ECE 374 B - Fall 2023

# 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). We will mark difficult to read solutions as incorrect and move on.
- **Every homework problem must be done *individually*.** Each problem needs to be submitted to Gradescope before 6AM of the due data which can be found on the course website: https://ecealgo.com/homeworks.html.
- For nearly every problem, **we have covered all the requisite knowledge required to complete a homework assignment prior to the "assigned" date.** This means that there is no reason not to begin a homework assignment as soon as it is assigned. Starting a problem the night before it is due a recipe for failure.

## Policies to keep in mind

- **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.
- **Being able to clearly and concisely explain your solution is a part of the grade you will receive.** Before submitting a solution ask yourself, if you were reading the solution without having seen it before, would you be able to understand it within two minutes? If not, you need to edit. Images and flow-charts are very useful for concisely explain difficult concepts.

**See the course web site (https://ecealgo.com) 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. For any integer $k$, the problem $k$SAT is defined as follows:

   - INPUT: A boolean formula $\Phi$ in conjunctive normal form, with exactly $k$ distinct literals in each clause.
   - OUTPUT: TRUE if $\Phi$ has a satisfying assignment, and FALSE otherwise.

   (a) Describe and analyze a polynomial-time reduction from **2**SAT to **3**SAT, and prove your reduction is correct.

   (b) Describe and analyze a polynomial-time algorithm for **2**SAT. *[Hint: This problem is strongly connected to topics earlier in the semester.]*

   (c) Why don't these results imply a polynomial-time algorithm for **3**SAT?

3. A disjunctive normal form (DNF) formula is the converse of CNF; i.e., it is an $\vee$ of a number of clauses where each clause is an $\wedge$ of some terms. E.g: $(x \wedge y \wedge z) \vee (z \wedge y \wedge \neg w) \vee (x \wedge \neg z)$. DNF-SAT is the analog problem of (CNF-)SAT: given a DNF formula $f$, determine if there is a satisfying assignment of the corresponding variables that renders the formula true.

   (a) Design and analyze an efficient algorithm for DNF-SAT. *Hint: DNF-SAT can be solved directly, a reduction is not needed.*

   (b) Demonstrate a reduction from 3SAT to DNF-SAT and analyze its runtime. (Hint: use the distributive law.) (Another hint: this reduction will not be efficient, it will not be a polynomial-time reduction).

4. 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*.