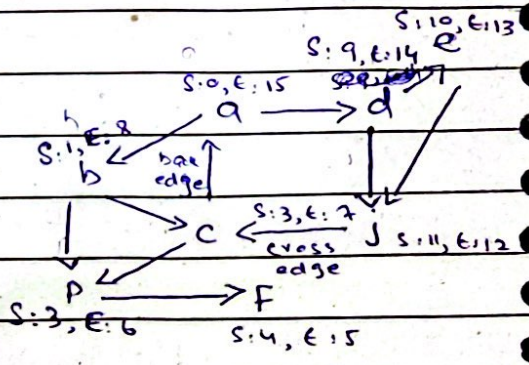
ALGO ASSIGNMENT #04

Date:

Question: 01 (a)

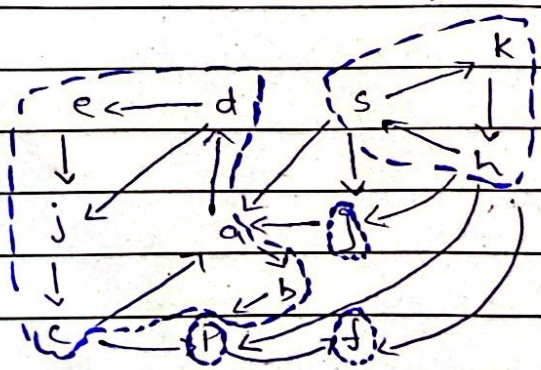
Samad Amir
21K-3873
BSE-5A

DFS(a) → DFS(b) or DFS(d)
DFS(b) → DFS(c) or DFS(p)
DFS(d) → DFS(e) or DFS(j)
DFS(c) → DFS(p)
DFS(p) → DFS(f)
DFS(e) → DFS(j)
DFS(j) → DFS(a) dead
DFS(f) → dead
DFS: [f, j, e, p, c, d, b, a]



b) cross edge {(j, c)}
forward edge {(a,d), (a,b), (b,p), (p,f), (d,e), (d,j), (b,c), (c,p), (e,j)}
Back edge {(c,a)}

c) a, b, c, d, j, e
p
f
g
h, k, s

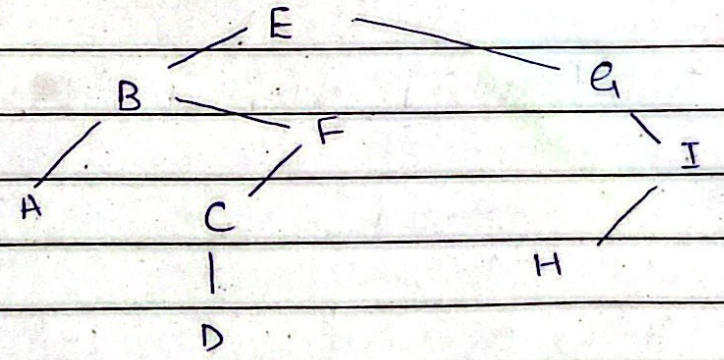


Date:

(Question 02)

a) Kruskal Algorithm begins by edges having the least weight

b) F – C = 7
G – I = 9
C – D = 11
B – F = 11
B – C = 11 ×
F – D = 15 ×
H – I = 16
B – E = 17
A – B = 17
A – C = 20 ×
E – F = 21 ×
E – G = 21
F – H = 23 ×



- no cycle
- no parallel edges
- no self loop

Total length = 7 + 9 + 11 + 11 + 16 + 17 + 18 + 21 = 110

Date:

c)

- for adding a node/edge we heapify and it takes logn times in heap sort to place a member in its correct position. log e.

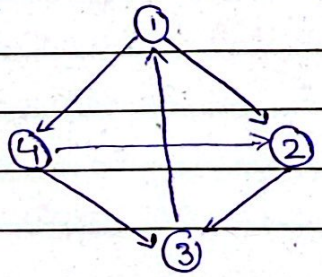
- Best case ; we will visit (n-1) edges
- worst case ; we will visit (e) edges.

In this example we visit n-1 edges

So,

$(n-1)(\log e) \Rightarrow O(n \log e)$

3)



$$D^{(0)} = \begin{bmatrix} 0 & 8 & \infty & 1 \\ \infty & 0 & 1 & \infty \\ 4 & \infty & 0 & \infty \\ \infty & 2 & 9 & 0 \end{bmatrix}$$

Recursive Relation

$$A[i,j] = \min\{A[i,j],\ A[i,k] + A[k,j]\}$$

$$D' = \begin{bmatrix} 0 & 8 & \infty & 1 \\ \infty & 0 & 1 & \infty \\ 4 & 12 & 0 & 5 \\ \infty & 2 & 9 & 0 \end{bmatrix}$$

Date:

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

$= \min [1, \infty + \infty]$

$= 1$

$A[2,4] = \min [A[2,4], A[2,1] + A[1,4]]$

$= \min [\infty, \infty + 1]$

$= \infty$

$A[3,1] = \min [A[3,1], A[3,1] + A[1,1]]$

$= \min [4, 4 + 0] = 4$

$A[3,2] = \min \{A[3,2] + A[1,2], A[3,2]\}$

$= \min [4 + 8, \infty]$

$A[3,4] = \min \{A[3,4], A[3,1] + A[1,4]$

$= \min [\infty, 4 + 1]$

$= 5$

$A[4,2] = \min [A[4,2], A[4,1] + A[1,2]$

$= \min [2, \infty + 8]$

$= 2$

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

$= \min [9, \infty + \infty]$

$= 9$

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

Date:

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

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

$= 9$

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

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

$= 1$

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

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

$= 4$

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

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

$= 5$

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

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

$= \infty$

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

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

$= 3$

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

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

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

$= 8$

Date:

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

$= \min(1, 9+5^-)$

$= 1$

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

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

$= 5$

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

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

$= 6$

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

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

$= 7$

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

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

$= 2$

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

Final output

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

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

$= 3$

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

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

$= 4$

Date:

$A[2,1] = \min(A[2,1], A[2,4] + A[4,1])$
$= \min(5, 6+7)$
$= 7$

$A[2,3] = \min(A[2,3], A[2,4] + A[4,3])$
$= \min(1, 6+3)$
$= 1$

$A[3,1] = \min[A(3,1), A(3,4) + A(4,1)]$
$= \min(4, 5+7)$
$= 4$

$A[3,2] = \min[A[3,2], A[3,4] + A[4,2])$
$= \min(12, 5+2)$
$= 7$

Time Complexity:
As three variable i, j, K where i, j will be used for putting values and K will be used to generate matrix and all three has to run till n so final output will be $n \times n \times n = O(n^3)$

40)
To verify the Dijkastra's Algorithm's correctness, we need to ensure that the relaxation step is safe.
lemma: the relaxation operation maintains the invariant
$d[v] \geq p(s,v)$ for all $v \in V$ where $d[v]$ is the the length of shortest path from S, $p(s,v)$ is from S to v

Date:

Procf:-

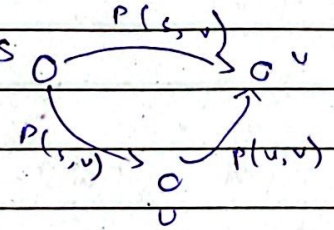
By the process of induction on the number of steps.

$d[v] \geq p(s,v)$ as per depending on our lemma

Now as per the triangle in equality

$$p(s,v) \leq p(s,v) + s(u,v)$$

Now to explain the triangle inequality, we use the example



There seems to have no contradiction with a weighted graph. As we can see from the graph, the directed path $s \to u \to v$ is the shortest path from s to v. Now this contradicts Δ inequality, but after the process of relaxation $p(s,v)$ becomes shortest path; it is derived from $p(s,v) + p(u,v)$

Equation of Δ inequality

$$p(s,v) \leq p(s,v) + p(s,v)$$

$$\therefore p(s,v) \leq d[u], p(u,v) - w(u,v)$$

$$\Rightarrow p(s,v) \leq d[u] + w(u,v)$$

$$\Rightarrow d[u] + w[u,v] \geq d[v]$$

$\Rightarrow p(s,v) \leq d[v]$ which is in accordance with all lemma.

Date:

5) Algo widest Path (Graph, Source)

previous [ ] = length of v

previous = -1

width [ ] = length of v

width = 0

vertices = v

queue . push [v . source]

width [source] = Max

while (queue is not empty)

P, u = queue . pop

for v in neighbours of u

dis = max [width [v], min (width [v], distance [u, v])]

if (dis > with [v])

width [v] = dis

previous [v] = u

queue . push (dis , v)

end if

end for

end while

return width, previous

return widest - path

	a	b	c	d	e	f	g
Previous	-1	-1	-1	-1	-1	-1	-1
width	∞	0	0	0	0	0	0

Heap [(∞, a)]

	a*	b	c	d	e	f	g
Previous	-1	a	a	-1	-1	-1	-1
width	∞	12	5	0	0	0	0

Heap [(5, c), (P, b)] Page ☐ Victory

Date:

	a	b	c*	d	e	f	g
Previous	-1	a	a	c	-1	c	-1
width	∞	12	5	3	0	5	0

Heap : [(3, d), (5, f), (12, b)]

	a	b	c	d*	e	f	g
Previous	-1	a	a	c	d	c	d
width	∞	12	5	3	3	5	3

Heap : [(3, e), (3, g), (5, f), (12, b)] No chang. when passing e, g

	a	b	c	d	e	f	g
previous	-1	a	a	c	f	c	f
width	∞	12	5	3	5	5	5

Heap [(5, e), (5, g), (12, b)]

	a	b	c	d	e*	f	g
Previous	-1	a	a	e	f	c	f
width	∞	12	5	5	5	5	5

Heap [(5, d), (5, g), (12, b)]

No change still while processing d and g

	a	b*	c	d	e	f	g
Previous	-1	a	b	b	b	c	f
width	∞	12	8	12	11	5	5

Heap : [(8, c), (11, e), (12, d)]

Date:

	a	b	c	d	e	f	g
Previous	-1	a	b	b	b	c	f
width	∞	12	8	12	11	8	8

Heap [(8, g), (1, c), (12, d)]

No chaye when processing g

	a	b	c	d	e	f	g
Previous	-1	a	b	b	b	e	e
width	∞	12	8	12	11	11	11

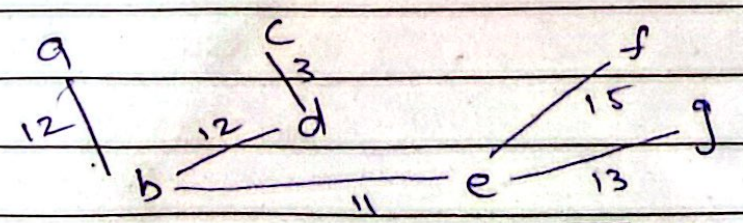
Heap : [(11, f), (11, g), (12, d)]

	a	b	c	d	e	f	g
Previous	-1	a	f	b	b	e	e
Width	∞	12	9	12	11	11	11

Heap : [(9, c), (11, g), (12, d)]

No chaye when process remaining

Target	path	width
b	a → b	12
c	a → b → e → f → c	9
d	a → b → d	12
e	a → b → e	11
f	a → b → e → f	11
g	a → b → c → g	11



widest path Tree

```
rows = 4
col = 5
bool check_word (char table[rows][col], start_row
                 start, dx, dy, word, cd)
{
  length = strlen(word)
  for (i=0; i<length; i++)
  x = start_row + i x dx
  y = start_col + i x dy;
  if (x<0 || x>rows || y<0 || y>cols || table[x][y] != word[i]
    return false
  end if
  end loop
    return true
  end
void word_search (char table[rows][col], word]
direction[4][2] = { [0,1], [1,0], [-1,0], [0,-1] }
    for (i=0; i<rows; i++)
      for (j=0; j<cols; j++)
        if (check_word (table, i, j, direction[d][0],
                        directions[d][1], word)
z = i + (length(word)-1) * directions[d][0]
q = j + (length(word)-1) * directions[d][1];
print (start, end, i, j, z, q);
end if
end loop
end loop
end loop
```

Page ☐ Victory

Date:

```
main ( )
table [rows] [cols] = { {'O', 'f', 'o', 'o', 'T'},
                        {'v', 'O', 'O', 'U', 'O'},
                        {'F', 'O', 'I', 'H', 'O'},
                        {'R', 'T', 'G', 'H', "f"}
                      };

word [ ] = " foot "
word_search (grid, word);
return 0;
```