CS 170 Homework 12

Due **4/24/2023, 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, you must explicitly write "none".

2 Max k-XOR

In the Max 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 r. Our goal is to decide if there is some assignment of variables that satisfies at least r clauses.

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

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

3 Dominating Set

A dominating set of a connected graph $G = (V, E)$ is a subset $D \subseteq V$, 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.

(Hint: Try reducing from Vertex Cover or Set Cover.)

4 Multiway Cut

In the multiway cut problem, we are given an undirected graph $G = (V, E)$ with edge weight function w, and k special vertices $s_1, s_2, \ldots, s_k$. Our goal is to find the smallest *multiway cut* of the graph, defined as the set of edges $F \subseteq E$ such that removing F from G disconnects the graph into at least k components, where every s_i is in a different component. When $k = 2$, this is exactly the minimum s-t cut problem, but if $k \geq 3$ the problem becomes NP-hard.

Consider the following algorithm: Let F_i be the set of edges in the minimum cut with s_i on one side and all other special vertices on the other side. Output F, the union of all F_i. Note that F is a multiway cut because removing F_i from G isolates s_i in its own component.

(a) Explain how each F_i can be found in polynomial time. (*Hint: assume max flow take polynomial time*)

(b) Let F^* be the smallest multiway cut. Consider the components formed by removing F^* from G, and let C_i be the set of vertices in the component that contains s_i. Let F_i^* be the set of edges in F^* with exactly one endpoint in C_i. How many different F_i^* does each edge in F^* appear in? Also, which is larger: F_i or F_i^*?

(c) Using your answer to the previous part, show that $|F| \leq 2|F^*|$.

(d) **(Extra Credit)** How could you modify this algorithm to output F such that $|F| \leq (2 - 2/k)\,|F^*|$? *Hint: do you need to include all the F_i's in F? Which one could you potentially leave out?*

5 Polynomial Identity Testing

Suppose we are given a polynomial $p(x_1, \ldots, x_k)$ over the reals such that p is not in any standard form. For example, we might be given the following polynomial:

$$p(x_1, x_2, x_3) = (3x_1 + 2x_2)(2x_3 - 2x_1)(x_1 + x_2 + 3x_3) + x_2$$

We want to test if p is equal to 0, i.e. whether $p(a_1, a_2, \ldots, a_k) = 0$ on all real inputs $a_1, a_2, \ldots, a_k$. This suggests a simple algorithm to check if p equals 0: test p on some $a_1, \ldots, a_k$ and say that p equals 0 if and only if $p(a_1, \ldots, a_k) = 0$. However, this algorithm will fail if $p(a_1, \ldots, a_k) = 0$ and p does not equal 0. We will show throughout this problem that the probability of this failure mode occurring is very low.

To show this, we will prove that for any non-zero degree-d polynomial p and any finite set of reals S, we have the following identity:

$$\Pr_{\{a_i\}_{i=1}^k \overset{\text{i.i.d.}}{\sim} U(S)}[p(a_1, \ldots, a_k) = 0] \leq \frac{d}{|S|} \tag{1}$$

where the notation $\{a_i\}_{i=1}^k \overset{\text{i.i.d.}}{\sim} U(S)$ just means that each a_i is chosen uniformly at random from S (and the a_i's are i.i.d.). Hence, by using a large enough subset S, we are able to correctly determine whether p equals 0 or not with high probability.

Note that the degree of, say, $p(x, y) = x^2y^2 + x^3$ is 4, because the monomial x^2y^2 has degree $4 = 2 + 2$. You may assume here that all basic operations on the reals take constant time.

(a) First, let's see why we might need to use a probabilistic algorithm instead of just using a deterministic one. Give a naive deterministic algorithm to test if a given polynomial p is equal to 0. What is the worst-case runtime of your algorithm?

(b) Show that if p is univariate (i.e. takes in a single variable input) and degree d, then

$$\Pr[p(a_1) = 0] \leq \frac{d}{|S|}$$

(c) Use induction on k to prove the identity in Equation (1) above. (*Hint: Write* $p(x_1, \ldots, x_n) = \sum_{i=0}^{d} p_i(x_1, \ldots, x_{n-1}) \cdot x_n^i$)

(d) Determine a way to use our randomized algorithm to check if two polynomials p and q are identical (i.e., yield the same outputs on all inputs).

6 Coding Question

For this week's homework, you'll implement a couple approximation algorithms for the Traveling Salesman Problem in the python jupyter notebook called `coping_with_tsp.ipynb`. There are two ways you can access the notebook and complete the problems:

1. Click here and navigate to the `HW12` folder if you prefer to complete this question on Berkeley DataHub.

2. Run

   ```
   git clone https://github.com/Berkeley-CS170/cs170-coding-notebooks-sp23
   ```

   in your computer's terminal (and navigate to the `HW12` folder) if you prefer to complete it locally. If you run into any issues with python import or autograder errors, please refer to the local setup instructions here.

Notes:

- *Submission Instructions:* Please download your completed `reductions.ipynb` file and submit it to the gradescope assignment titled "Homework 12 Coding Portion".
- *OH/HWP Instructions:* There will be designated office hours for you to come to ask for conceptual and debugging help. Please visit the coding OH post for more information.
- *Academic Honesty Guideline:* We realize that code for some of the algorithms we ask you to implement may be readily available online, but we strongly encourage you to not directly copy code from these sources. Instead, try to refer to the resources mentioned in the notebook and come up with code yourself. That being said, we **do acknowledge** that there may not be many different ways to code up particular algorithms and that your solution may be similar to other solutions available online.