Algorithms – Comprehensive Exam (Fall 2020)

Short Questions.

1. Solve the following recurrences to get their runtimes upper bounds:
   a. $T(n) = T(n-2) + n$

$$height = n/2$$

$$T(n) = \sum_{i=0}^{n/2} (n - 2i) = O(n^2)$$

$$T(n) = O(n^2)$$

$$n$$
$$n-2$$
$$n-4$$
$$n-6$$
$$n-8$$
$$\cdot$$
$$\cdot$$
$$\cdot$$
$$n-2i$$

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

Master Theorem: a = 1, b = 2

$$Since\ n = \Omega(n^{\log_2 1})$$

$$T(n) = O(n)$$

$$n$$
$$n/2$$
$$n/4$$
$$n/8$$
$$n/16$$
$$\cdot$$
$$\cdot$$
$$\cdot$$
$$n/2^i$$

   c. $T(n) = T(n-2) + T\left(\frac{n}{2}\right) + n$

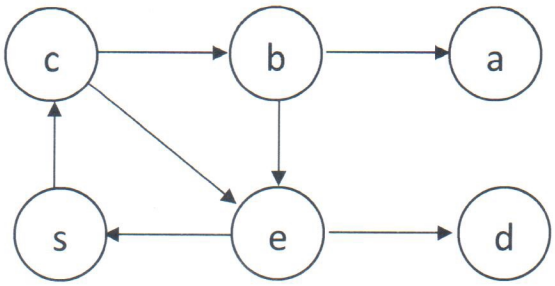
The longest branch: height = n/2

The shortest branch: height = lg n

$$T(n) = \sum_{i=0}^{n/2} (n - 2i) = O(n^2)$$

$$T(n) = O(n^2)$$

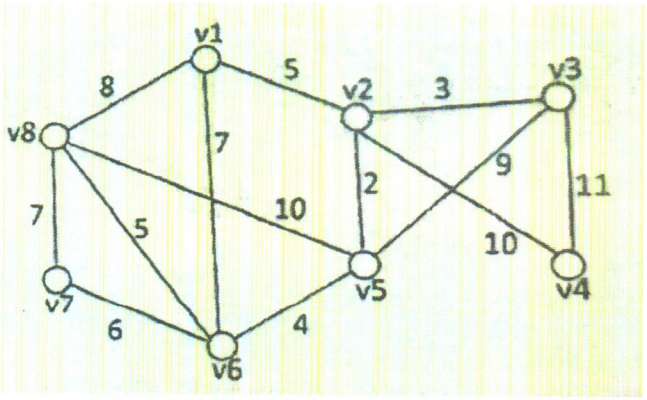
2. Give the visited vertex order of the following directed graph, starting with vertex s, under (1) breath-first search and (2) depth-first search, respectively.



BFS: s, c, e, b, d, a

DFS: s, c, b, a, e, d

3. Follow Prim's Algorithm and Kruskal's Algorithm, respectively to derive the minimum spanning tree (MST) for the following graph. For every algorithm, write down the edge picked in each step. For example, Step 1: (v1, v2).



Prim's Algorithm: Select the minimum edge of the entire graph and then select the next minimum edge connected to the edge already chosen, repeatedly.

Step	Edge	Edge Length
1	(v2, v5)	2
2	(v2, v3)	3
3	(v5, v6)	4
4	(v6, v8)	5
5	(v2, v1)	5
6	(v6, v7)	6
7	(v2, v4)	10

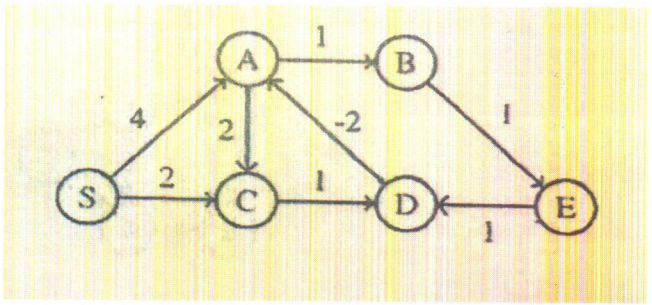
The minimum spanning tree cost is 35.

Kruskal's Algorithm: Always select a minimum cost edge, but not when a cycle is formed.

Step	Edge	Edge Length
1	(v2, v5)	2
2	(v2, v3)	3
3	(v5, v6)	4
4	(v6, v8)	5
5	(v2, v1)	5
6	(v6, v7)	6
7	(v2, v4)	10

The minimum spanning tree cost is 35.

4. Please use Johnson's Algorithm to reweight the directed graph shown below such that all edge weights are non-negative.



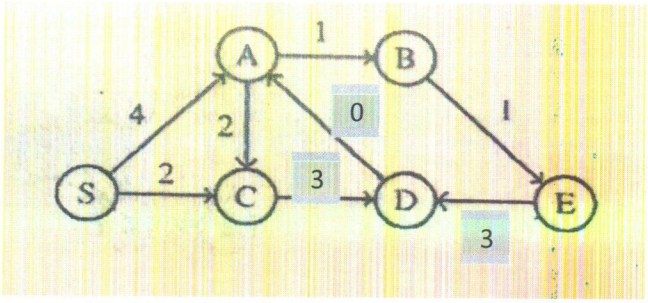
Step 1:
1. Take a source vertex outside.
2. The graph to all vertices 0 distance.

Step 2: Apply Bellman-Ford Algorithm

1. $w'(u,v) = w(u,v) + h(u) - h(v)$

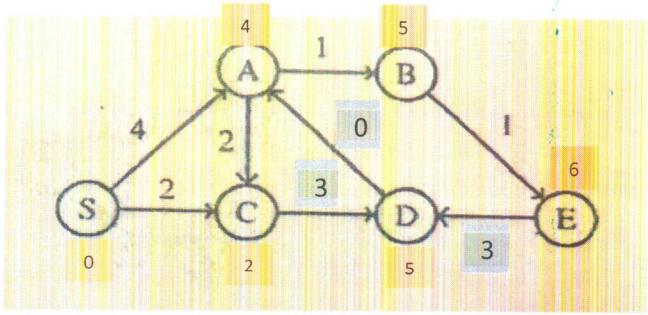
$h(S) = 0$
$h(A) = 0$
$h(B) = 0$
$h(C) = 0$
$h(D) = -2$
$h(E) = 0$

$w'(S,A) = w(S,A) + h(S) - h(A) = 4 + 0 - 0 = 4$
$w'(S,C) = w(S,C) + h(S) - h(C) = 2 + 0 - 0 = 2$
$w'(A,B) = w(A,B) + h(A) - h(B) = 1 + 0 - 0 = 1$
$w'(A,D) = w(A,D) + h(A) - h(D) = -2 + 0 - (-2) = 0$
$w'(A,C) = w(A,C) + h(A) - h(C) = 2 + 0 - 0 = 2$
$w'(B,E) = w(B,E) + h(B) - h(E) = 1 + 0 - 0 = 1$
$w'(E,D) = w(E,D) + h(E) - h(D) = 1 + 0 - (-2) = 3$
$w'(C,D) = w(C,D) + h(C) - h(D) = 1 + 0 - (-2) = 3$



Step 3: Apply Dijkstra's Algorithm

S=0; A=∞, B=∞, C=∞, D=∞, E=∞
Apply relaxation on each vertex
Vertex S: A=4, C=2; *Vertex C*: D=5;
Vertex A: B=5; *Vertex B*: E=6



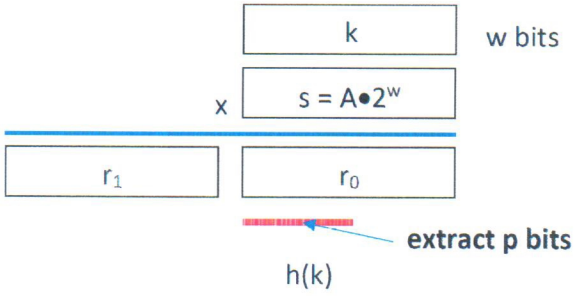
5. The utilization efficiency of a hash table depends heavily on its hashing function(s) employed. Describe with a diagram to illustrate how a multiplication method of hashing works on a machine with the word size of 2 bits for a hash table with 2^p entries, $p < w$.

A should be chosen as the golden ratio number: $A = \frac{\sqrt{5}+1}{2} = \frac{\sqrt{5}-1}{2}$

Multiplication Method:

Multiply k by a constant A, with $0 < A < 1$, and extract the fraction part of kA.

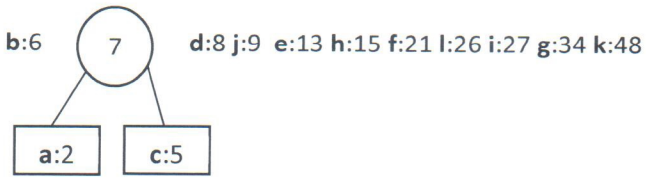
$$h(k) = \lfloor m(kA \bmod 1) \rfloor$$

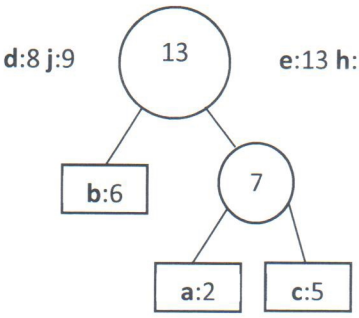


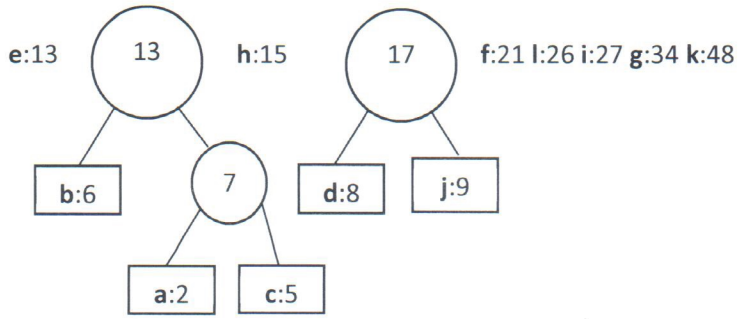
6. How your construction of an optimal Huffman code for the set of 12 frequencies: **a**:2 **b**:6 **c**:5 **d**:8 **e**:13 **f**:21 **g**:34 **h**:15 **i**:27 **j**:9 **k**:48 **l**:26

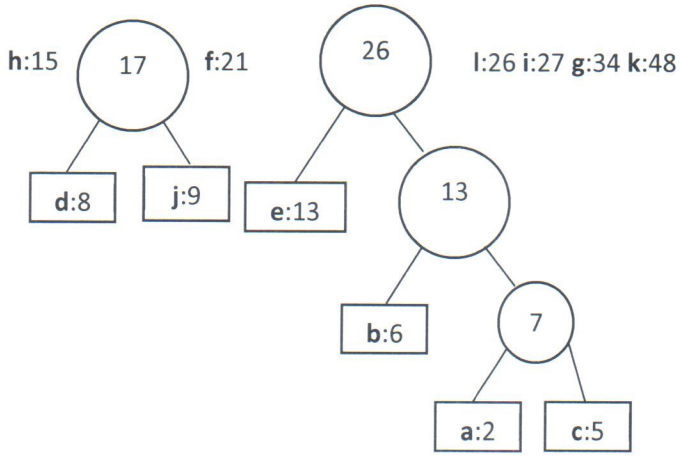
Sort in non-decreasing order:

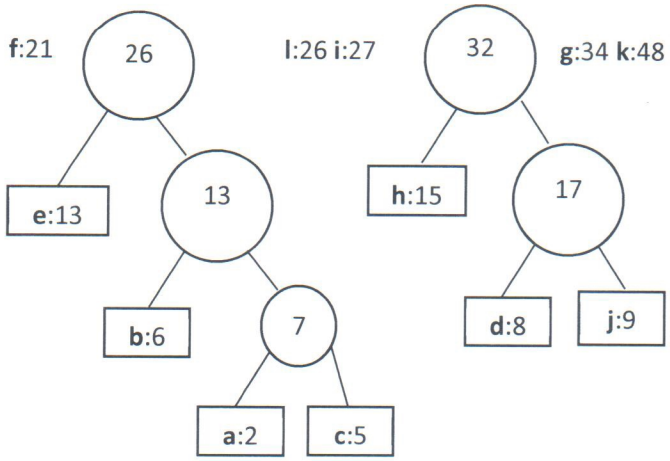
a:2 **c**:5 **b**:6 **d**:8 **j**:9 **e**:13 **h**:15 **f**:21 **l**:26 **i**:27 **g**:34 **k**:48

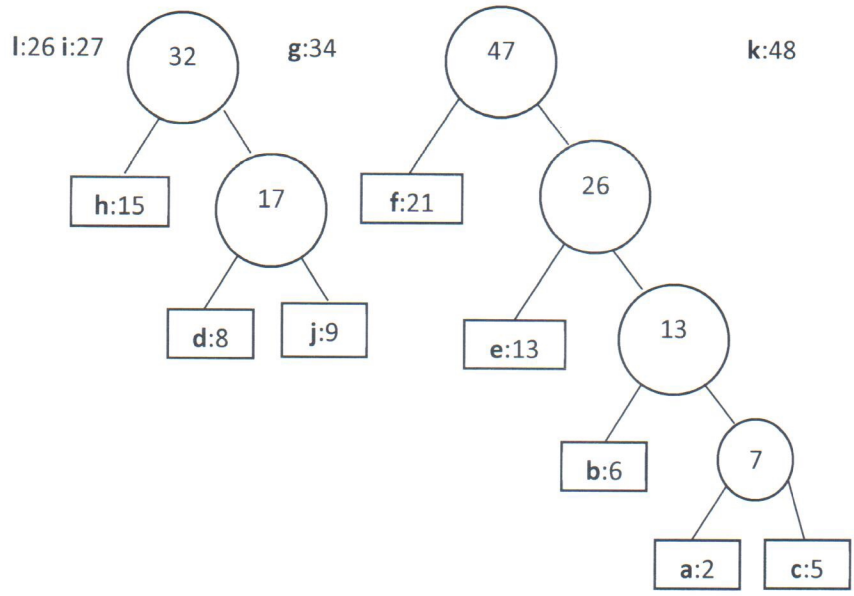


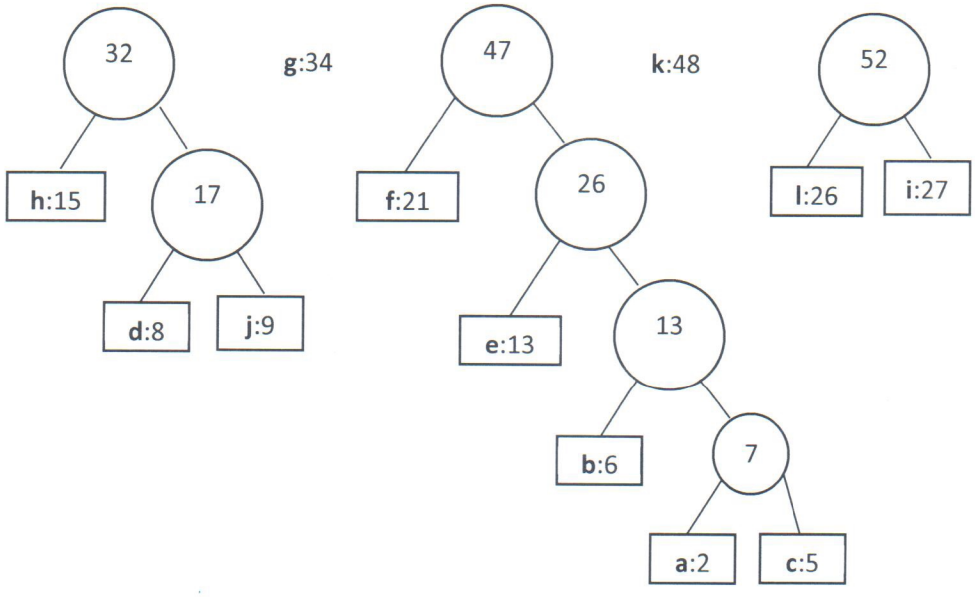


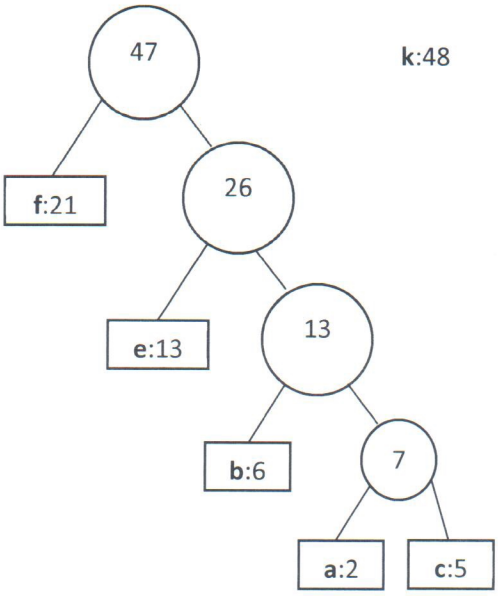

e:13
13
h:15
17
f:21 l:26 i:27 g:34 k:48
b:6
7
d:8
j:9
a:2
c:5


h:15
17
f:21
26
l:26 i:27 g:34 k:48
d:8
j:9
e:13
13
b:6
7
a:2
c:5

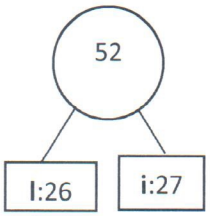

f:21
26
l:26 i:27
32
g:34 k:48
e:13
13
h:15
17
b:6
7
d:8
j:9
a:2
c:5

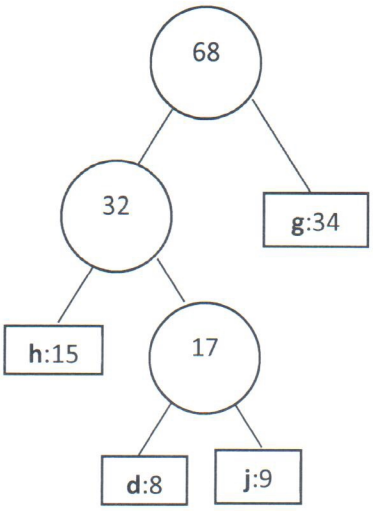

l:26 i:27
32
g:34
47
k:48
h:15
17
f:21
26
d:8
j:9
e:13
13
b:6
7
a:2
c:5

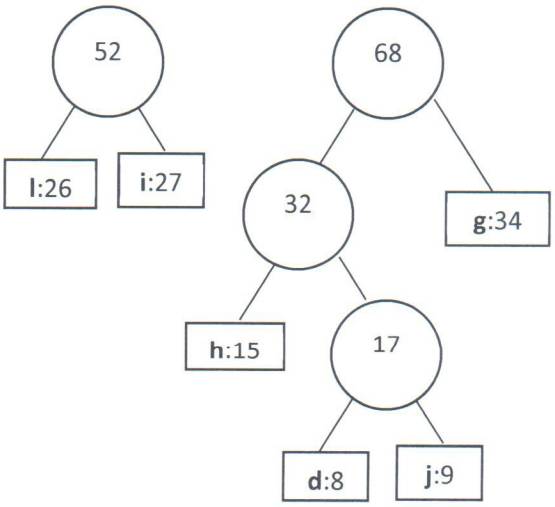

32
g:34
47
k:48
52
h:15
17
f:21
26
l:26
i:27
d:8
j:9
e:13
13
b:6
7
a:2
c:5

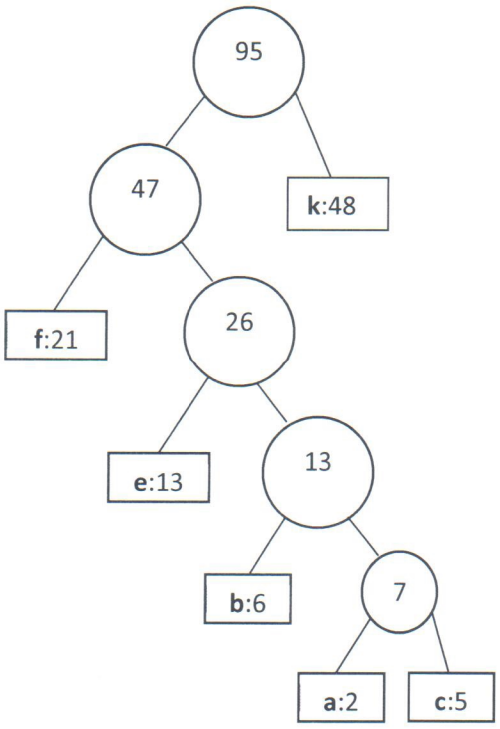

47
26
f:21
e:13
13
b:6
7
a:2
c:5

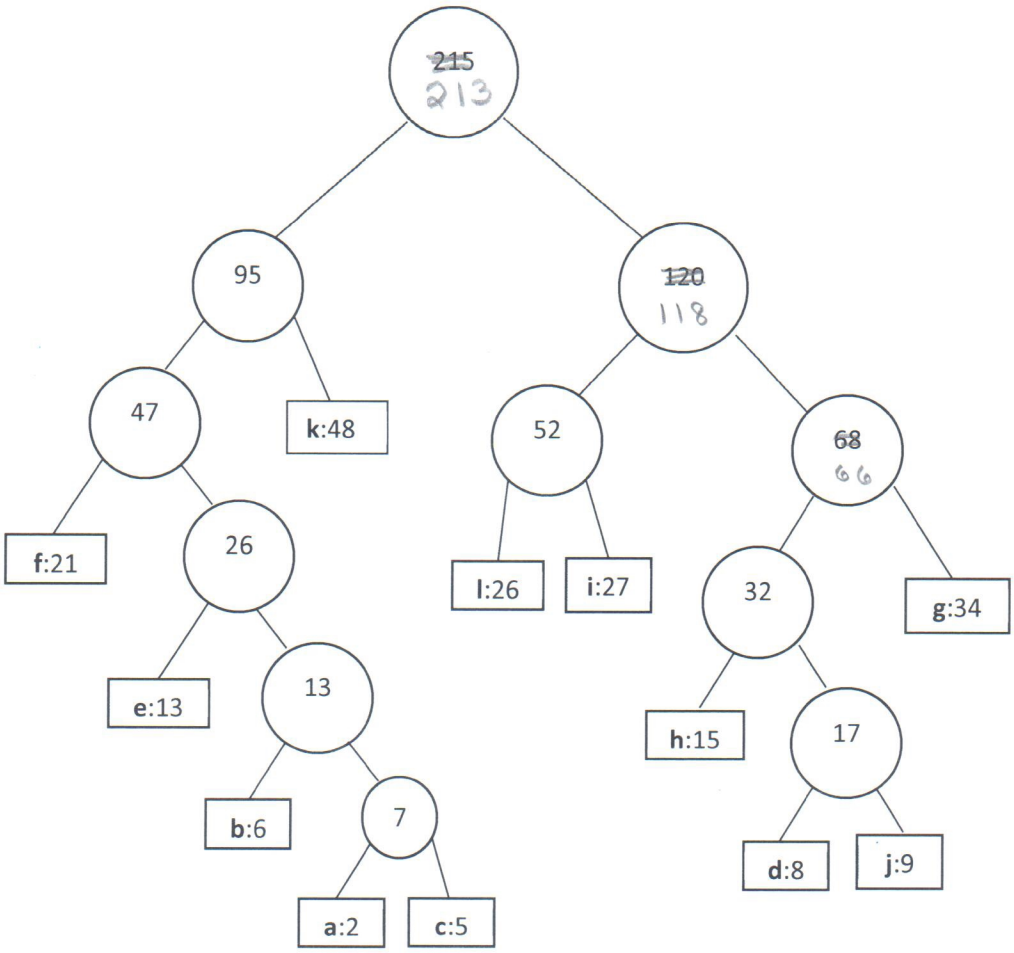
k:48


52
l:26
i:27


68
32
g:34
h:15
17
d:8
j:9

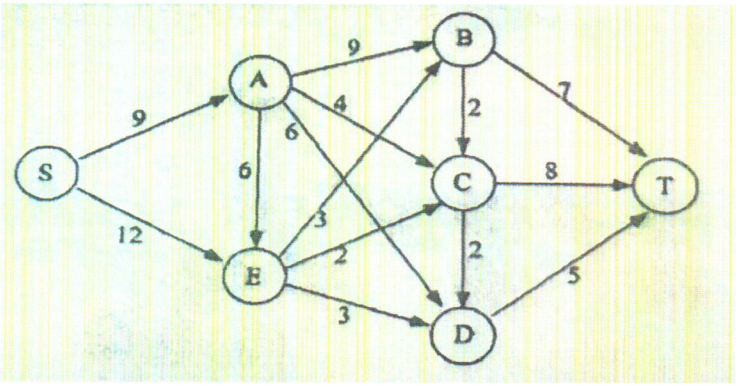

52
l:26
i:27
68
32
g:34
h:15
17
d:8
j:9


95
47
k:48
f:21
26
e:13
13
b:6
7
a:2
c:5


~~215~~
213
95
~~120~~
118
47
k:48
52
~~68~~
66
f:21
26
l:26
i:27
32
g:34
e:13
13
h:15
17
b:6
7
d:8
j:9
a:2
c:5

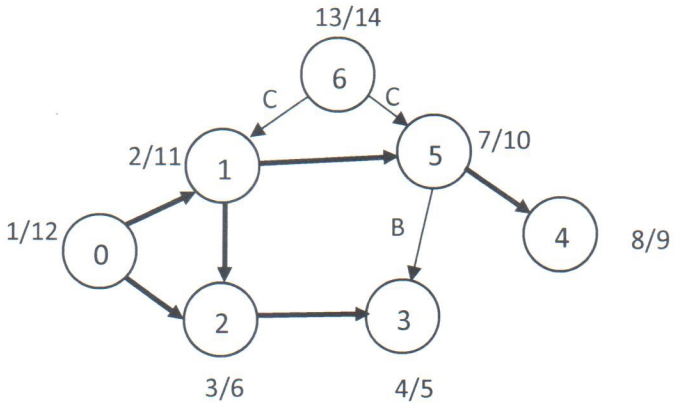
Long Questions

1. Follow the Ford-Fulkerson Algorithm to compute the max flow of the flow network given below. Illustrate the corresponding residual network at each step of the Ford-Fulkerson Algorithm. Also, compute the min-cut of the flow network, with each step shown.

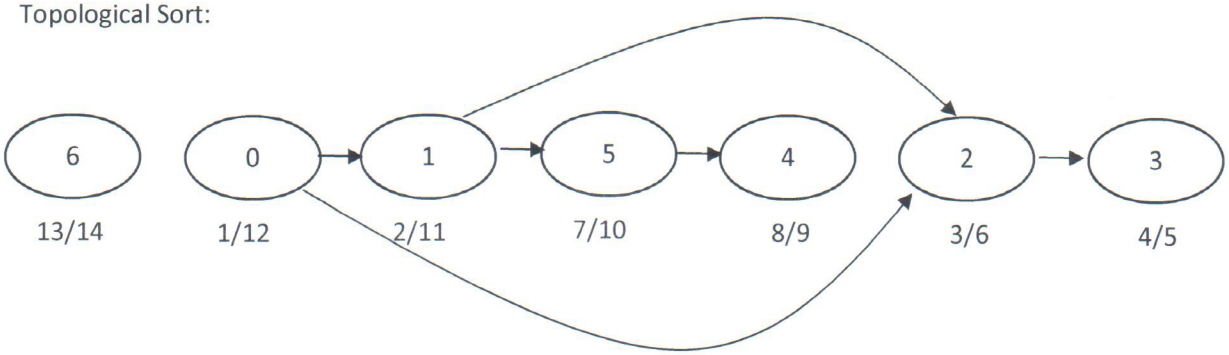


Source to Sink Path (Added Flow)	**Maximum Flow**	**Reached Max. Capacity**
S -> A -> C -> T (4)	4	* A -> C (4/4)
S -> E -> C -> T (2)	6	* E -> C (2/2)
S -> A -> B ->C -> T (2)	8	* C -> T (8/8)
S -> A -> B -> T (3)	11	* S -> A (9/9)
S -> E -> D -> T (3)	14	* E -> D (3/3)
S -> E -> B -> T (3)	17	

2. The following graph denotes a sequence of courses, with their dependencies on other courses shown. A directed edge from vertex u in the graph to vertex v indicates that course u must be taken before course v may start. Please apply DFS (hint: topological sort) to find an ordering of these courses that conforms to the shown dependencies. Illustrate each step.



Topological Sort:



3. An optimal binary search tree (OBST) for a given set of keys with known access probabilities ensures the minimum expected search cost for key accesses. Given the set of four keys with their access probabilities of $k_1 = 0.16$, $k_2 = 0.13$, $k_3 = 0.2$, $k_4 = 0.08$, respectively, and five non-existing probabilities of $d_0 = 0.12$, $d_1 = 0.1$, $d_2 = 0.06$, $d_3 = 0.11$, $d_4 = 0.04$ construct OBST following dynamic programming with memorization for the given four keys and demonstrate the constructed OBST, which contains all four keys (k_1, k_2, k_3, k_4) and five non-existing dummies (d_0, d_1, d_2, d_3, d_4). (Show your work using the three tables, for expected costs: e[i,j], access weights: w[i,j], and root[i,j] with i in e[i,j] and w[i,j] ranging from 1 to 5, j in e[i,j] and w[i,j] ranging from 0 to 4, and both i and j in root[i,j] ranging from 1 to 4).

```
OPTIMAL-BST(p, q, n)
1. let e[1..n+1, 0..n], w[1..n+1,0..n]
        and root[1..n,1..n] be new tables
2. for i = 1 to n + 1
3.      e[i, i-1] = q_{t-1}
4.      w[i,j] = q_{t-1}
5. for l = 1 to n
6.    for l = 1 to n – l + 1
7.        j = i + l – 1
8.        e[i, i-1] = ∞
9.        w[i,j] = w[i, j – 1] + p_j + q_j
10.       for r = i to j
11.           t = e[l, r – 1] + e[r + 1, j] + w[i,j]
12.           If t < e[i,j]
13.             e[i,j] = t
14.             root[i, j] = r
15. return e and root
```

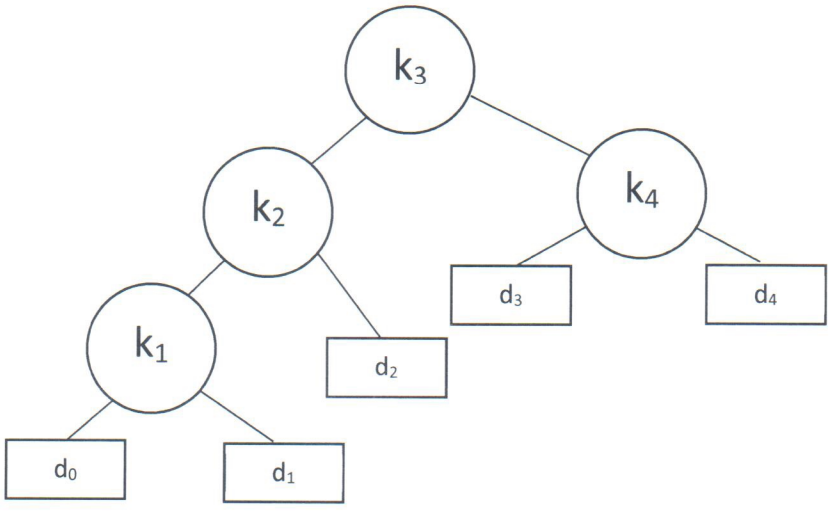
Given

i	0	1	2	3	4
k_i		0.16	0.13	0.2	0.08
d_i	0.12	0.1	0.06	0.11	0.04

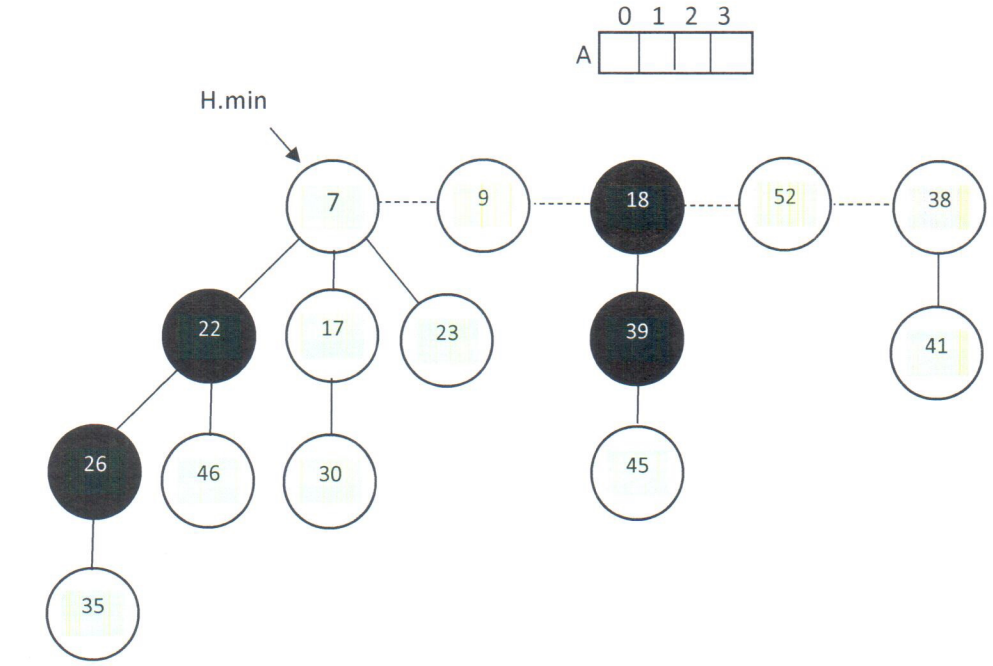
e		i				
		1	2	3	4	5
j	4	2.52	1.55	0.94	0.38	0.04
	3	2.02	1.16	0.37	0.11	
	2	1.14	0.45	0.06		
	1	0.6	0.1			
	0	0.12				

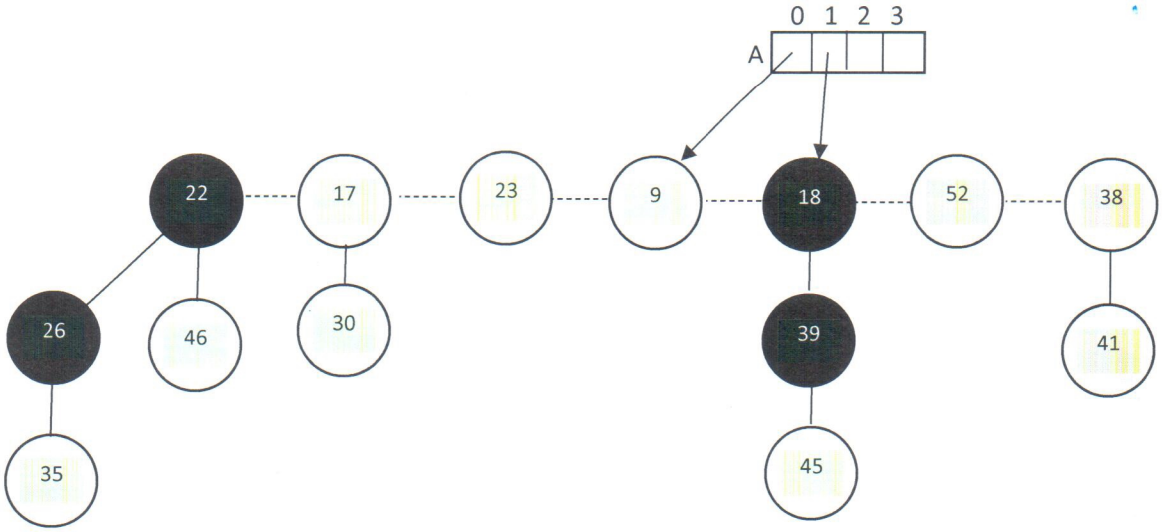
root	i				
		1	2	3	4
j	4	3	3	3	4
	3	2	3	3	
	2	1	2		
	1	1			

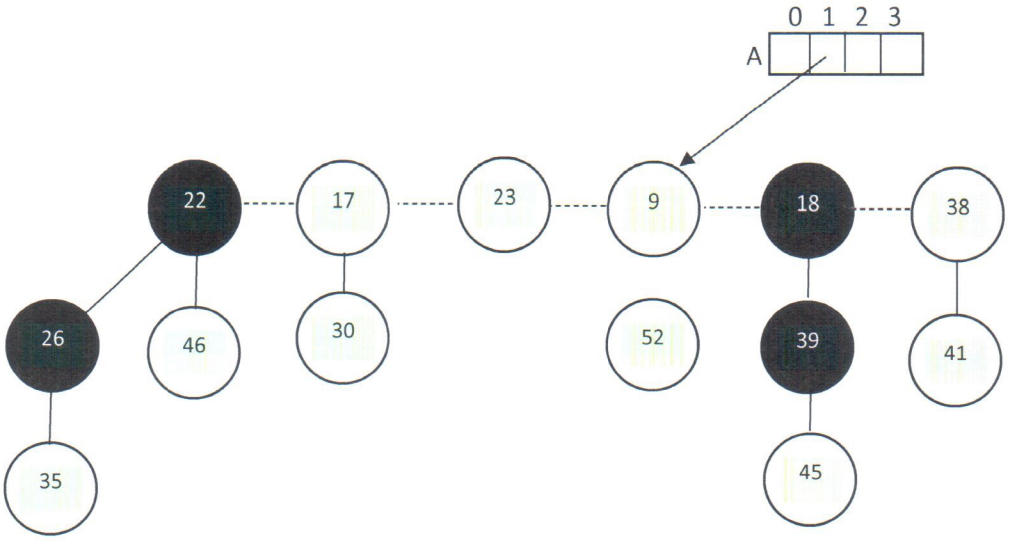
w	i					
		1	2	3	4	5
j	4	1.00	0.72	0.49	0.23	0.04
	3	0.88	0.6	0.37	0.11	
	2	0.56	0.29	0.06		
	1	0.38	0.1			
	0	0.12				

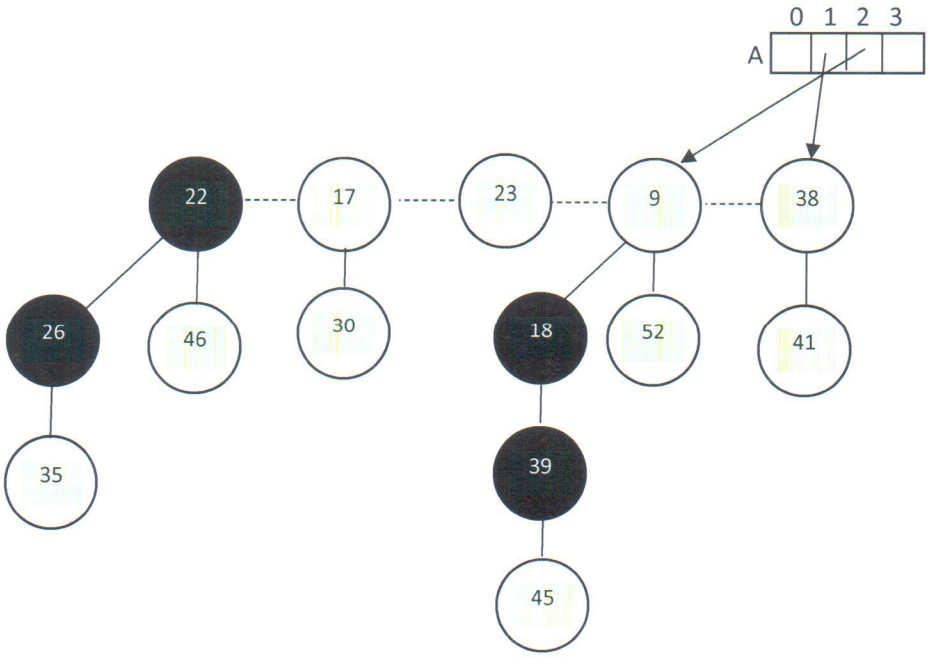


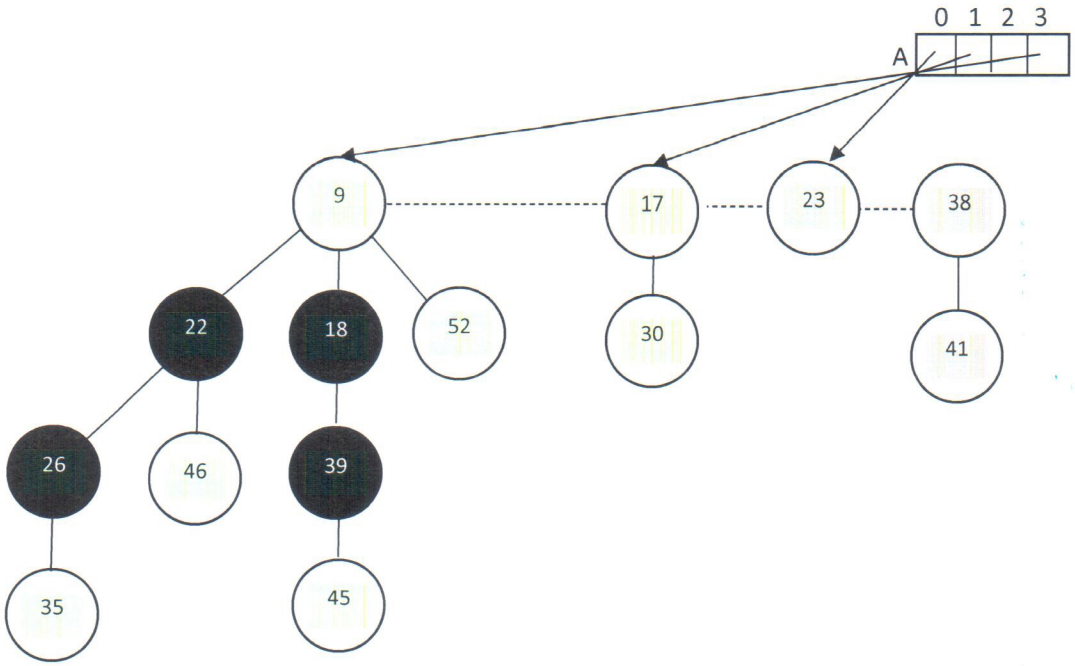
4. A Fibonacci min-heap (H) relies on the procedure of CONSOLIDATE to merge trees in the root list upon the operation of extracting the minimum node. For H below, show the final result after extracting H.min. (Also, give the result of every key consolidation step till its completion.)

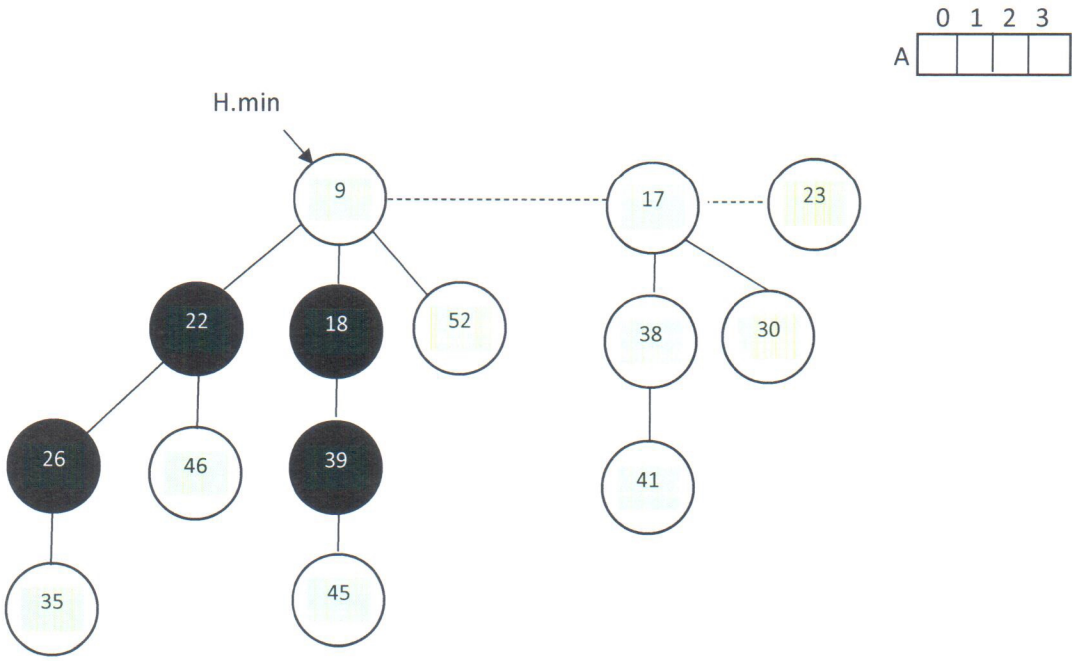





0 1 2 3
A
22
17
23
9
18
38
26
46
30
52
39
41
35
45


0 1 2 3
A
22
17
23
9
38
26
46
30
18
52
41
35
39
45


0 1 2 3
A
9
17
23
38
22
18
52
30
41
26
46
39
35
45


0 1 2 3
A
H.min
9
17
23
22
18
52
38
30
26
46
39
41
35
45