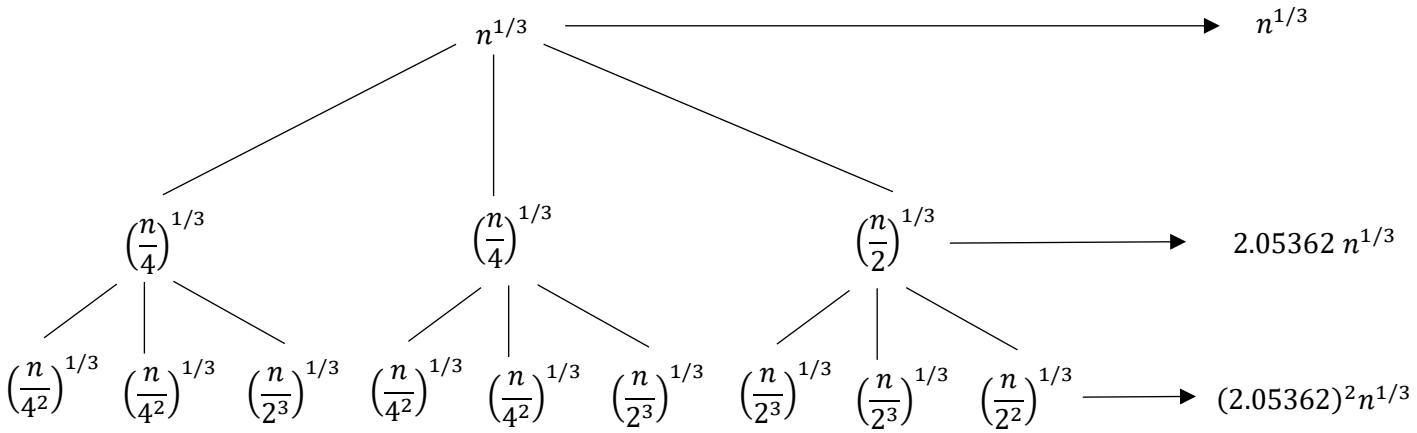
Shelby Williams **Algorithms Comprehensive Exam (Spring 2022)**

Short Questions:

1. Solve the two recurrence expressions below:
   a. $T_1(n) = 2T_1(n/4) + T_1(n/2) + n^{1/3}$



Longest Path: $\log_{3/2} n$

Shortest Path: $\log_3 n$

$(2.05362)^i n^{1/3}$

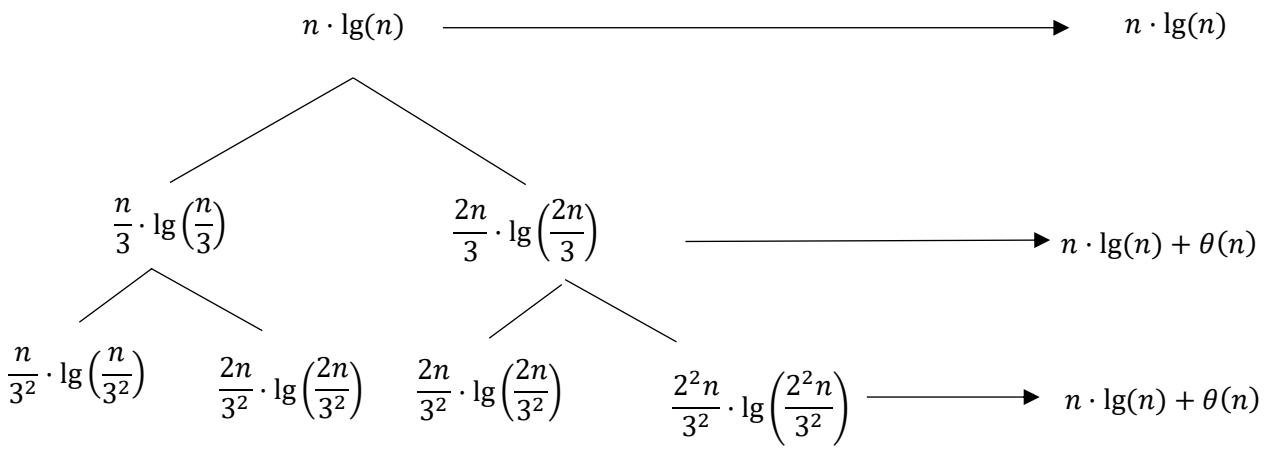
$$T_1(n) = \sum_{i=0}^{\log_{3/2} n} (2.05362)^i \cdot n^{1/3}$$

$$T_1(n) = n^{1/3} \cdot \sum_{i=0}^{\log_{3/2} n} (2.05362)^i$$

$$T_1(n) = n^{1/3} \cdot (2.05362)^{\log_{3/2} n} = n^{1/3} \cdot (n)^{\log_{3/2} 2.05362} = n^{1/3} \cdot n^{1.77476}$$

$$T_1(n) = n^{2.1081}$$

b. $T_2(n) = T_2(n/3) + T_2(2n/3) + n \cdot \lg(n)$

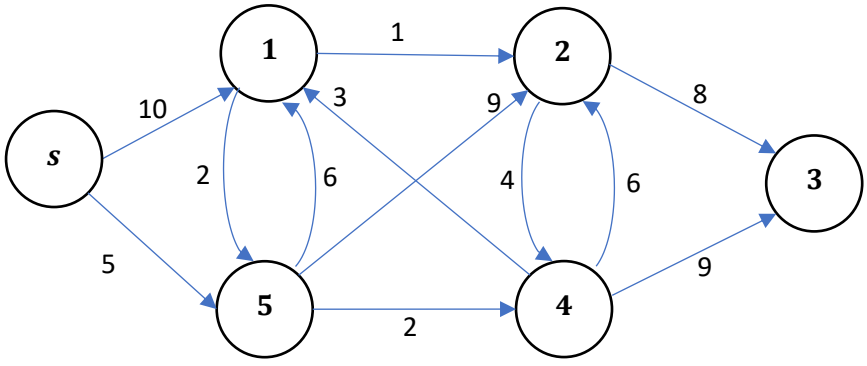


$$T_2(n) = \sum_{i=0}^{\log_{3/2} n} n \cdot \lg n + \theta(n)$$

$$T_2(n) = O(n \cdot \lg^2 n)$$

2. Follow depth-first search (DFS), starting from Node s, to traverse all nodes of the graph shown below. Mark (a) the type of every edge and (b) the discovery time and the finish times of each node.

   How do we utilize the DFS result to quickening the solution of single-source shortest paths in a direct acyclic graph?



DFS:

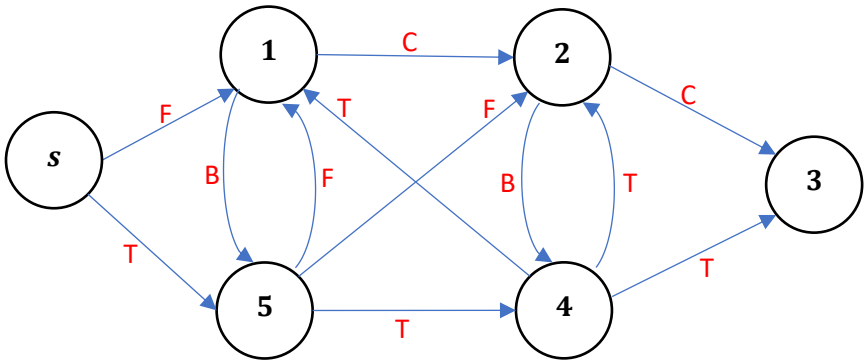
s: 1/, **5**: 2/ , **4**: 3/ , **3**: 4/ , **3**: 4/5, **2**: 6/ , **2**: 6/7, **1**: 8/9, **4**: 3/10, **5**: 2/11, **s**: 1/12

Tree Edges (T): (s,5), (5,4), (4,3), (4,2) , (4,1)

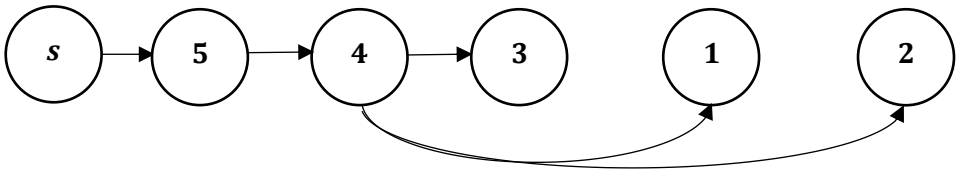
Back Edges (B): (2,4), (1,5)

Cross Edges (C): (2,3), (1,2)

Forward Edges (F): (5,1), (5,2), (s,1)



Sort nodes in non-decreasing finish time order.



3. Given that for an open-address hash table with load factor $\alpha = n/m < 1$, the expected number of probes in unsuccessful search under uniform hashing is at most $1/(1-\alpha)$, prove the expected number of probes in a successful probe under uniform hashing being at most $(1/\alpha)\cdot\ln(1-\alpha)^{-1}$ by giving a proof sketch which explains how many probes are needed to locate existing keys.

First unsuccessful probe: $1-\alpha$
Second unsuccessful probe: $\alpha(1-\alpha) = 1-\alpha^2$
Third unsuccessful probe: $\alpha^2(1-\alpha) = \alpha^3-\alpha^2$
There infinite sum is : $1 + \alpha + \alpha^2 + \alpha^3 + \ldots = 1/(1-\alpha)$

The number of successful inserts for the second key takes ≤ **1/(1-1/m)** probes**.**
The number of successful inserts for the third key takes ≤ **1/(1-2/m)** probes**.**
The number of successful inserts for the fourth key takes ≤ **1/(1-3/m)** probes**.**
The number of successful inserts for the $(i+1)^{th}$ key takes ≤ **1/(1-i/m).**
The expected number of successful inserts takes $\leq \frac{1}{n}\sum_{i=0}^{n-1}\frac{1}{1-\frac{i}{m}}$

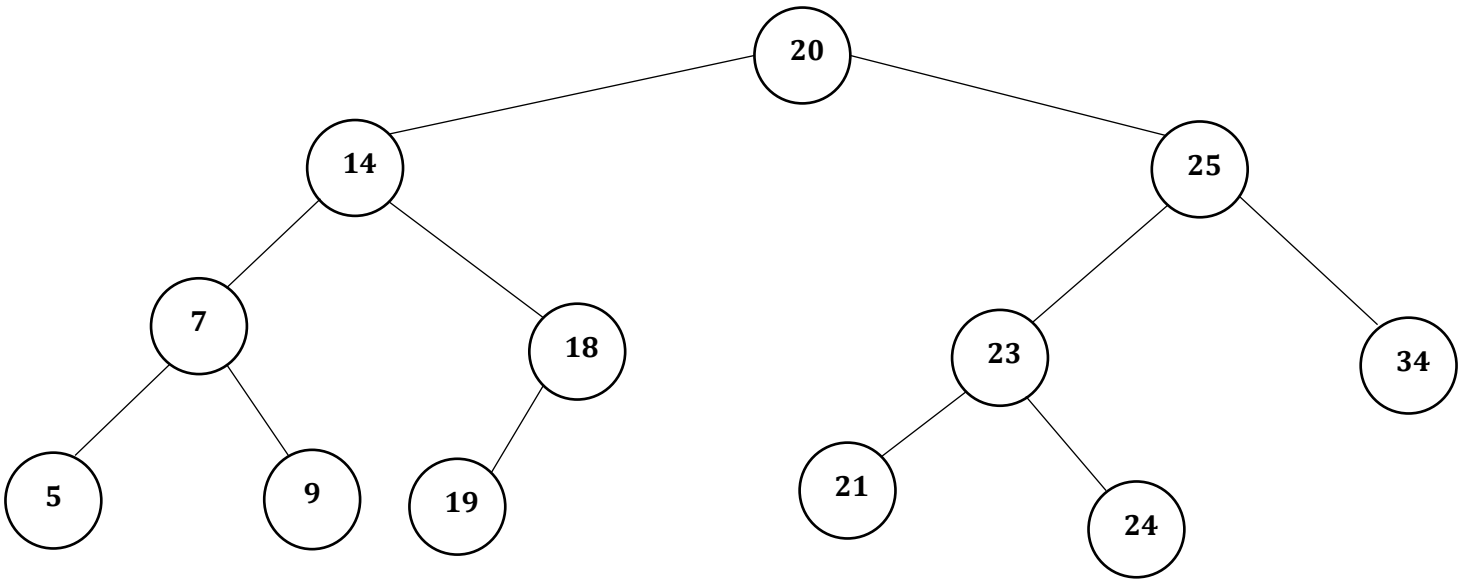
$$\frac{1}{n}\sum_{i=0}^{n-1}\frac{1}{1-\frac{i}{m}} = \frac{1}{m\alpha}\sum_{i=0}^{n-1}\frac{m}{m-i} = \frac{1}{\alpha}\sum_{i=0}^{n-1}\frac{1}{m-i} \leq \frac{1}{\alpha}\int_0^{n-1}\frac{1}{m-x}\,dx$$

$$\frac{1}{\alpha}[\ln(m-0) - \ln(m-n+1] = \frac{1}{\alpha}\ln\frac{m}{m-n+1} =$$

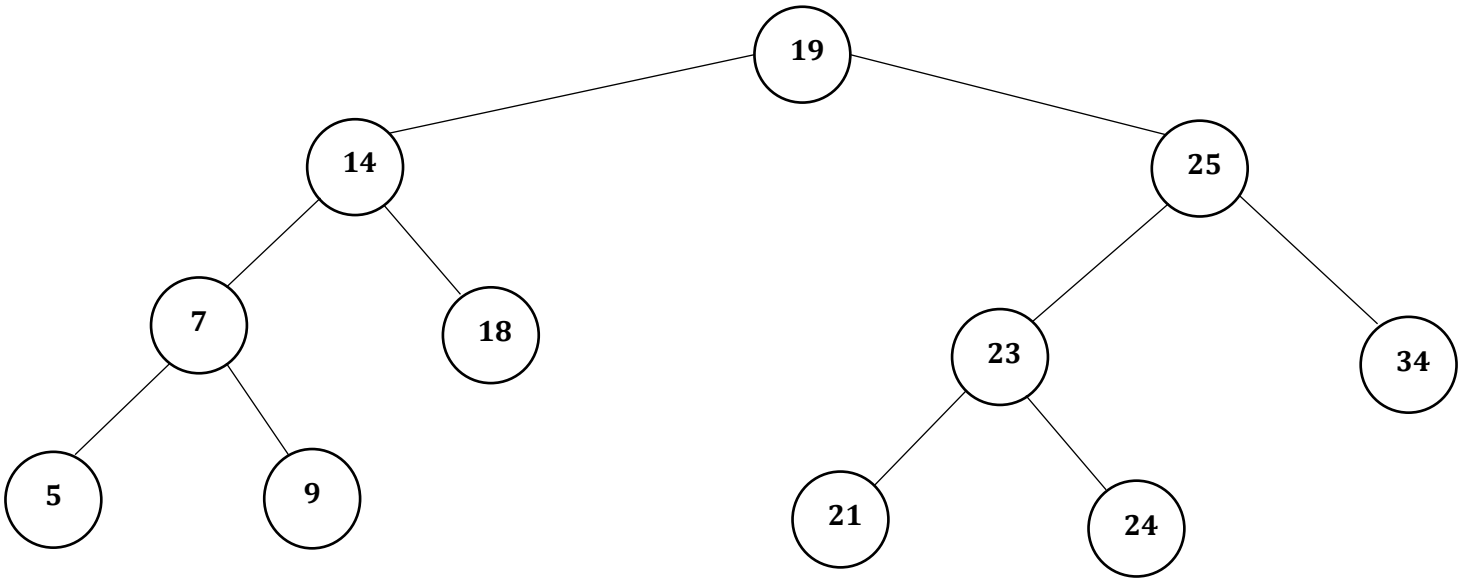
Let m-n >> 1

$$\frac{1}{\alpha}\ln\frac{m}{m-n} = \frac{1}{\alpha}\ln\frac{1}{1-n/m} = \frac{1}{\alpha}\ln\frac{1}{1-\alpha}$$

4. A binary search tree (T) is to be maintained following the in-order tree traversal order. Consider a sequence of arrival keys, {25, 23, 14, 7, 9, 21, 34, 18, 24, 19, 5}, to T which has just the root node with tis key = 20 initially.
    a. Show the resulting T after inserting all arrival keys.



    b. Show the resulting T after its root node is then deleted.



5. How many ones does the following code print when running with input n? Computer the exact value, if possible; otherwise provide a big-O bound.
    a. Ones(n):
    If n <= 0
        { print 1}
    else
        { Ones(n-1)
          Ones(n-2) }

n	Number of 1's printed
0	1
1	2
2	3
3	5
4	8

T(n) = T(n-1) + T(n-2)

T(n) = O (2^n)

    b. Ones(n):
    If n = 0
        { print 1}
    else
        for i = 1 to n
          { Ones(n-1) }

n	Number of 1's printed
0	1
1	1
2	2
3	6
4	24
5	120

T(n) = n!

T(n) = O (n!)

c. Ones(n):

```
If n = 0
      { print 1}
else
     for i = 1 to 2^n
       { Ones(n-1) }
```

n	Number of 1's printed
0	1
1	1
2	21
3	135
4	9865

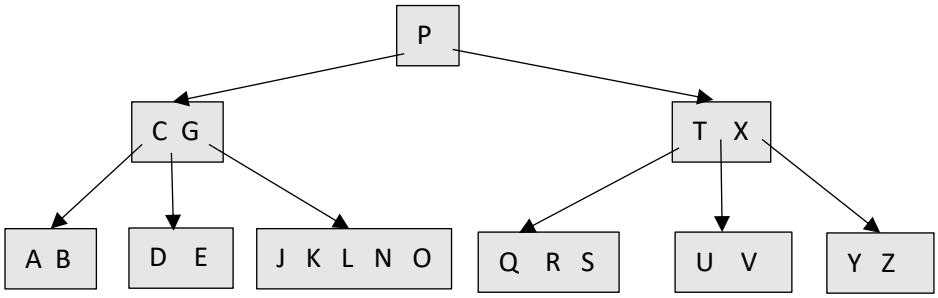
$T(n) = n! \cdot 2^n$

$T(n) = O(n! \cdot 2^n)$

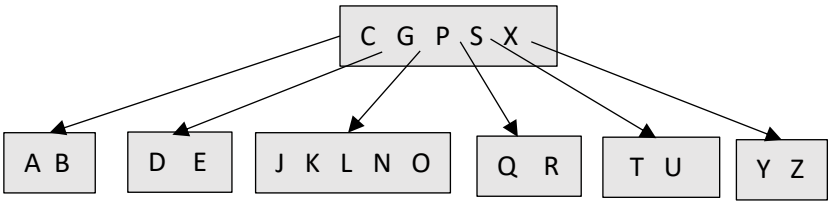
6. I skipped this one entirely.

Long Questions:

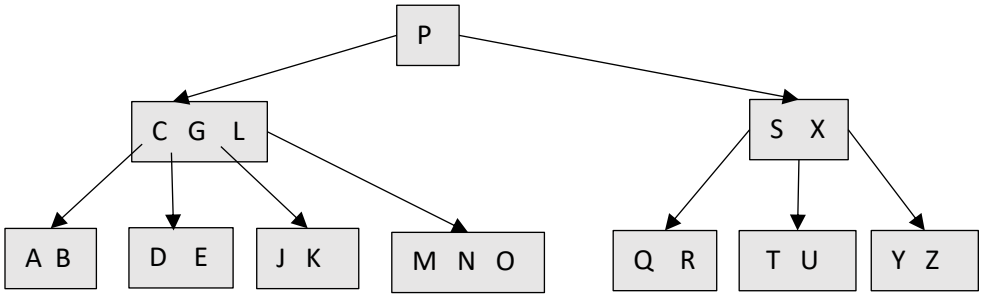
1. Given a B-tree with the minimum degree of t=3 below, show the results after (a) deleting V, (b) then followed by inserting M, (c) then followed by deleting B, and (d) then followed by deleting S.



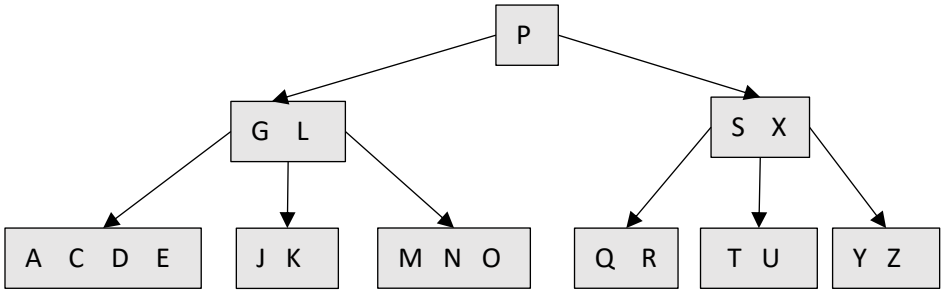
a. Deleting V…



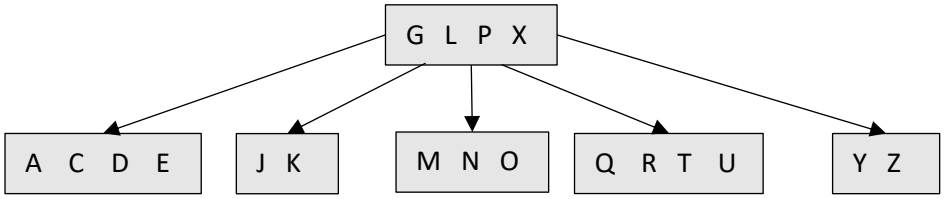
b. Inserting M…



c. Deleting B…

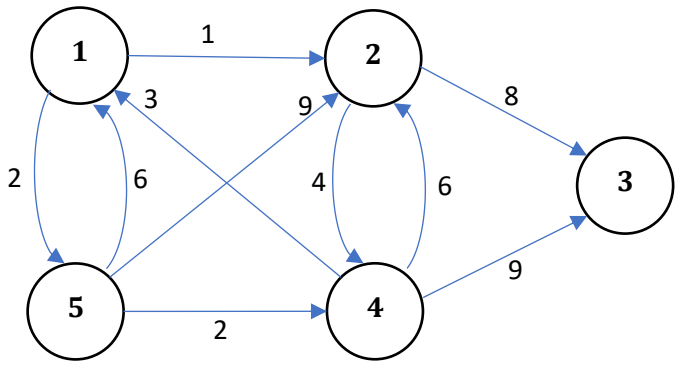


d. Deleting S…



2. The Floyd-Warshall algorithm (FW) obtains all pairs of shortest paths in a weighted directed graph. Consider the graph given in Problem SHORT-2 above, with Vertice s ignored. What is the recursive equation of $d_{i,j}^{(k)}$ for the shortest-path between i and j with intermediate vertices ∈ {1, 2, 3,…, k}?

$$d_{i,j}^{(k)} = min\left(d_{i,j}^{(k-1)}, d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)}\right)$$



$D^{(0)}$	0	1	∞	∞	2
	∞	0	8	4	∞
	∞	∞	0	∞	∞
	3	6	9	0	∞
	6	9	∞	2	0

$D^{(1)}$	0	1	∞	∞	2
	∞	0	8	4	∞
	∞	∞	0	∞	∞
	3	4	9	0	5
	6	7	∞	2	0

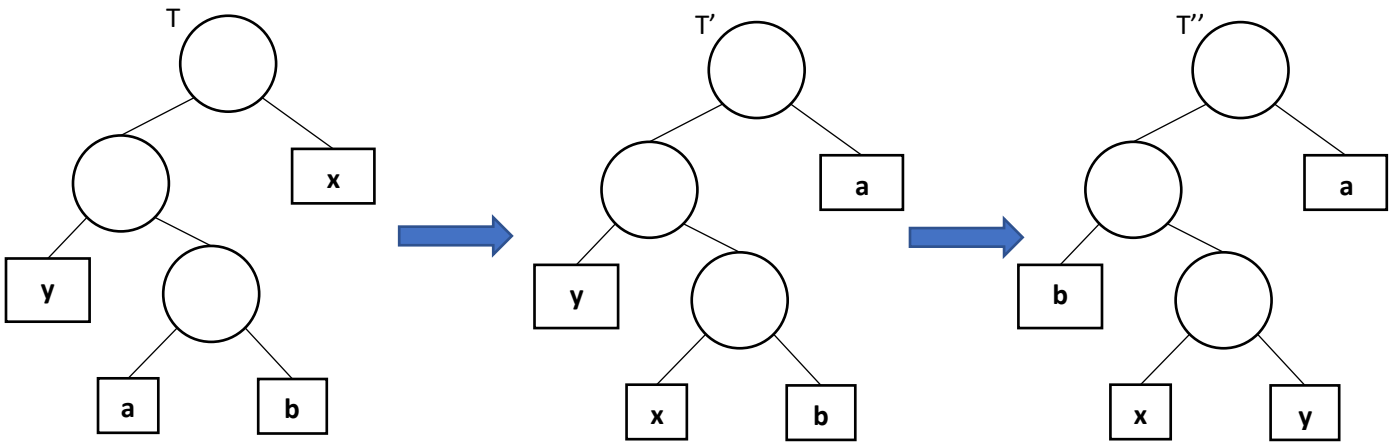
$D^{(2)}$	0	1	9	5	2
	∞	0	8	4	∞
	∞	∞	0	∞	∞
	3	4	9	0	5
	6	7	15	2	0

$D^{(3)}$	0	1	9	5	2
	∞	0	8	4	∞
	∞	∞	0	∞	∞
	3	4	9	0	5
	6	7	15	2	0

$D^{(4)}$	0	1	9	5	2
	7	0	8	4	9
	∞	∞	0	∞	∞
	3	4	9	0	5
	5	6	11	2	0

$D^{(5)}$	0	1	9	4	2
	7	0	8	4	9
	∞	∞	0	∞	∞
	3	4	9	0	5
	5	6	11	2	0

3. Sketch a proof of the Lemma below, using the tree provided.
   Let C be an alphabet in which each character c ∈ C has frequency c.freq. Let x and y be two characters in C having the lowest frequencies. Then there exists an optimal prefix code for C in which the codewords for x and y have the same length and differ only in the last bit.



T denotes an arbitrary optimal prefix code.

$$B(T) - B(T') = \sum_{c \in C} c.freq \cdot d_T(c) - \sum_{c \in C} c.freq \cdot d_{T'}(c)$$

$$B(T) - B(T') = x.freq \cdot d_T(x) + a.freq \cdot d_T(a) - x.freq \cdot d_{T'}(x) + a.freq \cdot d_{T'}(a)$$

$$B(T) - B(T') = x.freq \cdot d_T(x) + a.freq \cdot d_T(a) - x.freq \cdot d_T(a) + a.freq \cdot d_T(x)$$

$$B(T) - B(T') = (a.freq - x.freq)\big(d_T(a) - d_T(x)\big)$$

$$B(T) - B(T') \geq 0, therefore, B(T) \geq B(T')$$

$$But, B(T') \geq B(T), since\ T\ is\ optimal, so\ we\ have\ B(T') \equiv B(T)$$

Similarly, $B(T') \geq B(T'')\ to\ yield\ B(T) \geq B(T'), since\ B(T') \equiv B(T)$.

Given that T is optimal, we have B(T'') ≡ B(T), implying that T"" is another optimal prefix code with x and y differing only in the last bit.

4. A rabbit is running up a staircase with 10 steps and can hop 1 step or 2 steps at a time. How many possible ways the rabbit can hop up the stairs? Please give the exact number.

   If the staircase has n steps, please show your algorithm to compute all the possible ways the rabbit can hop up the stairs.

   (Hint: apply dynamic programming.)

   hops(n) = hops(n-1) + hops(n-2), where hops(1) = 1 and hops(2) = 2

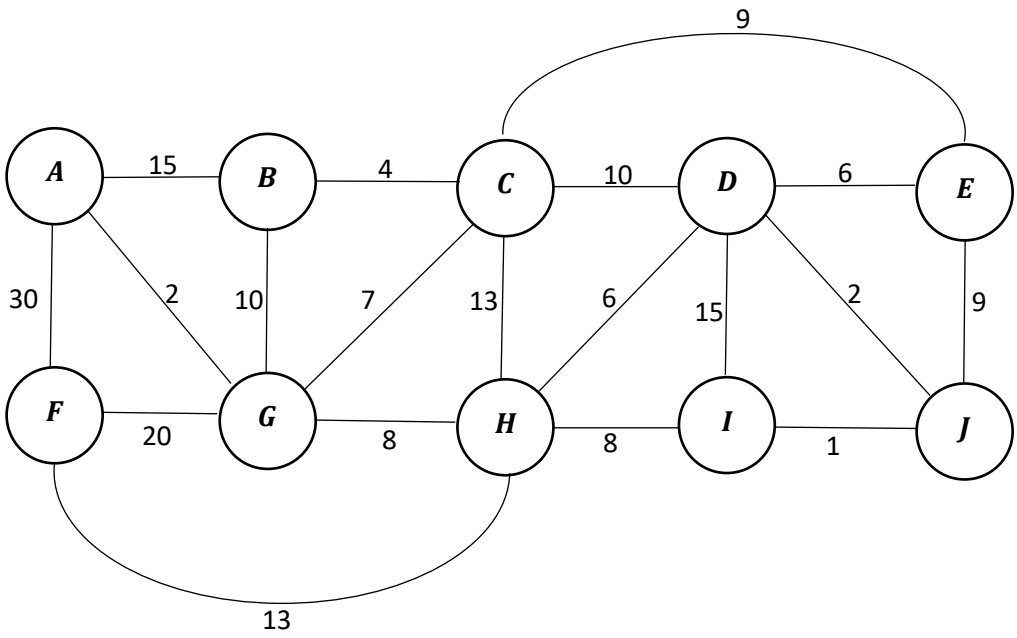
n	hops
1	1
2	2
3	3
4	5
5	8
6	13
7	21
8	34
9	55
10	**89**

Recursive Program:

```
def hops(n):
    if n==1:
        return(1)
    elif n==2:
        return(2)
    else
        return hops(n-1) + hops(n-2)
```

hops(10)=89

5. Given the un-directed graph below, run Dijkstra's algorithm, starting at Vertex A Note that the algorithm works in the same way on an un-directed graph as on a directed graph.



Show each step by filling out the table. The second column denotes the set S, which refers to the nodes whose shortest distances from A have been determined. The third to twelfth columns show the shortest distances from A to other vertices. Add rows when needed.

	Set S	A	B	C	D	E	F	G	H	I	J
Initialization	{ }	0	∞	∞	∞	∞	∞	∞	∞	∞	∞
1st iteration	{A}	0	15	∞	∞	∞	30	2	∞	∞	∞
2nd	{AG}	0	12	9	∞	∞	22	2	10	∞	∞
3rd	{AGC}	0	12	9	19	18	22	2	10	∞	∞
4th	{AGCH}	0	12	9	16	18	22	2	10	18	18
5th	{AGCHB}	0	12	9	16	18	22	2	10	18	18
6th	{ AGCHBD}	0	12	9	16	18	22	2	10	18	18
7th	{AGCHBDI}	0	12	9	16	18	22	2	10	18	18
8th	{AGCHBDIE}	0	12	9	16	18	22	2	10	18	18
9th	{AGCHBDIEJ}	**0**	**12**	**9**	**16**	**18**	**22**	**2**	**10**	**18**	**18**