg^* n + 1$, we can say that $\lg^* 2^n = \Theta(\lg^* n)$

4 Question 4 Induction

a.

Theorem. *Some theorem here.*

Proof. Proof by induction

Base Step: When $n = 1$, is true.

Induction Hypothesis: Suppose is true

Induction Step: Consider when □

b. Time complexity: $\mathcal{O}(n)$

Proof. ($\Leftarrow$) Assume by Contradiction,

($\Rightarrow$) □

5 Question 5 Probability

a.

Theorem. *Some theorem here.*

Proof. Proof by induction

Base Step: When $n = 1$, is true.

Induction Hypothesis: Suppose is true

Induction Step: Consider when □

b. Time complexity: $\mathcal{O}(n)$

Proof. ($\Leftarrow$) Assume by Contradiction,

($\Rightarrow$) □

6 Question 6 Graphs, Proof by Contradiction

Proof. Assume by Contradiction,

Assume $d(u, w) + d(w, v) < d(u, v)$.

This implies that there is a path from u to v via w whose total distance is less than the direct distance between u and v.

However, by the definition of $d(u, v)$, it is the shortest distance between u and v. Therefore, no other path, including one through w, can have a smaller distance.

This contradiction arises from our assumption, so the assumption must be false. Hence, we conclude that $d(u, w) + d(w, v) \geq d(u, v)$. □

7 Question 7 Trees, Proof by induction

Definition: A *stable parent* is defined as a node with two leaves. Let:

- n be the number of nodes,
- l be the number of leaves,
- p be the number of stable parents.

Proof. Proof by induction.

Assume the inductive hypothesis:

$$l_n = p_n + 1 \quad \Rightarrow \quad l_{n+1} = p_{n+1} + 1$$

We will now consider two cases:

Case 1: Add a child to a leaf.

- Since the leaf gains a child, while a new leaf is added, we have $l_{n+1} = l_n$.
- Since no node with 1 child gains another child, $p_{n+1} = p_n$.
- Therefore, $l_{n+1} = l_n = p_n + 1 = p_{n+1} + 1$.

Case 2: Add a child to a node with 1 child already.

- Since no leaf stops being a leaf, you have simply added a leaf, so $l_{n+1} = l_n + 1$.
- Since the node with 1 child gains another child, a new stable parent is formed, so $p_{n+1} = p_n + 1$.
- Therefore, $l_{n+1} = l_n + 1 = p_{n+1} + 1 = p_n + 1 + 1$.

Base case: For $n = 1$, we have:

- 1 node implies 1 leaf and 0 stable parents.
- Thus, $l_1 = 1$ and $p_1 = 0$.
- Therefore, $1 = 0 + 1$, which satisfies $l_1 = p_1 + 1$.

Q.E.D.