ICS 311: Algorithms Spring 2025

Problem Set 14

Kyle Berney *Due: Friday, May 2, 2025 at 4pm*

You may discuss the problems with your classmates, however **you must write up the solutions on your own** and **list the names** of every person with whom you discussed each problem.

Start **every** problem on a separate page, with the exception that Problems 2 can start on the same page as Problem 1 (Peer credit assignment).

1 Peer Credit Assignment (1 point extra credit for replying)

Please list the names of the other members of your peer group for this week and the number of extra credit points you think they deserve for their participation in group work.

- You have a total of 60 points to allocate across all of your peers.
- You can distribute the points equally, give them all to one person, or do something in between.
- You need not allocate all the points available to you.
- ***You cannot allocate any points to yourself!*** Points allocated to yourself will not be recorded.

2 Verifying Longest Path (40 pts)

A *simple path* in a graph G is a path that does not visit any vertex more than once. The problem of finding the *longest simple path* asks for the length of the longest simple path in a given graph G. The decision version of the longest simple path problem asks, given a graph G and an integer k, if there is a simple path in G that contains at least k edges. Consider the following language $\textsc{Longest-Path} = \{\langle G, k\rangle :$ there is a path in G that contains least k edges$\}$. Show that $\textsc{Longest-Path} \in NP$ as follows:

(a) (16 pts) Given an instance (G, k) of a problem that contains a path of length at least k, describe what the certificate would look like.

(b) (16 pts) Give a polynomial-time verification algorithm that would use the above certificate to verify that the instance indeed has a longest simple path.

Now that we know that $\textsc{Longest-Path} \in NP$, there is one more step remaining to prove that the longest simple path problem is NP-complete.

(c) (8 pts) What would you have to do to prove that the longest simple path problem is NP-complete. Simply identify what you would have to do, you don't actually have to do it for this homework.

3 Approximation Algorithm for Maximum Matching (60 pts)

Recall that for an undirected graph G, a ***matching*** is a set of edges such that no two edges in the set are incident on the same vertex. Last week we saw how to find a *maximum matching in a bipartite graph*. For general graphs, i.e., graphs that are not required to be bipartite, the problem of finding ***maximum matching*** can also be solved in polynomial time. In this problem, we will look at the linear time approximation algorithm for finding maximum matching in general undirected graphs, which is faster than the best know exact algorithm.

(a) (5 pts) A ***maximal matching*** (notice the difference in the ending) is a matching, such that no edge can be added to the graph without violating the matching property. Show that a maximal matching need not be a maximum matching by showing an undirected graph G and a maximal matching M in G that is not a maximum matching. *(Hint: You can find such a graph with only four vertices.)*

(b) (15 pts) Consider an undirected graph $G = (V, E)$. Design an $O(V + E)$-time ***greedy*** algorithm to find a maximal matching in G. Write down the pseudocode, explain why it finds the maximal matching correctly, and analyze its running time. ***All three parts must be present to receive any credit.***

In the rest of the problem you will show that the greedy algorithm from part (b) is a 2-approximation of maximum matching.

(c) (10 pts) Show that the size of a maximum matching in G is a lower bound on the size of any vertex cover for G.

(d) (10 pts) Consider a maximal matching M in $G = (V, E)$. Let $T = \{v \in V : \text{ some edge in } M \text{ is incident on } v\}$. What can you say about the subgraph of G induced by the vertices of G that are not in T? (A subgraph of $G = (V, E)$ induced by a subset of vertices $V' \subseteq V$ is subgraph $G' = (V', E')$ consisting of vertices V' and edges $E' \subseteq E$, such that for each edge $(u, v) \in E'$, both $u \in V'$ and $v \in V'$.)

(e) (10 pts) Conclude from part (d) that $2|M|$ is the size of a vertex cover for G.

(f) (10 pts) Using parts (c) and (e), prove that the greedy algorithm in part (b) is a 2-approximation algorithm for maximum matching.