Intro to Algo HW #3 - Saaif Ahmed

Tuesday, February 25, 2020 7:28 PM

Problem 4.11

The algorithm is as follows:

We first create a 2-d matrix that is size $V \; * V$

Given that matrix is sorting as (i,j) where row *i* is the node we start at, and column *j* will hold the distance from *i* to *j* we run Dijkstra's algorithm on each i in the matrix.

From this we calculate the distance from node u to node v for all nodes in V, and then we do the same for v to u, or more specifically we do:

$\left(dist(u,v), dist(v,u)\right) \; \forall \; v,u \in V$.

We find the minimum pair amongst all the pairs we generate.

If all the distance pairs we generate are infinite, then the graph is acyclic, otherwise return the minimum pair

Analysis: Running Dijkstra's algorithm on an adjacency matrix takes $O(V^2 \log(V))$.
If we run it V times then the complexity is $O(V^3 \log(V))$.
The next step of our algorithm calculates all the pairs possible in our matrix which takes $O(V^2)$.
So because $V^3 > V^2 > \log(V)$, in Big-O notation the algorithm is $O(V^3)$

Intro to Algo HW #3

Tuesday, February 25, 2020 7:42 PM

Problem 4.13

a)

The algorithm is as follows:

Run a modified DFS search starting from s. The modification to be made is that if the DFS encounters and edge whose weight $l_e > L$ then you remove that edge from the set E and you do not $explore()$ down that path.

If you reach t , run the post order and return the nodes as you come back up. Otherwise there is no path.

Analysis: DFS is linear time and it is all that we run so it is $O(V+E)$

b)

We must find the shortest path from s to t with a restriction on edge weights. Because these are positve edge weights we can use Dijkstra's algorithm.

We modify Dijkstra's by setting each node to the minimum fuel weight rather than the distance. We do this by setting $dist(v) = \max\big(dist(u,v), l(u,v)\big)\ \forall\ v,u \in V$

When we now run Dijkstra's on the graph it output the minimum fuel weight instead of the minimum distance, which gives us out fuel capacity margin.

Analysis: If we run Dijkstra's with a priority queue it is $O((V+E)\log(V))$ by definition thus the algorithm meets the runtime requirement

Intro to Algo HW #3

Tuesday, February 25, 2020 8:00 PM

Problem 4.20

The algorithm is as follows.

Create an array size of E' that will store tuples.

Run Dijkstra's starting at s to every node reachable.

Run Dijkstra's starting at t to every node reachable.

Then begin adding in the edges from E'. For each receiving node of the edge (1 from s , 1 from t) sum the value of $Dijkstra's\ from\ (s) + l_{e'} + Dijkstra's\ from\ (t)$ and store that into the array.

Return the minimum of the array that contains an edge from E'.

Analysis: We run Dijkstra's twice, and also work with the set E' twice. Thus the overal run time is $2 * O\big((V+E)\log(V)\big) + 2 * O(E')$ which in Big-O is $O((V+E)\log V)$

Intro to Algo HW #3

Tuesday, February 25, 2020 8:17 PM

Problem 5.5

a)

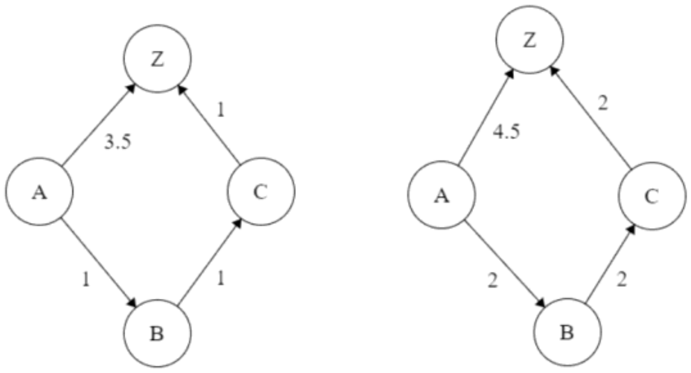
No the MST does not change. If every edge weight is increased by $+1$ then the MST generated by Kruskal's algorithm for example will not change. This is because if Kruskal's algorithm is deciding between 2 edges a, b and we know that $a < b$, then Kruskal's will take a.

Now we observe what happens if we increase weights by $+1$.

$a + 1 < b + 1 \rightarrow a < b$ (subtract 1 from both sides).
Kruskal's still takes edge a. This holds for all edges $e \in E$.

b)

Yes it can change. We show an example below.



Initially the shortest path from $A \rightarrow Z$ is $(A \rightarrow B \rightarrow C \rightarrow Z)$
However after we increase the edge weights the shortest path then becomes $A \rightarrow Z$

Intro to Algo HW #3

Tuesday, February 25, 2020 8:26 PM

Problem 5.6

Proof by Contradiction:

Assume there exists a graph G with multiple MSTs $M1, M2$ …
By definition, each MST should have a different set of edges to be unique, but must have the same weight.
There will be an edge e that is the smallest cost edge in one and only one of the MST.
However this is a contradiction. The construction of the MST is greedy and takes the smallest edge weight. If e is taken in one MST it must be taken in all the others. But that would mean that the MSTs are not unique.
This is the contradiction thus the original statement is true. There is only 1 unique MST in this scenario.