Quiz #2
Algorithms I
600.463

Thursday, April 20, 12-1.15pm

Brief Solutions

Name Alan Class: 463.01/463.02 JHU 2017 S

Signature Date

Problem 1 (30 points)

Problem 1.1 (10 points)

In a few sentences explain what is a residual network.

CLRS 3rd

pg 715-716

Problem 1.2 (10 points)

Define what is the maximum flow problem.

CLRS 3rd

pg 709–710

Problem 1.3 (10 points)

Can you describe and draw an example to restate the max-flow min-cut theorem?

CLRS 3rd

pg 723-724

Problem #2 (60 points)

By now you should know how to construct a minimum spanning tree (MST) from a graph. However, sometimes we don't want a MST—perhaps we want to force our spanning tree to contain specific edges, regardless of whether or not those edges are optimal.

Suppose you are given an undirected graph $G = (V, E)$ with edge weights w_e, and an acyclic set of edges F, where F is a subset of E. (You may assume the edges are decorated such that "$inF(e)$" will return either *true* if e is one of the edges in F, or *false* otherwise.) Design an algorithm that runs in $O(|E| \log |E|)$ time and returns the lightest spanning tree T such that the edges of F are all contained in T. Analyze the running time of your algorithm and prove the correctness.

Algo 1: step 1: find the edges in F
use BFS and inF(e) to find all edges in F. Set the weight of each edge in F as $-\infty$.

step 2: run Kruskal's algo (ordered incresingly by weight)

correctness: $-\infty$ weights will make sure the edges in F will be included by kruskal's algo. By the correctness of Kruskal's, the special MST is returned.

time: $O(E \log E)$ since edges are sorted by weights.

Algo 2: step 1: find all edges in F by edge list traversal and inF(e).

step 2: Add edges of F to T first.

step 3: run Kruskal's on $G=(V, E\setminus F)$. and update T.

Correctness: F is an acyclic set and can be added to a tree. Add F to T first will guarantee the existence of edges of F in T. Then the kruskal's will keep adding next possible lightest edge that still maintains a tree. After the termination, the kruskal's returns the required spanning tree.

Time: same as kruskal's.

Problem #3 (60 points)

Let G = (V, E) be a weighted and directed graph with weight function w. G has exactly two negative-weight edges and no negative-weight cycles. Design an algorithm to find the shortest path weights $\delta(s, v)$ from a given $s \in V$ to all other vertices $v \in V$ in $o(|V||E|)$ time (little o).

Here o(VE) requires an algo works strictly better than Bellman-Ford.

Let's consider Dijkstra's algo. We all know Dijkstra's cannot work with negative weights

① remove two negative edges from G. Denote the two edges as (u_1, v_1) and (u_2, v_2). Denote the new graph as G'.

② run Dijkstra's from s, v_1, v_2, seperately (3 times) in G'.

③ for each vertex $v \in V$, $v \neq s$

$\delta(s,v)$ in G is the min of the following 5 cases. let $\delta'(s,v)$ denote sp in G'.

a. $\delta'(s,v)$

b. $\delta'(s,u_1) + w(u_1,v_1) + \delta'(v_1,v)$

c. $\delta'(s,u_2) + w(u_2,v_2) + \delta'(v_2,v)$

d. $\delta'(s,u_1) + w(u_1,v_1) + \delta'(v_1,u_2) + w(u_2,v_2) + \delta'(v_2,v)$

e. $\delta'(s,u_2) + w(u_2,v_2) + \delta'(v_2,u_1) + w(u_1,v_1) + \delta'(v_1,v)$

Find the min of above cases as the sp from s to v. and this applies to all v in V.

running time: same as Dijkstra's and in $O(VE)$

correctness: (eliminated) Considered all possible cases.