Assignment 5

Vaibhav Agarwal(B15139)

Q1. Assuming all the capacities are integers,
s denotes the source, **t** denotes the sink in network flow problem.
m denotes number of edges in Network Flow Graph
C = $\Sigma_{e \text{ out of } s}\, c_e$. (Maximum possible flow out of the source).

In the Scaling max flow the total number of augmentations can be reduced if good bottleneck are chosen. This is done by using a scaling parameter Δ .The Δ is the value such that bottleneck will be at least Δ or greater so the Ford Fulkerson can be sped up by using scaling.

We have at most $[1 + \log_2 C]$ scaling phases and at most 2m augmentations in each scaling phase.

Also BFS for finding a new path takes O(n+m) => O(m) time, so the time complexity of Scaling Max Flow algorithm will be **$O(\, m^2 \log_2(C)\,)$** .

<u>Psuedocode:</u>

```
augment(f,P):
        (f , P)
        Let b = bottleneck(P, f)
        For each edge (u, v) ∈ P
                If e = (u, v) is a forward edge
                        then increase f(e) in G by b
                Else ((u, v) is a backward edge, and
                        let e = (v, u))
                        decrease f(e) in G by b
                Endif
        Endfor
        Return(f)
Scaling-Max-Flow:
        Initially f(e) = 0 for all e in G
        Initially set Δ to be the largest power of 2 that is no larger than the maximum capacity out of s:
        Δ ≤ max_{e out of s} c_e
        While Δ ≤ 1
                While there is an s-t path in the residual graph G_f
                        Let P be a simple s-t path in G_f
                        f' = augment(f , P)
                        Update f to be f'
                        Update the residual graph G_f to be G_f'
                Δ = Δ / 2
        Endwhile
        Return f
```

Q2. Assuming all the capacities are integers,
s denotes the source, **t** denotes the sink in network flow problem.
m denotes number of edges in Network Flow Graph
V denotes set of vertices in original Graph
E denotes set of edges in original Graph
n denotes the total number of nodes in the original graph.
C = $\mathbf{\Sigma}_{e \text{ out of } s}\, c_e$. (Maximum possible flow out of the source).

We have a Graph G = (V,E) and we are given with sets of vertices X,Y such that V=XUY, and every edge of the graph has one end in X and other end in Y. In this problem we are supposed to calculate maximum bipartite matching of a graph. That is , a subset M of all edges E whose each and every node involved appears at most once. In this problem we will have to calculate such a subset of largest possible size.

<u>Psuedocode:</u>

1. We add a source node s, and a sink node t to Graph.
2. We add edges from s to all nodes in X with capacity 1.
3. We add edges from X to nodes in Y with capacity 1 according to the previous edges.
4. We add edges from Y to all nodes in t with capacity 1.
5. Now we run ford fulkerson algorithm to find the max flow.
6. This max flow will denote the **maximum bipartite matching.**

Since the problem is solved using ford fulkerson , so the time complexity of Max Flow algorithm will be O(mC). In this case since maximum number of nodes in set X can be n so complexity is **O(mn)**

Q3. Assuming all the capacities are integers,
s denotes the source, **t** denotes the sink in network flow problem.
m denotes number of edges in Network Flow Graph
V denotes set of vertices in original Graph
E denotes set of edges in original Graph
n denotes the total number of nodes in the original graph.
C = $\mathbf{\Sigma}_{e \text{ out of } s}\, c_e$. (Maximum possible flow out of the source).

We have a Graph G = (V,E). In this problem we have to calculate the maximum number of edge disjoint paths. A pair of paths is said to be edge disjoint if there is no common edge among the two paths in the pair. To solve this problem ford fulkerson algorithm can be used.

<u>Psuedocode:</u>

1. Assign source as the source of network flow graph
2. Assign destination as the sink of network flow graph
3. Construct the structure of network flow graph exactly as the original graph.
4. Now assign capacities to every edges 1 in the network flow graph.
5. Now run ford fulkerson on the network graph
6. The flow used at each node connected to the source will define a new path , this is because that 1 unit of flow can be carried at 1 path since capacity of all edges are 1.
7. Find the sum of flows in the residual graph having a back edge to the source.
8. Now, return the value of the **maximum number of edge disjoint paths**.

Since the problem is solved using ford fulkerson , so the time complexity of Max Flow algorithm will be O(mC). In this case since maximum number of nodes in set X can be n so complexity is **O(mn)**