CS 345: Algorithms II

Submitted By:
Anirudh Kumar (Y9088)
Chandra Prakash (Y9181)

Assignment 5

1 Solution of 1

1.1 Inference:

The edge (v, u) must be present in the shortest path that has been considered in the i^{th} iteration.
If (v, u) be a forward edge then for a non zero flow along this path, if (u, v) is not present in the graph then this is introduce with capacity $f(v, u)$ in the residual graph. If (v, u) is present as a backward edge the (u, v) is introduced with residual capacity $c(u, v) - f(u, v)$. If (v, u) is not in the path then (u, v) can't be introduced by Ford-Fulkerson algorithm.

1.2 Proof:

Considering the most general case, before the i^{th} there is a path $s \to a \to b \to t$ such that this is the shortest path, edge (b, a) is not present in the graph and there is a path $a \to v$.Then in the i^{th} iteration edge (b, a) is introduced as result of which there is path $p_i(v)$=$s \to b \to a \to v$ then the claim is, $d_i(v) < d_{i-1}(v)$. We also have $d_{i-1}(a) < d_{i-1}(b)$ since the shortest path has been selected.Therefore

$$d_{i-1}(v) > d_i(v) > d_{i-1}(b) > d_{i-1}(a)$$

Since $d_{i-1}(v) > d_{i-1}(a)$ therefore $d_i(a) \geq d_{i-1}(a)$ must satisfy.
But in this case $d_{i-1}(v) > d_{i-1}(v) \Rightarrow$ path length of the path $s \to b \to a$ is less than the path length of the path $s \to a$(at the begining of the i^{th} iteration). Therefore, $d_i(a) < d_{i-1}(a)$. Therefore contradiction.

1.3 Proof:

Using (2) our claim is, when (u, v) is reappears in the graph, there must be an edge (v, u) in the graph at the begining of the j^{th} iteration. Therefore, $d_{j-1}(u) = d_{j-1}(v) + 1$. Therefore, after the j^{th} iteration, using (3) our claim is $d_j(u) > d_{j-1}(v) \geq d_i(v) = d_i(u) + 1$. Therefore, $d_j(u) > d_i(u)$.

1.4 Proof:

During each iteration of FF algorithm, at least one edge disappears from the residual graph.Let (u, v) be an edge that is removed in the i^{th} iteration then, (u, v) can appear atmost n times. If this is not the case then using (4) there would be a shorted path $d_k(u) > n$ ie $d_k(u) = \infty$ since the total number of vertices is n. Therefore, edge (u, v) can't be reached. We have a total of m edges in the graph and each edge reappears atmost n times. Therefore, after mn iterations there is no path between s and t. Therefore the algorithm terminates at this point and hence the running complexity is $O(m^2 n)$.

2 Solution of 2

2.1 Pseudo Code:

function FINDREACHABLE(G, s, t, k)
  while there is an s-t path in the graph G **do**
    Find a simple s-t path in the graph G (no vertex repeated).
    Ping vertices on the path in order of a binary search to find the nearest node on this path which is unreachable from s. Let this node
    be v and the node immediately preceding it be u.
    Remove (u, v) from the graph G.
  end while
  Do a DFS of the remaining graph G from s to find all the vertices reachable from s.
end function

2.2 Proof:

The length of s-t path can be atmost n (no vertex is repeated). So if we do a binary search on this sequence of vertices we require atmost $\log(n)$ ping operations. When we remove an edge (u, v) from the graph, it is the one removed by the hacker too, because if there is an edge (u, v), there is no way to make v unreachable and keep u reachable without removing (u, v). Also since hacker has removed edges from min-size cut, (u, v) belongs to min-size cut.

The algorithm terminates when there is no s-t path remaining. Since we remove the edges from the min-size cut, after k iterations of while loop no edge from the min-size cut remains and consequently no s-t path remains in the graph. So the while loop runs k times and in each iteration we execute

a maximum of $\log(n)$ ping operations. So the algorithm executes $O(k\log n)$ ping operations.
Finally we do a DFS on the remaining graph to find all the reachable vertices.

2.3 Time Complexity

The time complexity of ping(v) ooperation has not been given so we represent it by P. The overall time complexity of the algorithm can be represented as:

$$\mathrm{O}(\mathrm{k}(\mathrm{Plog(n)} + \mathrm{n}) + |S|)$$

This includes the time required to find an s-t path in the graph using BFS i.e. O(n). Second term is due to DFS done of the graph to report the set of reachable vertices. $|S|$ is the size of the set returned by the algorithm. No node except those in $|S|$ are reachable from s, so DFS takes $\mathrm{O}(|S|)$ time.