Assignment 9

Algorithms, Spring 2023

Honor code: *Work on this assignment alone or with one partner. Between different teams, collaboration is at level 1 [verbal collaboration only]. There are lots of resources online, such as animations, visualizations, practice problems, videos, and solutions— which you are encouraged to explore to deepen your understanding. However, you must be careful not to search for the specific problems in the assignment with the intent of getting hints for the solution. Searching for the assignment problems on the internet violates academic honesty for this class.*

---

1. The *transpose* of a digraph $G = (V, E)$ is the graph $G^T = (V, E^T)$, where $E^T = \{(v, u) \in V \times V | (u, v) \in E\}$. In other words, G^T is G with all edges reversed.

   (a) Describe efficient algorithms for computing G^T from G. Assume the graph is given as an adjacency list, and you want to compute the adjacency list of G^T. Analyze the running time of your algorithm.

   (b) Same problem as above: What if the graph is given as an adjacency-matrix, and you want to compute the adjacency matrix of G^T?

   We expect: pseudocode, justification and run time analysis.

2. In a directed graph, two vertices u and v are said to be in the same *strongly connected component (SCC)* if u can reach v and v can reach u.

   (a) Describe a linear time algorithm for computing the *strong component* containing a given vertex v.

   (b) On the basis of that algorithm, describe a simple algorithm for computing the strong components of a directed graph G.

   (c) Describe an algorithm which, given a directed graph G and two arbitrary vertices u, v, determines whether u and v are in the same SCC.

   We expect: pseudocode, justification and run time analysis.

   The following two problems concern bipartite graphs. An undirected graph is called *bipartite* if the set of vertices can be partitioned into two disjoint sets, such that all edges in G have one endpoint in each set (put differently, such that no edge in G has both endpoints in the same set).

3. We would like to determine whether a given undirected graph is bipartite. Consider the algorithm below.

Algorithm: Start with two empty groups A and B. Choose vertices in the graph in an arbitrary order. For each vertex chosen:

- If the vertex has no edges to any vertices in A, assign it to A.
- Otherwise, if the vertex has no edges to any vertices in B, assign it to B.
- Otherwise – that is, the vertex has edges to vertices in both A and B – declare that the graph is not bipartite, and stop.

If all vertices are assigned without ever declaring that the graph is not bipartite, declare that the graph is bipartite.

Show that the algorithm is **incorrect** by finding a single counterexample.

We expect: A counter example, and sufficient information to make it clear that it is a counterexample. Please try to make your counterexample as simple as possible.

4. Describe an algorithm which, given an undirected graph $G = (V, E)$, determines whether it is bipartite.

   Hint: Your algorithm should use that a graph is bipartite if it is 2-colorable (and the other way around). A graph G is called *k-colorable* if it is possible that the vertices can be assigned one of k different colors, in such a way that no edge connects vertices of the same color. Use BFS and try to 2-color the vertices. If you succeed, the graph is 2-colorable and therefore bipartite. If you don't, the graph is not 2-colorable and therefore not bipartite.

   We expect: pseudocode, justification and run time analysis.