Problem Set 7

Due dates: Electronic submission of the pdf file of this homework is due on **10/29/2021 before 11:59pm** on canvas. The homework must be typeset with LaTeX to receive any credit. All answers must be formulated in your own words.

Watch out for additional material that will appear on Thursday! Deadline is on Friday, as usual.
Name: [redacted]

Resources. (All people, books, articles, web pages, etc. that have been consulted when producing your answers to this homework)

On my honor, as an Aggie, I have neither given nor received any unauthorized aid on any portion of the academic work included in this assignment. Furthermore, I have disclosed all resources (people, books, web sites, etc.) that have been used to prepare this homework.

Signature: [redacted]

Read chapters 22 and 24 in our textbook before attempting to answer these questions.

Problem 1 (20 points)**.** Solve Exercise 22.4-3 on page 615 of our textbook [CLRS].

Solution. For problem 22.4-3, we must determine an algorithm that can determine if a given undirected graph contains a cycle, where the time complexity is $O(V)$. In an undirected graph, if there are no back edges, which is essentially the connection of vertex u to v in edges (u, v), then there is no cycle. Simply running DFS to find a back edge will give us a time complexity of the worst case. Based on case by case situation, we would still use DFS but it would end the algorithm as soon as there's an edge going to a vertex thats already been visited. Given the graph is acyclic, the time complexity would be $O(V)$, by simply running DFS, because if there is a back edge it will be recognized before seeing all $|V|$ distinct edges, and this is a forest so $|E| \leq |V| - 1$. If the algorithm ends early, we know that it went to an already visited vertex and it have most likely created a forest in which case the time complexity is $O(|V|)$.

Problem 2 (20 points)**.** Solve Exercise 24.1-3 on page 654 of our textbook [CLRS].

Solution. For problem 24.1-3, we must find a change to the Bellman-Ford algorithm that allows it to terminate in m+1 passes. A change that we would make in the algorithm is to take a copy of v values and check if they were updated or not, if they were not updated, then the algorithm would simply stop. We can confirm that every vertex has reached a shortest path weight in d, because if no d values changes after m iterations, they will also not change in m+1 iterations. Since we are not given m, it is more efficient to simply terminate when there are no more changes being made.

Problem 3 (20 points)**.** Solve Exercise 24.3-3 on page 663 of our textbook [CLRS]. [Use the version of Dijkstra's algorithm from the textbook]

Solution. For problem 24.3-3, we must determine if the algorithm is still correct if line 4 is changed to $|Q| > 1$. Yes, the algorithm will still work correctly after being changed. Given v is a remaining vertex that needs to be considered, we know that the shortest path weight d, contains the value that is similar to the largest value of the other given vertices. Considering the edge weights, we know that their d value would be just as large as the d values for other vertices, for the minimum case. From this we can say, that the d values would not change for any of the vertices, and there will still be a shortest path despite the loop executing $|V| - 1$ time instead of $|V|$ times.

Problem 4 (40 points)**.** Help Professor Charlie Eppes find the most likely escape routes of thieves that robbed a bookstore on Texas Avenue in College Station. The map will be published on Thursday evening. In preparation, you might want to implement Dijkstra's single-source shortest path algorithm, so that you can join the manhunt on Thursday evening. [Edge weight 1 means very desirable street, weight 2 means less desirable street]

Solution. This algorithm is a way to find a single shortest path, Dijkstra's algorithm implements the weights of edges, to find the shortest path possible, between the starting node and all the other nodes surrounding it. For the purpose of this question, we had to find the shortest path to 6,8,9,15,16, and 22, given the source node is 1. In order to achieve this, we could list out all possible paths to get to those 6 possible exits and note their weights. Whichever has the smallest weight would be the shortest path. Here are the paths:
1. For 6: 1, 2, 3, 4, 5, 6 where the weight is 6
2. For 8: 1, 2 ,3 ,8 where the weight is 4
3. For 9: 1, 11 ,10 ,9 where the weight is 3
4. For 15: 1, 11, 17, 16, 14, 15 where the weight is 6
5. For 16: 1, 11, 17, 16 where the weight is 4
6. For 22: 1, 2, 21, 22, where the weight is 4

Note that in some cases there can be more than one path presented that has the same weight, in which, any option is still the shortest path.

Discussions on ecampus are always encouraged, especially to clarify concepts that were introduced in the lecture. However, discussions of homework problems on ecampus should not contain spoilers. It is okay to ask for clarifications concerning homework questions if needed. Make sure that you write the solutions in your own words.

Checklist:
- ☐ Did you add your name?
- ☐ Did you disclose all resources that you have used?
  (This includes all people, books, websites, etc. that you have consulted)
- ☐ Did you sign that you followed the Aggie honor code?
- ☐ Did you solve all problems?
- ☐ Did you typeset your answers entirely in LaTeX?
- ☐ Did you submit the pdf file of your homework?