CSC373 - Problem Set 4

Authors: Luke Bacchus, Naslin Rahman, Zhuoqian Li

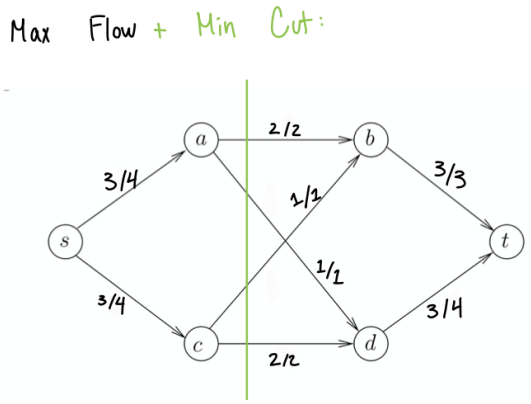
March 9, 2021

Question 1

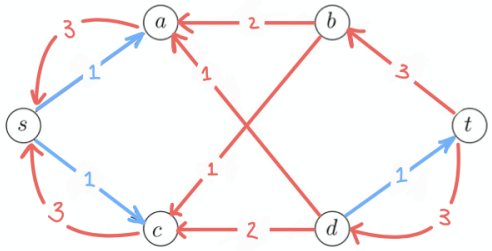
a.

Question 2

a. Graph for max flow and min cut:

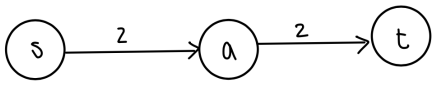


b. Residual Graph of Flow Network:



c. Set of bottleneck edges: (c,d),(a,d)

d. Simple flow network with no bottleneck edges



e. Find the maximum flow f of (G, s, t, c), and its corresponding resodial graph G_f.

To find a bottleneck edge, we want to find a path from s to t where all edges still have space left except for on edge.

Algorithmn: Starting from edge s, explore other nodes connected to this node with an edge from the node. (Ex: Explore node u if there is an edge (s,u)).

Choose one node to further explore at a time, and keep track of unexplored nodes in an array so you can return at a later time. Along with these unexplored nodes, keep track the number of edges along the path to that node who do not have an edge value in Gf, meaning that f(e) = c(e) in the max flow graph. These edges can be visualized by the blue arrows as seen in 2b.

If this number exceeds 2, then this is an invalid path and return to the most previous unexplored node. If the node t is reached, and the number of these kinds of edges is exactly one, then that edge is a bottleneck edge.

If we are on a path with a previously found bottleneck edge, we do not need to continue exploring. Continue until all possible paths are explored to get all possible bottleneck edges.

Correctness:

Let BE represent the be the set of bottleneck edges by the algorithm for (G,s,t,c).

Let BE* be the actual set of bottle neck edges for (G,s,t,c).

Assume $BE! = BE*$.

Case 1: $\exists(u,v) \in Es.t.(u,v) \in BE$ and $(u,v) \notin BE*$ This would mean that the algorithm found an edge (u,v) that is not actually a bottleneck edge.

If the algorithmn found (u,v) it would mean that there exists a path from s to t where for every e in the path except for (u,v), f(e) < c(e). And f((u,v)) = c((u,v)) on the max flow graph. Thus, increasing its capcity where c'((u,v)) > c((u,v)) would increase the value of a maximum flow of (G,s,t,c').

Thus, meaning that (u,v) is a bottleneck edge, which results in a contradiction.

Case 2: $\exists(u,v) \in Es.t.(u,v) \notin BE$ and $(u,v) \in BE*$ This would mean that there is a bottleneck edge that was not found by the algorithmn.

For this to be possible, it would mean that for all the paths from s to t that contain (u,v), a) (u,v) had an edge in G_f or b) another edge on that path did not have an edge in G_f.

In case a), this would mean that on those paths, f((u,v)) < c((u,v)) in the max flow graph. It would mean that raising the capacity of (u,v) would not increase the value of a maximum flow of (G,s,t,c') since there is a restriction elsewhere.

In case b), this would mean that for those paths, even if you raised c((u,v)), there is another edge limiting the flow of that path to t.

Both cases would contradict the definition of a bottleneck edge. Meaning that the algorithmn must've found (u,v)

Both cases result in a contradiction so our assumtion that $BE! = BE*$ is false.

Runtime: O((m+n)C) to run the Ford-Fulkerson Algorithmn, and O((m+n)C) for the worst case of the runtime of any possible path from s to t with max C iterations. Checking G_f is constant.

Thus, O((m+n)C).