Name – Sayem Lincoln

PID- A54207835

CSE 431 Assignment 3

1. Comment: as it is said in lecture 6 in case of pre order we first print current node, then we move to our left and do the same when we finish it we move to our right. If we draw image shown in picture like a tree it’s easy to understand. Same happens with post order and in order

Preorder:

First tree: a, b, f, g, c, h, d, l, o, k, p

Second tree: e, i, n, j, m

Post order:

First tree: d, k, o, p, l, h, c, g, f, b, a

Second tree: m, j, n, i, e

In order:

First tree: d, h, k, o, l, p, c, g, f, b, a

Second tree: m, j, n, i, e

1. Comment: we only updated the if statement. If in case of dfs we always make a recursive call now we check if current distance plus edge length is greater than l than we don’t do a recursive call.

If v is unmarked

Mark v

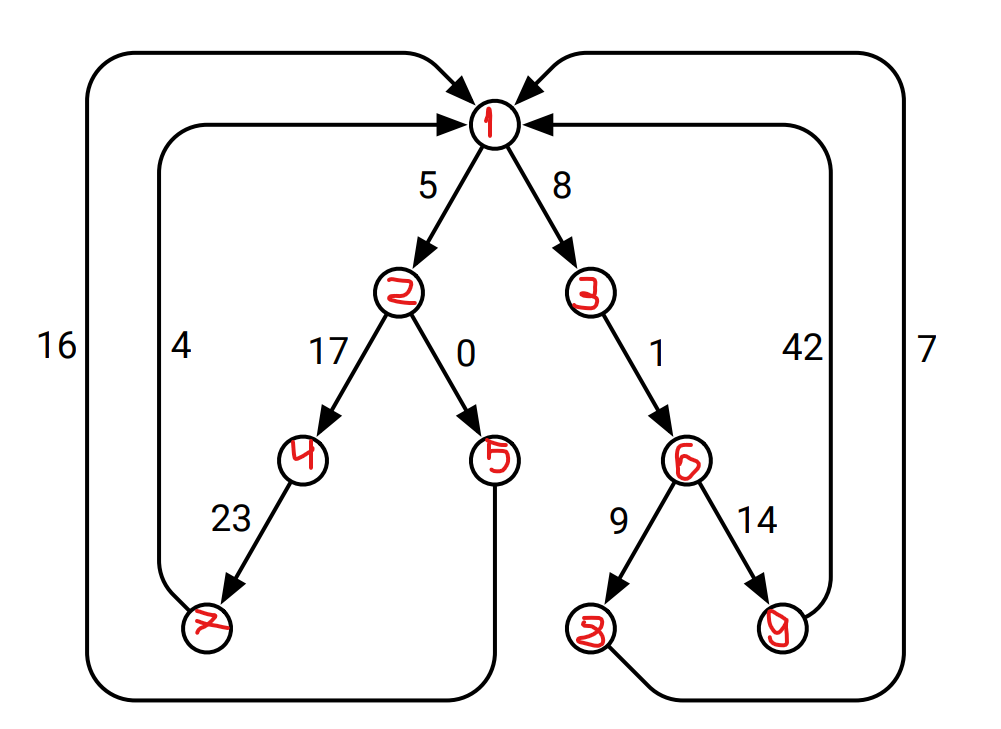
For each edge v -> w

If current\_dist + dist[v][w] <= l

DFS(w, current\_dist + dist[v][w] )

Dist[v][w] is the edge connecting v and w

Current\_dist is second argument of DFS function I save all the distances in recursive call



I numbered all the vertices.

1. First we start in 9 and d[9] = 0 and distance to all the other vertices is INT\_MAX

We add 9 in our set

9 has one neighbor 1 distance from 9 to 1 is 42; 42 < INT\_MAX so d[1] = 42

Now we find vertex with minimum distance that is 1;

We add 1 to our set

1 has 2 neighbors d[2] + w(1,2) = 42 + 5 = 47; 47 < INT\_MAX so d[2] = 47;

Analogically d[1] + w(1,3) = 50 and d[3] = 50

Now minimum distance has 2

2 has 2 neighbors d[2] + w(2, 4) = 47 + 17 = 64; 64 < d[4] so d[4] = 64,

Analogically d[2] + w(2, 5) = 47 +0 = 47 so d[5] = 47

We add 2 to our set

Now minimum distance has 3

3 has 1 neighbor d[3] + w(3, 6) = 51; 51 < INT\_MAX d[6] = 51

We add 3 in our set

Minimum is 5

D[5] + w(5 , 1) = 51 + 16 which isn’t less than d[1] so d[1] stays the same

We add 5 in our set

Now minimum is 6

D[6] + w(6, 8) = 51 + 9 = 60 ; d[8] = 60

D[6] + w(6 , 9) = 51 + 14 which is more than d[9] so we move on

We add 6 in our set

Now minimum is 8

D[8] + w(8,1) is more than d[1] and nothing changes

Now minimum is 4

D[4] + w(4, 7) = 64 + 23 = 87; d[7] = 87

We add 4 in our set

Now minimum is 7

D[7] + w(7 , 1) is more than d[1] so nothing changes

We add 7 in our set

1. a.

Dijsktra (G, vs) // starting vertex vs

D[vs] ← 0 // width estimate for starting node is 0

For each v in V and v ≠ vs

D[v] ← INT\_MIN // initialize all width estimates to minimum value possible

Q.add(vs) // Q is maximum priority queue of nodes

While !Q.empty() do

u ← extractmax(Q) // extract node with maximum width

For each v in Adj(u)

Dist ← max(min(d[u], w(u,v)), d[v])

If Dist > d[v] // check if path width can be updated

D[v] ← Dist

Q.add(v)

b .

We have 0 in our queue

0 have 2 neighbors

max(min(d[0], w(0,5)), d[5]) = 3; d[5] = 3

Analogically d[6] = 4

We add 5 and 6 is our queue

Now maximum is 6 in our queue

6 has 2 neighbors

max(min(d[6], w(6,2)), d[2]) = 0 d[2] = 0

max(min(d[6], w(6,4)), d[4]) = 0 d[4] = 4

We add 2 and 4 in our queue

Our new maximum is 4

4 has 2 neighbors

max(min(d[4], w(4,2)), d[2]) = 1 1 > d[2] so d[2] = 1

We add 2 in our queue which already is there

max(min(d[4], w(4,3)), d[3]) = 2 2 > d[3] so d[3] = 2

We add 3 in our queue

Now maximum is 5

5 has 2 neighbors

max(min(d[5], w(5,6)), d[6]) = d[6] so nothing changes

max(min(d[5], w(5,4)), d[4]) = d[4] so nothing changes

Now we have 3

3 doesn’t have any neighbors

Now we have 2

2 have 3 neighbors

max(min(d[2], w(2,3)), d[3]) = d[3] nothing changes

max(min(d[2], w(2,1)), d[1]) = 1 1 > d[1] so d[1] = 1

We add 1 in our queue

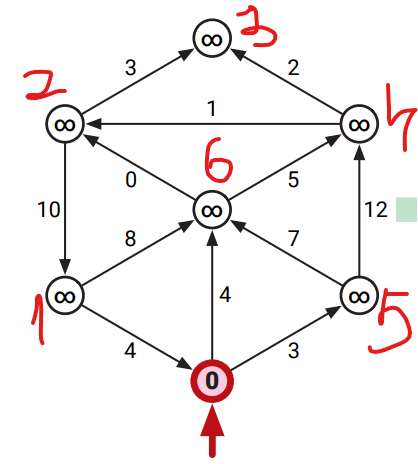
Now we only have 1 in our queue

It has 2 neighbors

max(min(d[1], w(1,6)), d[6]) = d[6] nothing changes

max(min(d[1], w(1,0)), d[0]) = d[0] nothing changes

Queue is empty



5. For every pair u and v there is two edge u-v or v-u so we have n \* (n - 1) / 2 different edges. Two graphs are different if their set of edges aren’t the same. So, we need to count how many subsets of these edges are there. For a set containing m elements there are exactly 2^m subsets. So, in our case answer will be 2 ^ (n \* (n - 1) / 2)