CSCI570 Homework 1

Name: Saunak Sahoo **USC ID: 3896400217**

Ans 1

1. $2^{\log_e n} = 2^{\frac{\log_2 n}{\log_2 e}}$ using base change property
   $= n^{\frac{1}{\log_2 e}} = n^{\log_e 2}$
2. 2^{3n}
3. 3^{2n}
4. $n^{n \log(n)}$
5. $log(n)$
6. $nlog(n^2) = 2nlog(n)$
7. n^{n^2}
8. $log(n!) = log(n \times (n-1)... \times 2 \times 1) = log(1) + log(2) + ... + log(n-1) + log(n)$
9. $log(log(n^n)) = log(nlog(n)) = log(n) + log(log(n))$

The least growing functions here are logarithmic. We have 5, 9 and 8 of this type.
In this case we can clearly see that 9 has an additional factor of $log(log(n))$ in addition to the $log(n)$ from 5. So here we have 5<9.
8 is sum of logarithms of integers from 1 to n. Comparing 8 and 9, 8 has many more terms added and for value of n tending to infinity it will be greater. So now we have 5<9<8

Next the functions with a higher growth rate are of type nlog(n). Here we have 6 that is of this type. However, we also have The order of functions becomes 5<9<8<6

Next, the functions here that have a higher growth rate are polynomial functions. There is only one polynomial function here i.e. 1. So our order of functions becomes 5<9<8<6<1

Next we have exponential functions.
We have 2 and 3 as exponential functions of the form $const^x$. We know that $2^x < 3^x$. So here will get 2<3. So our order of functions becomes 5<9<8<1<2<3
We are now left with 4 and 7. While their base is common n, we know that $n^2 > n \log n$. Hence we would have 4<7

Finally, our order of functions are as follows:

$log(n) < log(log(n^n)) < log(n!) < nlog(n^2) < 2^{\log_e n} < 2^{3n} < 3^{2n} < n^{n \log(n)} < n^{n^2}$

Ans 2

Base case:

For k=1 we have $k^3 + 5k = 1^3 + 5X1 = 1 + 5 = 6$ which is divisible by 6

Inductive Hypothesis:

Assume that this condition holds for k=m.

So we are assuming that $m^3 + 5m$ is divisible by 6

Inductive Step:

We have to prove that the condition also holds true for k = m+1

So here we have to prove that $(m+1)^3 + 5(m+1)$ is also divisible by 6

Expanding this expression we get

$(m+1)^3 + 5(m+1)$

$= m^3 + 3m^2 + 3m + 1 + 5m + 5$

$= (m^3 + 5m) + 3m^2 + 3m + 6$

We have assumed that $m^3 + 5m$ is divisible by 6. If we can prove that the remaining terms $(3m^2 + 3m + 6)$ are divisible by 6, then the whole expression would be divisible by 6

To check for divisibility by 6, an expression must be divisible by both 2 and 3.

Divisibility by 2:

We have to check for the evenness of the expression here. Substituting m=2 we get

$3X2^2 + 3X2 + 6 = 12 + 6 + 6 = 24$ which is divisible by 2

Divisibility by 3:

$(3m^2 + 3m + 6) = 3(m^2 + m + 2)$ since 3 is a factor of this expression, it is divisible by 3 as well

Hence, we have proven that $(m^3 + 5m) + (3m^2 + 3m + 6) = (m+1)^3 + 5(m+1)$ is divisible by 6

Ans 3

Base case:

For n=1 we have $1^3 = \frac{1^2(1+1)^2}{4} = 1$

For n=2 we have $1^3 + 2^3 = \frac{2^2(2+1)^2}{4} = 9$

Inductive Hypothesis:

Assume that this condition holds true for n=m

So we are assuming that $1^3 + 2^3 + ... + m^3 = \frac{m^2(m+1)^2}{4}$

Inductive Step:

We have to prove that for this condition also holds true for n=m+1

So here we have to prove that $1^3 + 2^3 + ... + m^3 + (m+1)^3 = \frac{(m+1)^2(m+2)^2}{4}$

$\frac{(m^2+2m+1)(m^2+4m+4)}{4} = \frac{m^4+6m^3+13m^2+12m+4}{4}$

Now we have assumed that $1^3 + 2^3 + ... + m^3 = \frac{m^2(m+1)^2}{4}$

Substituting this in LHS we get:

$\frac{m^2(m+1)^2}{4} + (m+1)^3$

$= \frac{m^2(m^2+2m+1)}{4} + m^3 + 3m^2 + 3m + 1$

$= \frac{m^4+2m^3+m^2+4m^4+12m^2+12m+4}{4}$

$= \frac{m^4+6m^3+13m^2+12m+4}{4}$

= RHS

Hence proven $1^3 + 2^3 + ... + n^3 = \frac{n^2(n+1)^2}{4}$ for every positive integer n

Ans 4

Proof by contradiction

A prime number is a whole number greater than 1 which cannot be expressed as a product of 2 smaller natural numbers. In other words, prime numbers have only 1 and the number itself as its factors.

Let us assume that the Prime Filtering algorithm is incorrect.
So here, we are assuming that there exists an index iXj in the *isPrime()* array (where i≤n and j≤$\lfloor \frac{n}{i} \rfloor$) which is marked as true (i.e. it is a prime number). However, as we know from the definition of a prime number from the first statement, a prime number cannot be expressed as a product of 2 natural numbers. Hence, our assumption is wrong

So by proof of contradiction the Prime Filtering Algorithm is correct

Time complexity

To calculate the number of times the algorithm runs, let us consider the number of times it runs for each iteration of i

i=2 → $\frac{n}{2}$ times

i=3 → $\frac{n}{3}$ times

…

So the total number of iterations of the inner loop is

$= \frac{n}{2} + \frac{n}{3} + ... + \frac{n}{n}$

$= n[\frac{1}{2} + \frac{1}{3} + ... + \frac{1}{n}]$

$[\frac{1}{2} + \frac{1}{3} + ... + \frac{1}{n}]$ is a Harmonic Progression. Roughly, for n inputs → ∞ this converges to log(n)

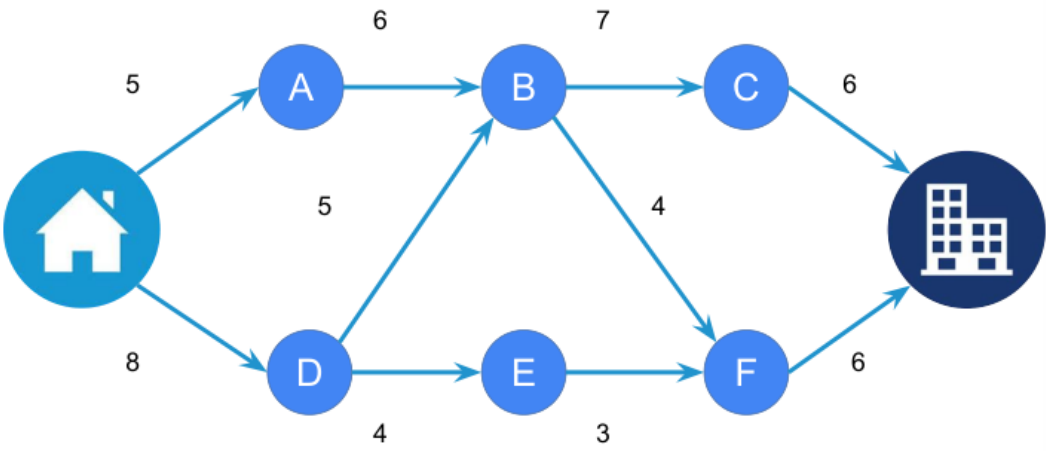
So our expression becomes

$= n[log(n)]$

= $O(nlog(n))$ is the time complexity

Ans 5

In order to find the shortest path, we need to perform topological sort first:



Step 1: Topological Sort

Here we chose vertex H as a vertex with no in-degree to start, and delete its outgoing edges

💡 Output = H

Next we can chose vertices A or D as both have 0 in-degree. Let us chose A and delete its outgoing edges

💡 Output = H A

Next we can chose D with no in-degree, and delete its outgoing edges

💡 Output = H A D

Next we can chose vertices B or E as both have 0 in-degree. Let us chose B and delete its outgoing edges

💡 Output = H A D B

Next we can chose E with no in-degree, and delete its outgoing edges

💡 Output = H A D B E

Next we can chose vertices C or F as both have 0 in-degree. Let us chose C and delete its outgoing edges

💡 Output = H A D B E C

Next we can chose F with no in-degree, and delete its outgoing edges

> 💡 Output = H A D B E C F

Finally the last node left is S

> 💡 Output = H A D B E C F S

Step 2: Shortest path using distance array and topological sort

Here we shall initialize a distance array from vertex H to all other vertices with an initial value of ∞

A	B	C	D	E	F	S
∞	∞	∞	∞	∞	∞	∞

We shall start with each vertex from the topological sort and update the distance array if:

Distance from vertex in topo sort to given edge < Distance given in distance array

We can see we have H→ A = 5 and H→ D = 8, which are both less than the value given in the distance array of A = ∞ and D = ∞. So we shall update the distance array

A	B	C	D	E	F	S
5	∞	∞	8	∞	∞	∞

Next in topological sort we have A. A(5 from distance array) →B (6 from graph) = 5 + 6 = 11. Since 11< ∞ (for B) we shall update distance array

A	B	C	D	E	F	S
5	11	∞	8	∞	∞	∞

Next in topological sort we have D.
D(8 from distance array) →E (4 from graph) = 12. Since 12< ∞ we shall update distance array
D(8 from distance array) →B (5 from graph) = 8 + 5 = 13. Since 11> 13 we shall keep the distance to B as 11

A	B	C	D	E	F	S
5	11	∞	8	12	∞	∞

Next in topological sort we have B.
B(11 from distance array) →C (7 from graph) = 18. Since 18< ∞ we shall update distance array
B(11 from distance array) →F (4 from graph) = 15. Since 18< ∞ we shall update distance array

A	B	C	D	E	F	S
5	11	18	8	12	15	∞

Next in topological sort we have E.
E(12 from distance array) →F (3 from graph) = 15. Since 15 = 15 we shall NOT update distance array

A	B	C	D	E	F	S
5	11	18	8	12	15	∞

Next in topological sort we have C.
C(18 from distance array) $\rightarrow$ S (6 from graph) = 24. Since 24< ∞ we shall update distance array

A	B	C	D	E	F	S
5	11	18	8	12	15	24

Next in topological sort we have F.
F(15 from distance array) $\rightarrow$ S (6 from graph) = 21. Since 21< 24 we shall update distance array

A	B	C	D	E	F	S
5	11	18	8	12	15	21

So from this distance array we can see that the minimum distance between H $\rightarrow$ S = 21

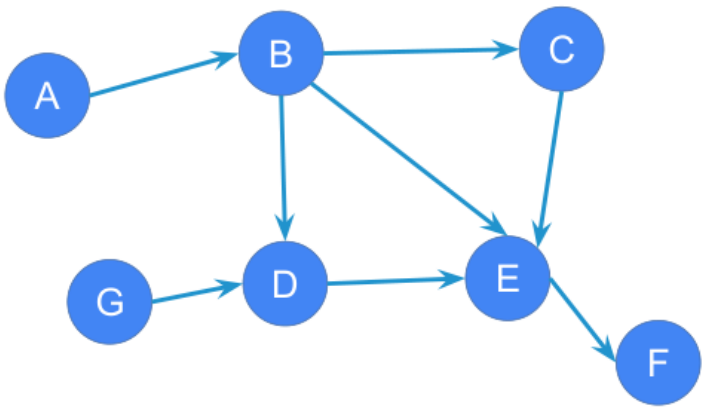
The possible shortest paths are as follows:

- $H \rightarrow A \rightarrow B \rightarrow F \rightarrow S$
- $H \rightarrow D \rightarrow E \rightarrow F \rightarrow S$

The other possible paths are as follows:

- $H \rightarrow A \rightarrow B \rightarrow C \rightarrow S$ Length = 5 + 6 + 7 + 6 = 24
- $H \rightarrow D \rightarrow B \rightarrow C \rightarrow S$ Length = 8 + 5 + 7 + 6 = 26
- $H \rightarrow D \rightarrow B \rightarrow F \rightarrow S$ Length = 8 + 5 + 4 + 6 = 23

Ans 6



Here we chose vertex A as a vertex with no in-degree to start, and delete its outgoing edges

💡 Output = A

Next we can chose vertices G or B as both have 0 in-degree. Let us chose G and delete its outgoing edges

💡 Output = A G

Next we can chose B with no in-degree, and delete its outgoing edges

💡 Output = A G B

Next we can chose vertices C or D as both have 0 in-degree. Let us chose D and delete its outgoing edges

💡 Output = A G B D

Next we can chose vertices C or E as both have 0 in-degree. Let us chose C and delete its outgoing edges

💡 Output = A G B D C

Next we can chose E with no in-degree, and delete its outgoing edges

💡 Output = A G B D C E

Finally the last node left is F

💡 Output = A G B D C E F

The other topological sorts following the same algorithm are as follows:

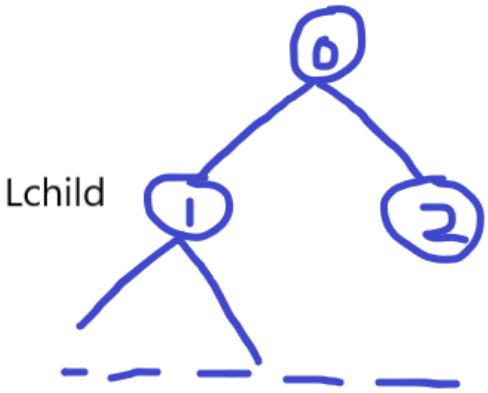
- A B C G D E F
- A B G D C E F
- A B G C D E F
- A G B C D E F
- G A B C D E F
- G A B D C E F

Ans 7

1)

Base case

For t=0 (i.e. the root node of a complete binary tree), it's left child is 2t+1 = $2X0 + 1 = 1$. This is true



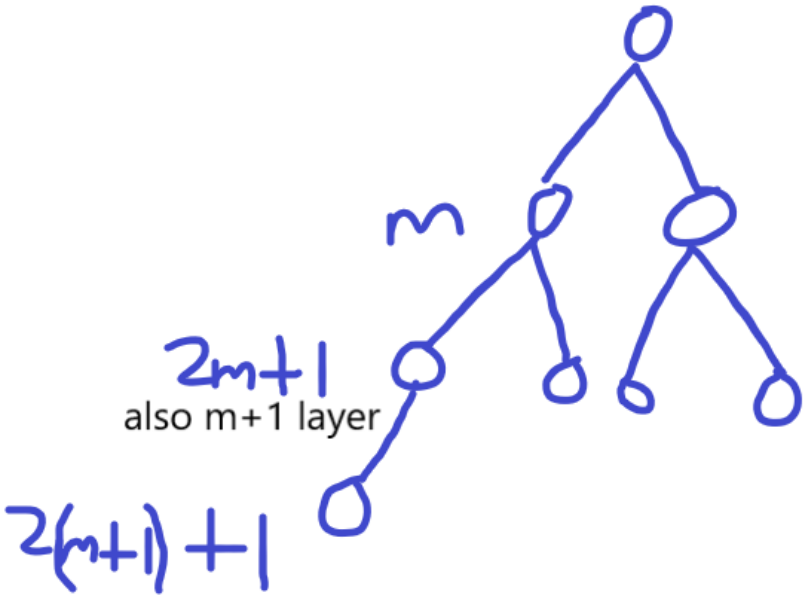
Inductive Hypothesis

Assume that this condition holds true for some positive integer m i.e. the leftmost node of a layer with label m has its left child labeled 2m + 1

Inductive Step

We need to prove that for a leftmost node with label m+1 has it's left child labeled 2(m+1) + 1

When we move from the leftmost node with label m to the leftmost node with label m+1:
The left child of the leftmost node with label m becomes the leftmost node with label m+1



And according to our inductive hypothesis, the left child would be 2(m+1) + 1

Hence proven

Inductive Hypothesis

For a layer k, we know from the previous question that that for a leftmost node t, it's left child would be 2t+1

Now the left child for the (m+1)th node of layer k with label t + m would be = 2(t+m) + 1

Inductive Step

Consider layer k+1, the leftmost node for this layer would have label 2t+1. Its left child would be (following Inductive Hypothesis) 2(2t+1) + 1 = 4t+3

For a complete binary tree, every parent node has 2 child nodes. If the parent nodes are p nodes apart, their left children would be 2p+1 nodes apart

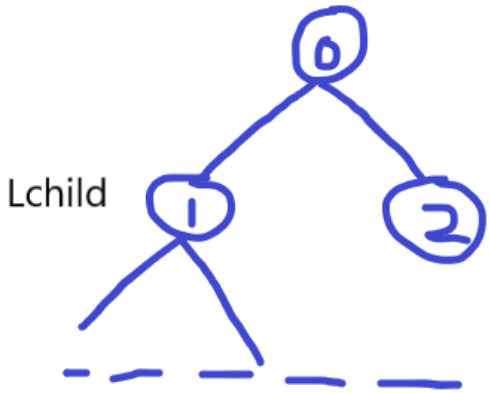
In the layer k+1, the leftmost node 2t+1 and 2(t+m)+1 is 2m+1 nodes apart. Thus we have in layer k+2, the distance between the leftmost node and the left child of 2t + 2m - 1 should be 2(2m-1) + 1 = 4m-1 apart, making the child node at 4t + 3 + 4m - 1 + 1 = 4t + 4m + 3 = 2(2t + 2m +1) + 1

Hence proven

2)

Base case

For t=0 (i.e. the root node of a complete binary tree), it's left child is 2t+1 = $2X0 + 1 = 1$. This is true



Ans 8

1) Time for *removeLastNodes()* = Time taken to get list of nodes with the largest distance + (Time taken to remove each node at largest distance X Worst case number of nodes at largest distance)

= O(1) + (O(1) X O(k))

= O(k)

Since in the worst case scenario, the number of nodes at the largest distance in a full binary tree will be O(k) for k nodes (i.e. no of leaves)

2) Time for *addTwoNodes()* = Time taken to get list of leaf nodes + (Time taken to add each nodeX 2 X Worst case number of leaf nodes)

= O(1) + (O(1) X 2 X O(k))

=2 X O(k)

=O(k)

Since in the worst case scenario, the number of leaf nodes in a full binary tree is O(k) for k nodes

Ans 9

For our i^{th} operation cost is as follows:

i=1	cost = $2^{log(i)-1} = 2^{-1} = \frac{1}{2}$ because $i = 1 = 2^j$ where j = 0
i=2	cost = $2^{log(i)-1} = 2^{-1} = \frac{1}{2}$ because $i = 1 = 2^j$ where j = 1
i=3	cost = 1 because $i = 1 = 2^j$ for any integer j

So we have a cost as a binary sequence for log(n) operations

= $\frac{1}{2} + 1 + 2 + 4 + ...log(n)times$

= $\frac{1}{2} + 2^{logn+1} - 1$ using the property of a binary sequence of n numbers = $2^{n+1} - 1$

= $\frac{1}{2} + 2X2^{logn} - 1$

=$-\frac{1}{2} + 2n$

= O(n)

The cost for the remaining n-log(n) operations is 1. So cost of these operations = 1 X (n - log(n)) = O(n)

Combining the whole cost we have

=O(n) + O(n)

=O(n)

Hence Amortized cost = total cost of operations / total number of operations

=$\frac{O(n)}{n} = O(1)$

Ans 10

In a singly linked list dictionary, we always insert at the end of the linked list and so the cost is O(1)

In the worst case scenario, we will have to traverse through all n inserted items to perform a lookup

Hence, total cost of all operations

= (Cost to insert one item X n items) + Cost to lookup once after n operations

=(O(1) X n) + O(n)

=O(n) + O(n)

=O(n)

Amortized cost = total cost of operations / total number of operations

$= \frac{O(n)}{n} = O(1)$