CS 3510 – Assignment 4

Due `Friday, July 2, 2022` at `11:59pm` on Canvas

Instructor: Shahrokh Shahi

- Please type your answers (LaTeX is highly recommended) and upload a single PDF file named `<Your-GT-Account>.pdf`, e.g., GBurdell3.pdf, including all your answers. You can submit multiple times. Canvas keeps track of the submissions and append a version number when you re-submit. We always grade your most recent submissions.
- Please read the policies, and do not forget to acknowledge your collaborators and cite your references.
- If you do not understand a question, please ask on Piazza or come to office hours well ahead of the due date.
- It is recommended to start reviewing the course material by reading the lecture slides and reviewing the demo codes. Then, the suggested readings from textbooks and solving the practice problems can provide the additional preparation for solving the homework problems. Please note, for the textbook readings, you do not need to cover the topics which have not been covered in the lectures.

Suggested Reading

	CLRS	KT
Section(s)	Chapter 22-23	Chapter 3, 4

Suggested Practice Problems

	CLRS	KT
Practice problems	Exercise: 22.1-(1-6), 22.2-(1,2,4,5), 22.3-(2,3,7), 22.4-(1,5), 22.5-(2,3) 23.1-(1,2,5), 23.2-(1,2) Problems: 22-4, 23-3	Exercise: chapter 3: 1,2,6,10 chapter 4: 2, 8, 9, 10

Additional reading and problems:
– DPV (Chapter 3, 5)

1 Graph Traversal (20 pts)

Let $G = (V, E)$ be an undirected graph with the additional property that every edge also has a color, either red or blue. Let u and v be distinct vertices in $G = (V, E)$.

(a) (10 pts) Design an efficient (linear) algorithm that decides whether or not there exists a path from u to v such that the path contains only red edges. Justify the correctness of your algorithm and discuss the running time.

(b) (10 pts) Design an efficient (linear) algorithm that decides whether or not there exists a path from u to v such that within the path, all blue edges appear after all red edges .Justify the correctness of your algorithm and discuss the running time.

Solutions

(a) Design: If we want to find a path containing only red edges, we simply need to delete all the blue edges in the graph. Then, we can run a traversal algorithm, i.e., DFS or BFS, from u on the resulting graph to check if v is marked visited. If not, we return False.

Correctness: The algorithm is correct because a path between u and v that contains only red edges exists only if, after deleting the blue edges, u and v are in the same connected component.

Running time: Deleting all the blue edges takes $O(E)$ time and the traversal takes $O(V + E)$ time. Therefore, the overall running time is $O(V + E)$.

(b) Design: We begin by deleting all the blue edges from the graph and running a traversal, BFS or DFS, from u on the resulting graph to find the set of vertices reachable from u with only red edges. We then restore all the blue edges and delete all the red edges. Next, we run the traversal from v on the resulting graph to find the set of vertices reachable from v with only blue edges. Now, we check to see if there is a vertex that is both reachable from u using only red edges and reachable from v using only blue edges and return True if this is happens, and return False otherwise.

Correctness: A path between u and v where all the blue edges appear after all the red edges only exists if we can find a vertex such that all edges up to it are red and all edges after it are blue, which is what our design search for.

Running time: The total time taken to modify edges is $O(E)$. The two traversal operations take $O(V + E)$. The time taken to iterate through the two sets of visited vertices afterwards is $O(V)$. Since every step is linear, the running time of the algorithm is thus $O(V + E)$.

2 Graph Traversal (10 pts)

For each of the two following statements, decide whether it is **T**rue or **F**alse. If it is true, provide a short explanation and if it is false, give a counterexample.

(a) (5 pts) Given a tree $T = (V, E_T)$, we run both BFS and DFS starting from the same node s.
`True or False?` The resulting BFS tree and DFS tree are the same.

(b) (5 pts) Given a graph $G = (V, E)$, we run BFS and DFS starting from the same node s, and obtain the same traversal tree by both algorithms.
`True or False?` The graph G is a tree.

Solutions

(5 pts= 2 (T/F) + 3 (reasoning))

(a) True As the original graph is tree, there cannot be any non-tree edges. In other words, both DFS and BFS must use the same edges to traverse the graph, so the resulted trees will be identical.

(b) True If graph G is not a tree, then it must contain a cycle. The BFS algorithm split the cycle into at least two branches as it traverses the cycle. However, DFS will traverse all nodes of the cycle on the same path (branch). Therefore, if the traversal trees are identical, it means the graph must be connected and cannot have any cycle. Thus, it must be a tree.

3 Minimum Spanning Tree (10 pts)

Suppose $G = (V, E)$ is a weighted connected graph and let $e = (u, v)$ be a minimum-weight edge in the given graph G. Show that $e = (u, v)$ belongs to some minimum spanning tree of G.

Solutions

Let T be an MST of $G = (V, E)$ and $e \in E$ is the minimum-weight edge. For the sake of contradiction, assume e is not in T, and let's see what happens. Let the path from u to v in T include the edge (v, x) (why? because (u, v) is not part of the tree T, but T is still a tree and it must be connected; therefore, it must be some other path from u to v in the form of $u \to \ldots \to x \to v$). Now we can construct a tree T' by removing edge (v, x) from T and adding (u, v). Since T is a spanning tree, T' is also a spanning tree. However, because we replaced (v, x) with (u, v) and the $w(u, v) \leq w(v, x)$, then T' must also be an MST. Therefore, the minimum-weight edge $e = (u, v)$ must be in some MST of G.

4 Minimum Spanning Tree (10 pts)

Let $G = (V, E)$ be an undirected, connected, weighted graph, and let F be a subgraph of G such that it is a forest (i.e. F includes one or more trees and does not contain any cycles). Design an efficient algorithm to find a spanning tree that contains all the edges of F, and that has minimum cost among all spanning trees containing F.

Solution

Algorithm Design:

- Initialize an MST T with all the edges from F. ($T = F$)
- Grow the MST T using Kruskal's algorithm by using sorted edges from the set $E_G - E_F$. At each step, a "safe" edge, i.e., a least-weight edge connecting two distinct components, will be added to T until all nodes are covered.

Correctness: Clearly the output of our design contains F. Since we use the same approach as for Kruskal's algorithm, T will be a spanning tree with minimal weight guaranteed by the correctness of Kruskal's algorithm.

Running time: In the worst case, we need to run Kruskal's algorithm on the entire graph, when $F = \Phi$. Thus, the runtime is also $O(|E| \log(|V|))$.