**Question 4: Mandatory Batman Question**

Divyansh Mathur,210357

1. **To design an algorithm that works in O(|V | · (|V | + |E|) = O(n · (n + m))**

dist(s,t): (shortest) distance path between s and t.

Point to note is that the shortest distance between 2 nodes doesn’t change if the edge not lying in the shortest path is removed. So we need to find shortest distance between s and t only when edges in this shortest path is removed.

To find dist(s,t): To find the shortest path between s and t simply start bfs from s and terminate it when it reaches t.

To store shortest path: Once we are pushing a node’s neighbor to queue, we know that this neighbor comes just after the node and hence we can say the neighbor’s parent is the node. Using this-

We make another array parent[n] where we store the parent of each vertex ( node ),

*Pseudocode to record parent[v] and distance(s,t) when none edgeis deleted:*

->Start BFS from s Until we reach t

->Initialising boolean visited array visited[n]=false  all false initially

->mark visited [s]=true

->Initialising Distance array Distance[n]=infinty(very high value)

->mark Distance [s]=0

->Initialising parent array parent[n]=-1 ( random not to be used value)

->Initialising empty queue Q

->then pushing s(starting node) to it

Q.push(s)

while(Q is not empty){

    u= Q.front

    Q.pop

    if(u == t) end BFS

    for( each neighbor v of u){

        if( visited[v]==false){

            if(Distance[v]>Distance[u]+1){

                Distance[v]=Distance[u]+1

            }

            visited[v]=true

            parent [v]=u

            Q.push(v)

        }

    }

}

*Data Structure to store the ans:* We are given with matrix (nxn), such that M[u,v] stores the dist(s,t) when (u,v) edge is removed. From above analysis we can infer that (u,v) affects dist(s,t) only if it is part of the shortest path and edges in shortest path are of the form (u,parent(u)), Since dealing with undirected graph we will do same operation on (parent(u),u). We need to alter values of (u,parent(u)) and (parent(u),u) starting from t.

We will run BFS each time to find distance (s,t) removing (u,parent(u)) edge

*Pseudocode to update matrix M:*

**UpdateMatrix**{

    node=t

    while(parent[node]!=-1){

        M[node][parent[node]]=newdistance(node, parent[node])

        M[parent[node]][node]=M[node][parent[node]]

        node= parent [node]

    }

}

**newdistance**(node , parent [node]){

    ->Start BFS from s Until we reach t

    ->Initialising boolean visited array visited[n]=false  all false initially

    ->mark visited [s]=true

    ->Initialising Distance array Distance[n]=infinty(very high value)

    ->mark Distance [s]=0

    ->Initialising empty queue Q

    ->then pushing s(starting node to it)

    Q.push(s)

    while(Q is not empty){

        u= Q.front

        Q.pop

        if(u == t) end BFS

        for( each neighbor v of u){

**if(u== node and v== parent[node] || v==node and u== parent [node]) skip;**

            else if( visited[v]==false){

                Distance[v]=Distance[u]+1

                visited[v]=true

                parent [v]=u

                Q.push(v)

            }

        }

    }

    return Distance[t]

}

1. **Time Complexity Analysis:** We know that BFS takes O(n+m) running time. In our case also BFS will take worst case O(n+m) running time. Also we are calling BFS each time for each node in the shortest path from s to t. There can be at max n-2 nodes in between and n-1 (u,parent(u)) pairs. Hence overall time complexity = O(n(n+m))
2. **Proof of correctness:**

It is obvious that the edges that don’t constitute the shortest path don’t disturb the dist(s,t)

Assertion: The algorithm correctly calculates the shortest distance once edge (u, parent(u)) is removed such that this edge lies in the shortest path.

Proof:

A[u]-Since we are running BFS and while running BFS we are skiping the edge encountered , we are in sense destroying the edge ( which is required), then we are calculating new distance, which implies that the new distance is the new shortest path.

A[parent[u]]- When the edge (parent(u), parent (parent (u))) is destroyed, again the shortest path cant be taken since it is destroyed now by again running BFS we find the new shortest path.

Hence Prooved.