Homework 4

COMP221 Spring 2024 - Suhas Arehalli

Complete the problems below. This assignment is out of **50** points, and the point values of each question are listed after the question's instructions. Note that point values are roughly inversely correlated with expected difficulty with secondary consideration for the amount of work and centrality of the material associated with them.

- Write up your solutions to all parts of this homework using some typesetting software (LaTeX/Overleaf is recommended, but any typeset PDF is acceptable).
- You'll submit this assignment through Github Classroom (link on Moodle), with your written solutions in a pdf named "[LASTNAME1]_[LASTNAME2]_Homework1.pdf".
- **The due date for this assignment will be posted on the course website.**
- **You may submit assignments in pairs.** Put both peoples' names at the top of your submission.
  - **Both people are expected to have contributed to all parts of the assignment** – work together, and make sure you and your partner are both convinced of your solutions.
  - If you discussed problems with other students, list their names at the top of your assignment (there will be no penalty for this — it's encouraged!).
  - However, don't look at other group's written solutions/code, or share your written solutions/code with other groups. **In addition, do not search around the internet for solutions to these problems.** Seek help from your partner and course staff, the textbook, and course materials. Consult the syllabus for more details on academic integrity policies.

Note that we may not have covered all of the topics involved in this assignment at the point of release. This is to give you time to complete the parts you can as early as you can. I recommend you make note of parts we haven't covered yet so you can come back to them.

1 Algorithmic Design & Analysis *(50pts)*

1. **Degree Constrained Spanning Trees** *18pts*

   The low-degree spanning tree problem (Low-Degree-Span) is a decision problem that asks whether, for a graph $G = (V, E)$ and $k \in \mathbb{Z}_{\geq 0}$, whether there exists a spanning tree where each vertex in the tree has degree $\leq k$.

(a) (Skiena 11-12a) Provide (with pseudocode) a many-one/Karp reduction showing that this problem is NP-hard. (*10pts*)
****Hint**:** Consider the Hamiltonian Path Problem from the textbook.

(b) (Skiena 11-12b) Consider the *high-degree spanning tree problem* (HIGH-DEGREE-SPAN) which asks, for graph $G = (V, E)$ and $k \in \mathbb{Z}_{\geq 0}$, if G contains a spanning tree which contains a vertex with degree $\geq k$ (for clarification, count only edges within the spanning tree when computing degree here!). Provide pseudocode for a polynomial-time algorithm to solve this problem. (*8pts*)
****Hint****: Consider methods we've seen before that have you construct spanning trees (not necessarily minimal!) in a graph!

2. **Approximate Vertex Cover** *32pts*

Recall that the (minimum) vertex cover problem asks you to construct, given a graph $G = (V, E)$, $C \subseteq V$ such that every edges $e \in E$ is incident to a vertex in C. This transforms into a decision problem where, given that graph and an integer k, you must determine whether a vertex cover of size $\leq k$ exists on this graph.

(a) Provide an algorithm to *verify* a solution to vertex cover (decision) problem. You may assume that the *certificate* $C \subseteq V$ is a purported vertex cover. Find an *efficient* algorithm to determine whether the given certificate verifies that the answer to the vertex cover problem from given G and k is true. (*8pts*)

(b) Suppose G is a tree. Provide a linear time algorithm ($\Theta(|V| + |E|)$) for ~~constructing a~~ determining the size of the minimum vertex cover. (*6pts*)
****Hint****: Remember that edges in a tree connect a parent and a child, so a vertex cover must select one of them. Then consider the recursive structure of a tree (A tree rooted at vertex r is r, edges that connect to it's children, and subtrees rooted at those children). From this, try a dynamic programming approach!

(c) Prove that the algorithm in the previous part is correct. *6pts*

(d) For every edge $e \in E$, by definition a vertex cover must contain at least one of the vertices incident to that edge. Write pseudocode for an algorithm that incrementally builds a vertex cover by repeatedly selecting edges that are not yet covered and then adding *both* vertices along those edges to the vertex cover. (*4pt*)

(e) Provide a big-O time complexity for the algorithm in the previous part in terms of $|V|$. (*4pt*)

(f) Prove that this is a *2-Approximation* of the vertex cover problem. That is, suppose your algorithm constructs a set cover C, and let C^* is the minimum set cover; Prove that $|C| \leq 2|C^*|$. (*3pt*)

(g) Consider an alternative strategy where you repeatedly select the *highest-degree vertex* remaining in the graph. That is, repeatedly add the highest-degree vertex v to a vertex cover C and then remove all edges incident to v from E until no edges remain in E. Prove, by counterexample, that this solution is *not* a *2-Approximation*. You may assume that the tie-breaks in your greedy algorithm break in your favor (i.e., construct the *worse* approximation) (*1pt*)
****Hint**:** Consider a bipartite graph with one of the two parts containing vertices with identical degree.