ECE 374 B ✧ Spring 2024
Homework 8

- **Submit your solutions electronically on the course Gradescope site as PDF files.** If you plan to typeset your solutions, please use the LaTeX solution template on the course web site. If you must submit scanned handwritten solutions, please use a black pen on blank white paper and a high-quality scanner app (or an actual scanner, not just a phone camera).

☞ Some important course policies ☜

- **You may use any source at your disposal**—paper, electronic, or human—but you ***must*** cite ***every*** source that you use, and you ***must*** write everything yourself in your own words. See the academic integrity policies on the course web site for more details.
- **Avoid the Three Deadly Sins!** Any homework or exam solution that breaks any of the following rules will be given an ***automatic zero***, unless the solution is otherwise perfect. Yes, we really mean it. We're not trying to be scary or petty (Honest!), but we do want to break a few common bad habits that seriously impede mastery of the course material.
  - Always give complete solutions, not just examples.
  - Always declare all your variables, in English. In particular, always describe the specific problem your algorithm is supposed to solve.
  - Never use weak induction.

See the course web site for more information.

If you have any questions about these policies,
please don't hesitate to ask in class, in office hours, or on Piazza.

1. The traveling salesman problem can be defined in two ways:

   - The Traveling Salesman Problem
     - Input: A weighted graph G
     - Output: Which tour $(v_1, v_2, \ldots, v_n)$ minimizes $\sum_{i=1}^{n-1}(d[v_i, v_i+1]) + d[v_n, v_1]$
   - The Traveling Salesman *Decision* Problem
     - Input: A weighted graph G and an integer k
     - Output: Does there exist and TSP tour with cost $\leq k$

   Suppose we are given an algorithm that can solve the traveling salesman decision problem in (say) linear time. Give an efficient algorithm to find the actual TSP tour by making a polynomial number of calls to this subroutine.

2. A **Hamiltonian cycle** in a graph is a cycle that visits every vertex exactly once. A **Hamiltonian path** in a graph is a path that visits every vertex exactly once, but it need not be a cycle (the last vertex in the path may not be adjacent to the first vertex in the path.)

   Consider the following three problems:

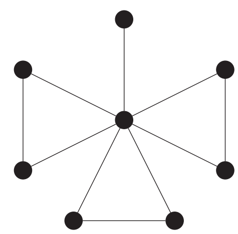
   - *Directed Hamiltonian Cycle* problem: checks whether a Hamiltonian cycle exists in a *directed* graph,
   - *Undirected Hamiltonian Cycle* problem: checks whether a Hamiltonian cycle exists in an *undirected* graph.
   - *Undirected Hamiltonian Path* problem: checks whether a Hamiltonian path exists in an *undirected* graph.

   (a) Give a polynomial time reduction from the *directed* Hamiltonian cycle problem to the *undirected* Hamiltonian cycle problem.

   (b) Give a polynomial time reduction from the *undirected* Hamiltonian Cycle to *directed* Hamiltonian cycle.

   (c) Give a polynomial-time reduction from undirected Hamiltonian *Path* to undirected Hamiltonian *Cycle*.

3. The low-degree spanning tree problem is as follows. Given a graph G and an integer k, does G contain a spanning tree such that all vertices in the tree have degree at most k (obviously, only tree edges count towards the degree)? For example, in the following graph, there is no spanning tree such that all vertices have a degree at most three.



   (a) Prove that the low-degree spanning tree problem is NP-hard with a reduction from Hamiltonian path.

   (b) Now consider the high-degree spanning tree problem, which is as follows. Given a graph G and an integer k, does G contain a spanning tree whose highest degree vertex is at least k? In the previous example, there exists a spanning tree with a highest degree of 7. Give an efficient algorithm to solve the high-degree spanning tree problem, and an analysis of its time complexity.

4. Some SAT reductions:

   (a) Stingy SAT is the following problem: given a set of clauses (each a disjunction of literals) and an integer k, find a satisfying assignment in which at most k variables are true, if such an assignment exists. Prove that stingy SAT is NP-hard.

   (b) The Double SAT problem asks whether a given satisfiability problem has at least two different satisfying assignments. For example, the problem $\{\{v_1, v_2\}, \{\overline{v_1}, v_2\}, \{\overline{v_1}, \overline{v_2}\}\}$ is satisfiable, but has only one solution ($v_1 = F, v_2 = T$). In contrast, $\{\{v_1, v_2\}, \{\overline{v_1}, \overline{v_2}\}\}$ has exactly two solutions. Show that Double-SAT is NP-hard.