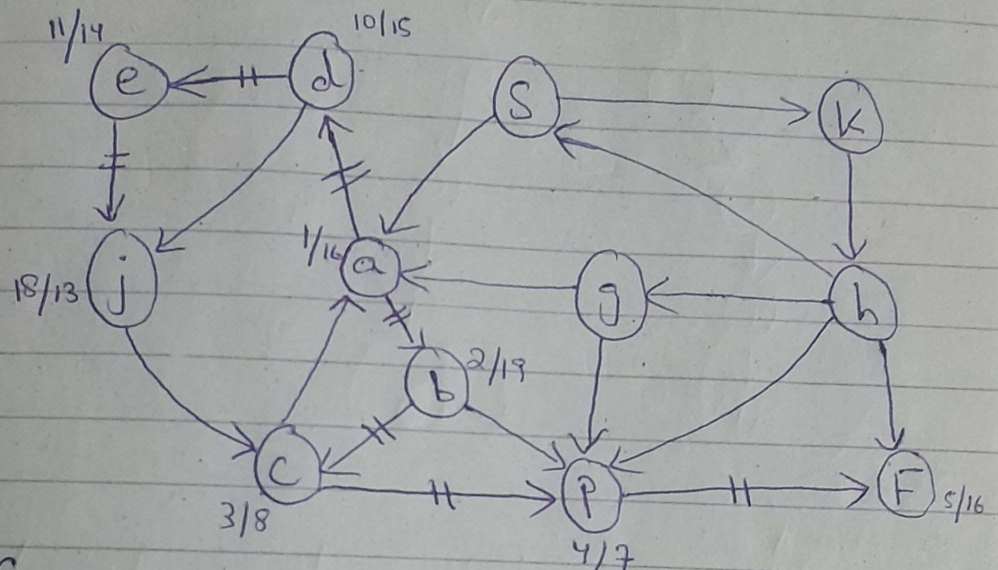
Roll no = 21k-3881
Section = BSE-5A

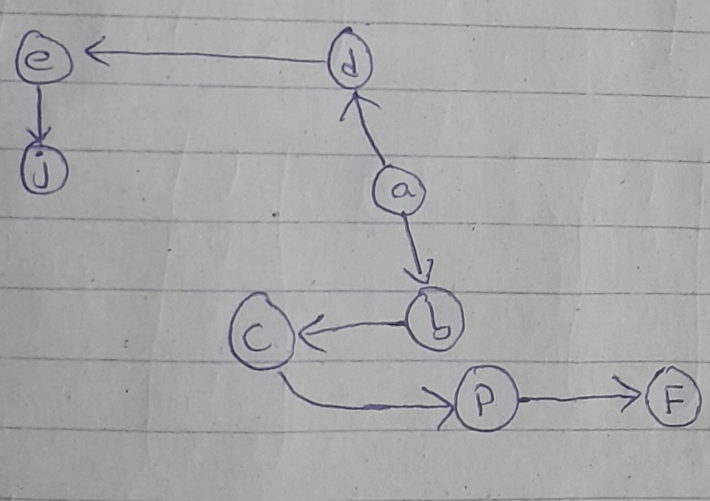
Design And Analysis of Algorithm (DAA)

Assignment = 04

Q1) (a)

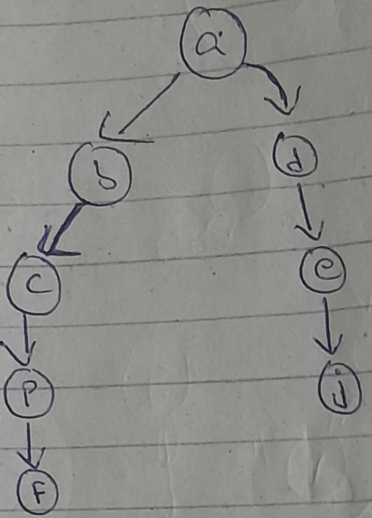


Graph:



21k-3881

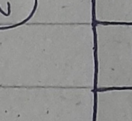
Tree:



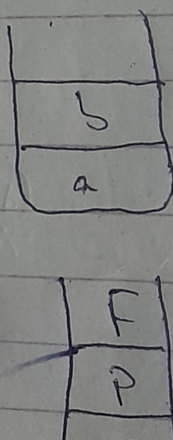
Stack:

Starting from "a":

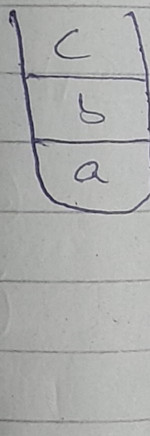
(i)



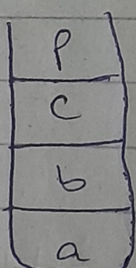
(ii)



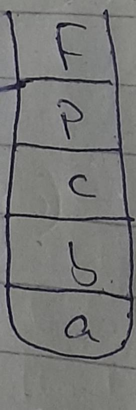
(iii)



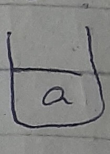
(iv)

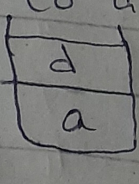


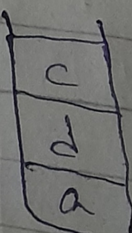
(v)

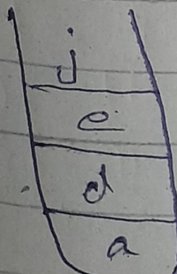


Return to "a":

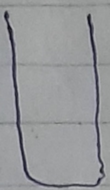








21k-3881



empty Stack, no more path

An

(b)

Back Edges: {(c-a)}

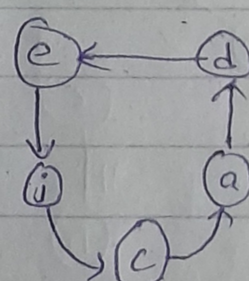
Forward Edges: {(b-p), (d-j)}

~~Cores~~

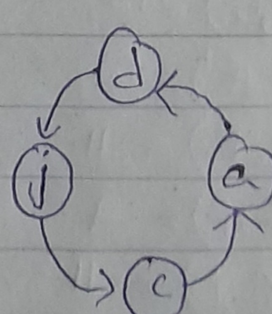
Cross Edges: {(j-c)}

(c)

(i) d-e-j-c-a-d

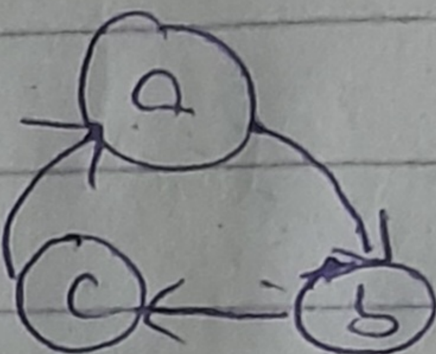


(ii) d-j-c-a-d

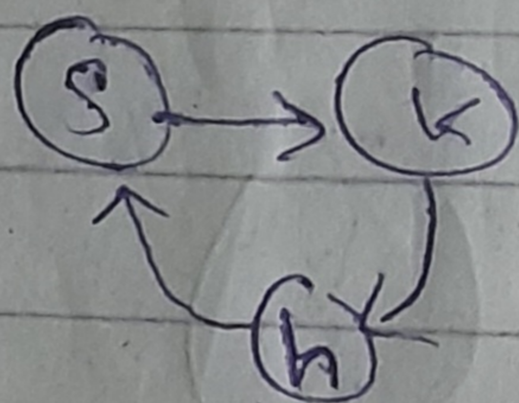


21k-3889

(3) a - b - c - a



(4) s - k - h - s

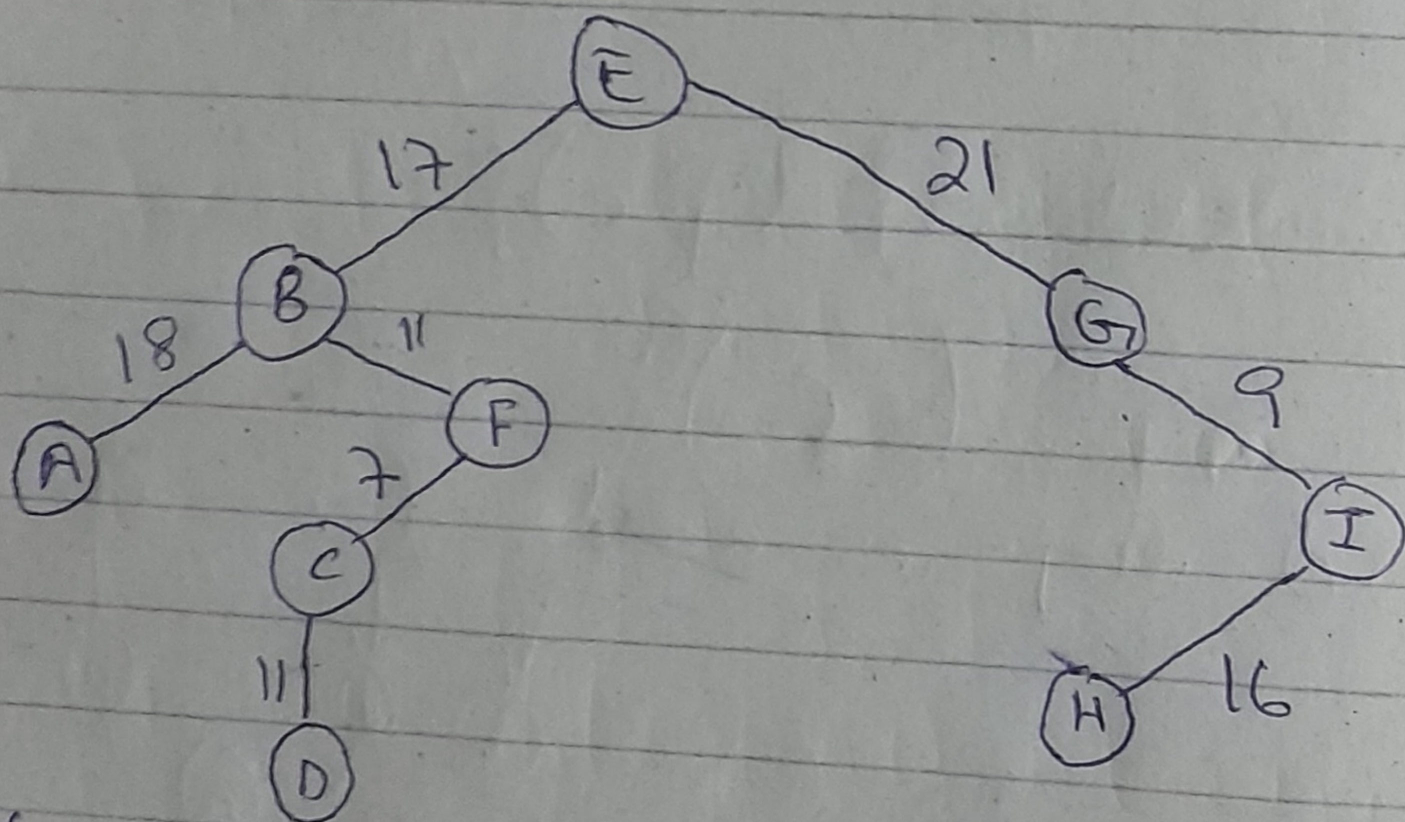


21k-3881

Q2) (a)

Ans = kruskal's Algorithm start/begin by selecting an edges but not a node.

Q2) (b)



(A,B) = 18
(B, E) = 17
(B, F) = 11
(C, F) = 7
(C, D) = 11

(E, G) = 21
(G, I) = 9
(H, I) = 16

Sum = 110 total length

Q2) (c)

Ans = The time Complexity of Kruskals algorithm is primarily dominated by sorting of edges, which is $O(E \log E)$, where E is no. of Edges. For the checking of cycles and union find operations typically $O(\log V)$, where V is no. of vertices.

So, the overall time complexity of algorithm is $O(E \log E + V \log V)$

$O(E \log E)$

21k-3881

Q3)

Formula:

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

$$A^1 = \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 8 & \infty & 1 \\ \infty & 0 & 1 & \infty \\ 4 & 12 & 0 & 5 \\ \infty & 2 & 9 & 0 \end{bmatrix} \end{array}$$

$$A^1[2,3] = \min\left(A^0[2,3],\ A^0[2,1] + A^0[1,3]\right)$$

$$'' = \min(1,\ \infty + \infty)$$

$$A^1[2,3] = 1$$

$$A^1[2,4] = \min\left(A^0[2,4],\ A^0[2,1] + A^0[1,4]\right)$$

$$'' = \min(\infty,\ \infty + 1)$$

$$A^1[2,4] = \infty$$

21k-3881

$A^1[3,2] = \min(A^0[3,2], A^0[3,1] + A^0[1,2])$

$\quad " \quad = \min(\infty, 4+8)$

$A^1[3,2] = 12$

$A'[3,4] = \min(A^0[3,4], A^0[3,1] + A'[1,4])$

$\quad " \quad = \min(\infty, 4+1)$

$A'[3,4] = 5$

$A'[4,2] = \min(A^0[4,2], A^0[4,1] + A^0[1,2])$

$\quad " \quad = \min(2, \infty + 8)$

$A'[4,2] = 2$

$A'[4,3] = \min(A^0[4,3], A^0[4,1] + A^0[1,3])$

$\quad " \quad = \min(9, \infty + \infty)$

$A'[4,3] = 9$

Now,

$$A^2 = \begin{bmatrix} 0 & 8 & 9 & 1 \\ \infty & 0 & 1 & \infty \\ 4 & 12 & 0 & 5 \\ \infty & 2 & 3 & 0 \end{bmatrix}$$

$A^2[1,3] = \min(A'[1,3], A'[1,2] + A'[2,3])$

$\quad = \min(\infty, 8+1)$

$A^2[1,3] = 9$

Alk-3881

$A^2[1,4] = \min\left(A^1[1,4],\ A^1[1,2] + A^1[2,4]\right)$

" $= \min(1,\ 8+\infty)$

$A^2[1,4] = 1$

$A^2[3,1] = \min\left(A^1[3,1],\ A^1[3,2] + A^1[2,1]\right)$

" $= \min(4,\ 12+\infty)$

$A^2[3,1] = 4$

$A^2[4,1] = \min\left(A^1[4,1],\ A^1[4,2] + A^1[2,1]\right)$

" $= \min(\infty,\ 2+\infty)$

$A^2[4,1] = \infty$

$A^2[3,4] = \min\left(A^1[3,4],\ A^1[3,2] + A^1[2,4]\right)$

" $= \min(5,\ 4+\infty)$

$A^2[3,4] = 5$

$A^2[4,3] = \min\left(A^1[4,3],\ A^1[4,2] + A^1[2,3]\right)$

$= \min(9,\ 2+1)$

$A^2[4,3] = 3$

21k-3881

$$A^3 = \begin{bmatrix} 0 & 8 & 9 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 12 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{bmatrix}$$

$A^3[1,2] = \min(A^2[1,2], A^2[1,3] + A^2[3,2])$

$\quad = \min(8, 9+12)$

$A^3[1,2] = 8$

$A^3[1,4] = \min(A^2[1,4], A^2[1,3] + A^2[3,4])$

$A^3[1,4] = \min(1, 9+5)$

$A^3[1,4] = 1$

$A^3[2,1] = \min(A^2[2,1], A^2[2,3] + A^2[3,1])$

$\quad = \min(\infty, 1+4)$

$A^3[2,1] = 5$

$A^3[2,4] = \min(A^2[2,4], A^2[2,3] + A^2[3,4])$

$\quad = \min(\infty, 1+5)$

$A^3[2,4] = 6$

21k-3881

$A^3[4,1] = \min(A^2[4,1], A^2[4,3] + A^2[3,1])$

$\quad = \min(\infty, 3+4)$

$A^3[4,1] = 7$

$A^3[4,2] = \min(A^2[4,2], A^2[4,3] + A^2[3,2])$

$\quad = \min(2, 3+12)$

$A^3[4,2] = 2$

$$A^4 = \begin{bmatrix} 0 & 3 & 4 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 7 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{bmatrix}$$

$A^4[1,2] = \min(A^3[1,2], A^3[1,4] + A^3[4,2])$

$\quad = \min(8, 1+2)$

$A^4[1,2] = 3$

$A^4[1,3] = \min(A^3[1,3], A^3[1,4] + A^3[4,3])$

$\quad = \min(9, 1+3)$

$A^4[1,3] = 4$

$A^4[2,1] = \min(A^3[2,1], A^3[2,4] + A^3[4,1])$

$\quad = \min(5, 6+7)$

$A^4[2,1] = 5$

21k-3881

$A^4[2,3] = \min(A^3[2,3], A^3[2,4] + A^3[4,3])$

$= \min(1, 6+3)$

$A^4[2,3] = 1$

$A^4[3,1] = \min(A^3[3,1], A^3[3,4] + A^3[4,1])$

$= \min(4, 5+7)$

$A^4[3,1] = 4$

$A^4[3,2] = \min(A^3[3,2], A^3[3,4] + A^3[4,2])$

$= \min(12, 5+2)$

$A^4[3,2] = 7$

$$\therefore \begin{bmatrix} 0 & 3 & 4 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 7 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{bmatrix}$$

Ans

⇒ The time complexity is $O(n \times n \times n) = O(n^3)$

⇒ Three loops will be used; one for matrix and two loops making matrix.

21k-3881

Q4)

Ans = Dijkstra algorithm is a greedy approach used for finding shortest path in weighted graph.

⇒ Initialization = Begins with source vertex, set its distance to 0 and to all other vertices distances to infinity.

⇒ Iteration = Repeat the given/following steps until all vertices are included.
→ Update the distances of its neighbouring vertices if shortest path is found through the current vertex.

⇒ Termination = The algorithm terminates when all vertices are included and shortest path to each vertex is determined.

21K-3881

Q5) (a)

Ans = Pseudocode

dis[s] ← 0

for all v ∈ V − {s}

do dis[v] ← −∞ Set all other distance to negative infinity

S ← ~~∅~~ ∅ (S, set at visited vertices initial empty)

Q ← V (Q, the queue initially contain all vertices).

while Q ≠ 0

do u ← maxDistance (Q, dis)

S ← S ∪ {u}

for all v ∈ neighbours [u] do

dis[v] = max (dis[v], dis [u] + w (u, v))

update distance

return dis;

21k-3881

(5)

①	a	b	c	d	e	f	g
	$\boxed{0}$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
a		$\boxed{12}$	5	$-\infty$	$-\infty$	$-\infty$	$-\infty$
ab			20	$\boxed{24}$	23	$-\infty$	$-\infty$
abd			27		30	$\boxed{31}$	$-\infty$
abdf			40		$\boxed{46}$		41
abdfe			40				$\boxed{59}$
abdfeg			$\boxed{40}$				
abdfegc							

$\therefore$ {0(a), 12(b), 24(d), 31(f), 46(e), 59(g), 40(c)} Ans

21K-3881

Q6) (a)

O	E	O	O	T
V	O	Q	U	O
E	O	I	H	O
R	T	G	H	F

(b and c)

Use function of DFS to find the word indexes ⇒ if the remaining word is empty return start and end coordinates.

⇒ if the current index is out of bound or current character does not match current word letter return none

⇒ Move in current direction check if the next character = next word character.

if the entire word like the 1st case is found then return start and end position.

Initialize a empty list to store

21k-3881

start and end indexes of word, it is list to store multiple start and end points also remove duplicates.

Loop over entire grid:

⇒ if current position = current character call DFS on all direction i.e - up, down, left and right.

⇒ if DFS returns valid position append to list.

return the list containing coordinates.