Abdul Sami Qasim
22i-1725

Algorithms

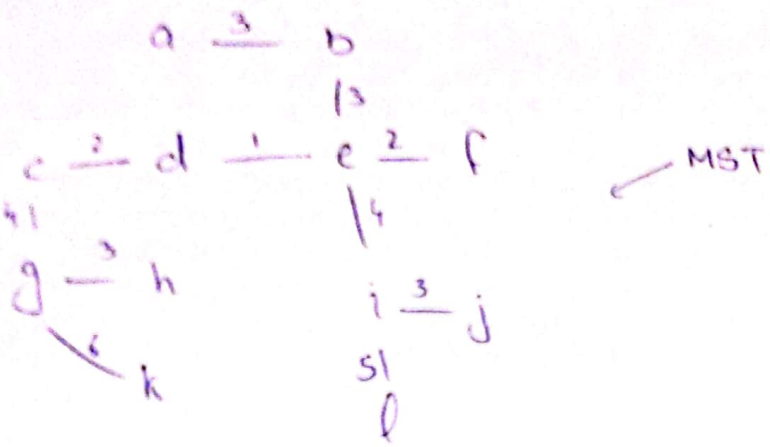
Assignment 3

CY-D

To:-

Amina Siddique

3) a)



sequence:

a to b
b to e
e to d
d to c
e to f
c to g
g to h
e to i
i to j
i to l
g to k

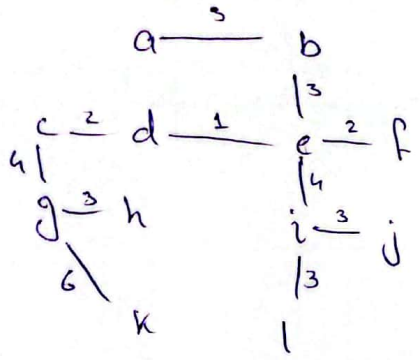
b) This algorithm does not check connectivity by itself and it assumes that the provided graph is connected.

c) if we include another vertex, it has to be included in the MST as well. the edges wont all be included. Only the one with the least weight will be included in the MST.

2) making a list of connections by sorting weight.

d-e
e-f
c-d
a-b
b-e
g-h
i-j
a-d
c-g
e-i
c-a
d-h
f-j
i-l
b-f
h-i
g-k
h-k
k-l
l-j

we connect these edges one by one until all vertices are connected, neglect any edge that results in a cycle.



This is the resulting MST

)

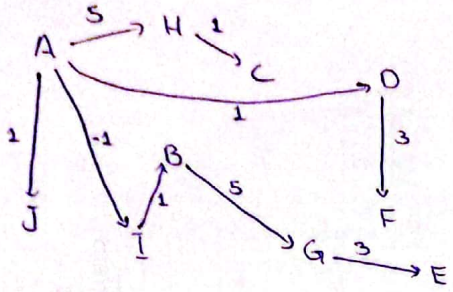
Dijkstra does not work on negative edge graphs, it is a limitation for this algorithm, we use bellman-ford algo for negative edge graphs.

Using dijkstra:-

table:-

A	B	C	D	E	F	G	H	I	J
0	∞	∞	∞	∞	∞	∞	∞	∞	∞
0	∞	∞	1	∞	∞	∞	5	-1	1
0	0	6	1	∞	4	∞	5	-1	1
0	0	6	1	∞	4	5	5	-1	1
0	0	6	1	8	4	5	5	-1	1

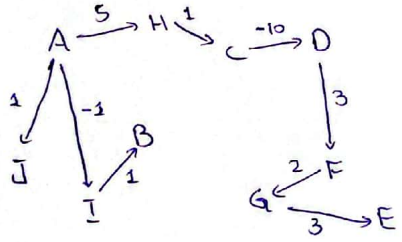
The resulting MST :-



Now with bellman-ford :-

A	B	C	D	E	F	G	H	I	J
0	∞	∞	∞	∞	∞	∞	∞	∞	∞
0	∞	∞	1	∞	∞	∞	5	-1	1
0	0	6	1	∞	4	∞	5	-1	1
0	0	6	-4	∞	-1	1	5	-1	1
0	0	6	-4	4	-1	1	5	-1	1

resulting MST :-



b)

Dijkstra fails to provide a correct answer when negative edges are involved. It's a limitation for this algorithm.

Q4)

```
LCS (St1, St2){
    m= St1.length
    n= St2.length
    dp[m+1][n+1] ← 0
    maxlength ← 0
    endpos ← 0
    for i from 1 to m{
        for j from 1 to n{
```

```
        if (dp St1[i-1] == St2[j-1]){
            dp[i][j] ← dp[i-1][j-1]+1
            if (dp[i][j] > maxlength){
                maxlength ← dp[i][j]
                endpos ← i
            }
        }
        else{
            dp[i][j] ← 0
        }
    }
}
startindex ← endpos - maxlength
LCS ← A[startindex : endpos]

Return LCS, maxlength
}
```

Now, performing dryrun:-

		b	c	d	e	g	g	f	e	f	g	g
	0	0	0	0	0	0	0	0	0	0	0	0
a	0	0	0	0	0	0	0	0	0	0	0	0
b	0	1	0	0	0	0	0	0	0	0	0	0
c	0	0	2	0	0	0	0	0	0	0	0	0
d	0	0	0	3	0	0	0	0	0	0	0	0
e	0	0	0	0	4	0	0	0	1	0	0	0
f	0	0	0	0	0	0	0	1	0	2	0	0
e	0	0	0	0	0	0	0	0	2	0	0	0
f	0	0	0	0	0	0	0	1	0	3	0	0
g	0	0	0	0	0	0	0	0	0	0	4	1
h	0	0	0	0	0	0	0	0	0	0	0	0

Q5)

```
count (n){
    dp [n+1][n+1] ← 0
    dp [0][j] ← 1
    dp [i][0] ← 0
    for i from 1 to n{
        for j from 1 to n{
            if (i<j){
                dp [i][j] ← dp [i][j-1]
            }
            else{
                dp [i][j] ← dp [i][j+1] + dp[i-j][j]
            }
        }
    }
    dp [n][n] ← dp [n][n-1]
    return dp [n][n-1]
}
```

i\j	0	1	2	3	4	5	6	7	8	9	10
0	1	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1
2	0	1	2	2	2	2	2	2	2	2	2
3	0	1	2	3	3	3	3	3	3	3	3
4	0	1	3	4	5	5	5	5	5	5	5
5	0	1	3	5	6	7	7	7	7	7	7
6	0	1	4	7	9	10	11	11	11	11	11
7	0	1	4	8	11	13	14	15	15	15	15
8	0	1	5	10	15	18	20	21	22	22	22
9	0	1	6	12	18	23	26	28	29	30	30
10	0	1	7	14	23	30	35	38	40	41	41

if n=10, max count is 41