HW3

Xinhao Luo

Tuesday 12th May, 2020

1 DPV Problem 4.11

Algorithm

Step 1 Initialize an adjacent matrix for each element in the graph

Step 2 for each node n in the graph, run a Dijsktra Shortest Path to each node k in store distance to list at (n, k) (Initialize default distance to ∞)

Step 3 Initialize a temp variable min_dist to ∞

Step 4 Loop through each element in the matrix above the diagonal, and sum with its corresponded symmetric value of the matrix diagonal, and update min_dist if the sum is smaller than current min_dist

Step 5 Return no if the min_dist is ∞, else min_dist is the required target value

Runtime

Step 1 initialize size will be $O(v^2)$, where v is the number of element in the graph

Step 2 Dijsktra Algorithm will take $O(v^2)$, as describe in lecture, and we will need to loop through each node with the algorithm, so we have $O(v^3)$

Step 3 O(1)

Step 4 Loop will be $O(v^2)$

Step 5 O(1)

Conclusion The largest component of the time complexity is $O(v^3)$, so the overall time complexity of the algorithm is $O(v^3)$

2 DPV Problem 4.13

a) **Algorithm**

Step 1 For all edge e, disconnect all path that has length larger than L, the limitation of the car capacity

Step 2 Start from node s, do an DFS, and stop if node t is reached, and return true

Step 3 Return false if DFS is finish and node t is not found in when search

Runtime

Step 1 $O(|E|)$ for iterating through the graph and disconnect them

Step 2 $O(|E| + |V|)$ for DFS

Step 3 O(1)

Conclusion Overall time complexity is $O(|E| + |V|)$, a linear solution

b) **Algorithm**

Step 1 Start from point s, do an Dijsktra from node s, initialize distance as -1.
Change the update algorithm: Each time we decrease the distance from node u to v, we keep the $fuel(v) = max(fuel(u), len(u, v))$.

Step 2 node t's distance will be the minimum L.

Runtime

Step 1 $O((|E|+|V|) \times log(|V|))$ for Dijsktra Algorithm, using set implementation for findMinimum.

Step 2 O(1) for return value

Conclusion Overall time complexity is $O((|E| + |V|) \times log(|V|))$

3 DPV Problem 4.20

Algorithm

Step 1 Do Dijsktra from node s and t, save for later usage.

Step 2 Set result as the distance shortest distance from s to t; Node a = s, b = t as the final result

Step 3 For each edge e' from available edges, add the edge(u, v) to the graph. Calculate distance from $total\_distance = dist(s, u) + len(edge) + dist(v, t)$, while $dist(v, t)$ is the same as $dist(t, v)$ as the it is an undirected graph. If the total_distance is smaller than result, let $result = total\_distance$, and node $a = u, b = v$

Step 4 After the iteration, check node a, b. If a == s, b == t then there is no better solution, otherwise the path should be set between node a, b

Runtime

Step 1 Dijesktra's time complexity is $O((|V| + |E|) * log(|V|))$, using Set implementation

Step 2 O(1) for initialize value

Step 3 $O(|E'|)$ for the for loop

Step 4 $O(1)$ for return value

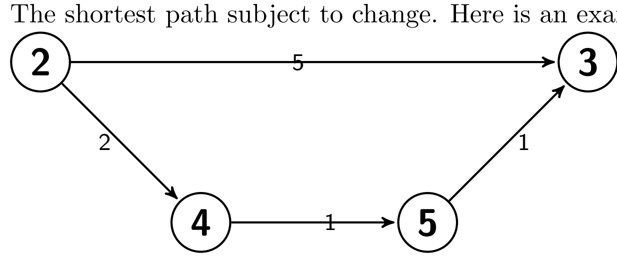
Conclusion The overal time complextity is $O((|V| + |E|) * log(|V|))$

4 DPV Problem 5.5

a) The minimum spanning tree will not change.
Prove by Contradiction

- Assume we have mst2, after +1 for all edge, and mst1 for the original graph; and mst2 is different from mst1.
- Define cost() as the sum of the total weighted edge of the graph.
- We have $cost(mst1) + (|v| - 1) > cost(mst2)$
- However, based on equation above, we also have: $cost(mst1) > cost(mst2) - (|v| - 1)$. This means that before the +1 for all edge, mst2 is a better solution for the graph than mst1, which is **fishy**
- the tree will not change.

b) The shortest path subject to change. Here is an example



5 DPV Problem 5.6

Prove by Contraction

1. Assume there are two different mst(mst1, mst2) for the distinct tree
2. Since mst1 is different from mst2, we know there must be at least one edge different between mst1 and mst2
3. Define cost() collect the sum of the mst
4. since each edge in the graph distinct, we have cost(mst1) != cost(mst2)
5. Then it must be that mst1 is either better or worse than mst2, which makes one of them not qualify for being an mst. This conclusion is **fishy**
6. There is only one mst for undirected weighted graph which all edges are unique