CS 170 Homework 12

Due **Friday 11/22/2024, at 10:00 pm (grace period until 11:59pm)**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, explicitly write "none".

2 Upper Bounds on Algorithms for NP Problems

(a) Recall the 3-SAT problem: we have n variables x_i and m clauses, where each clause is the OR of at most three literals (a literal is a variable or its negation). Our goal is to find an assignment of variables that satisfies all the clauses, or report that none exists.

Give a $O(2^n m)$-time algorithm for 3-SAT. Just the algorithm description is needed.

(b) Using the fact that 3-SAT is **NP**-hard, give a $O(2^{n^c})$-time algorithm for any problem in **NP**, where c is a constant (that can depend on the problem). Just the algorithm description and runtime analysis is needed.

*Hint: Since 3-SAT is **NP**-hard, you can use it to solve any problem in **NP**!*

Note: This result is known as $\mathbf{NP} \subseteq \mathbf{EXP}$.

(c) Recall the halting problem from CS70: Given a program (as e.g. a `.py` file), determine if the program runs forever or eventually halts. Also recall that there is no finite-time algorithm for the halting problem. Let us define the input size for the halting problem to be the number of characters used to write the program.

Given an instance of 3-SAT with n variables and m clauses, we can write a size $O(n+m)$ program that halts if the instance is satisfiable and runs forever otherwise. So there is a polynomial-time reduction from 3-SAT to the halting problem.

Based on this reduction and part (b): Is the halting problem **NP**-hard? Is it **NP**-complete? Justify your answer.

3 k-XOR

In the k-XOR problem, we are given n boolean variables $x_1, x_2, \ldots, x_n$, a list of m clauses each of which is the XOR of exactly k distinct variables (that is, the clause is true if and only if an odd number of the k variables in the clause are true), and an integer c. Our goal is to decide if there is some assignment of variables that satisfies at least c clauses.

(a) In the Max-Cut problem, we are given an undirected unweighted graph $G = (V, E)$ and integer α and want to find a cut $S \subseteq V$ such that at least α edges cross this cut (i.e. have exactly one endpoint in S). Give and argue correctness of a reduction from Max-Cut to 2-XOR.

Hint: every clause in 2-XOR is equivalent to an edge in Max-Cut.

(b) Give and argue correctness of a reduction from 3-XOR to 4-XOR.

4 Dominating Set

A dominating set of a graph $G = (V, E)$ is a subset D of V, such that every vertex not in D is a neighbor of at least one vertex in D. Let the Minimum Dominating Set problem be the task of determining whether there is a dominating set of size $\leq k$. Show that the Minimum Dominating Set problem is NP-Complete. You may assume that G is connected.

Hint: Try reducing from Vertex Cover or Set Cover.

5 Approximating Independent Set

In the maximum independent set (MIS) problem, we are given a graph $G = (V, E)$, and our goal is to find the largest set of vertices such that no two vertices in the set are connected by an edge. For this problem, we will assume the degree of all vertices in G is bounded by d (i.e. $\forall v \in V,\ \deg(v) \leq d$).

Consider the following greedy algorithm to approximate the maximum independent set:

```
1: procedure GREEDY-MIS(V, E)
2:     I ← ∅
3:     while G ≠ ∅ do
4:         choose a vertex v in G
5:         I = I ∪ {v}
6:         remove v and all its neighbors from G
7:     return I
```

(a) Provide an example where the greedy approximation does not give the optimal solution.

(b) Provide an approximation ratio for the given greedy algorithm in terms of $|V|$, $|E|$, and/or d. Briefly justify your answer.

