PSET 4

Due on **Tuesday November 10, 2024 at 11:59 pm ET**

CS 601: Algorithms and Complexity, Fall 2024 (Akash Kumar) Version: **1.0**

Submitted by <**name**>

Grading Policies:

(a) **Total points: 100**

(b) There are **four** problems in this pset (each of which have one or more subparts). Please solve/attempt them all.

(c) You are allowed –indeed you are encouraged– to, collaborate with other students in the class. Please list all the people you collaborated with.

(d) **Quite a few problems should be available online.** You should feel free to consult online resources, but please use those only after you spend some serious time thinking about the problems.

(e) I cannot emphasize this point enough. **With all the freedom given in the points above, it is imperative that you write your solutions yourself.** Please do not copy/paste what your friends wrote or what electronic material you find online. This defeats the purpose of solving a problem set. Indeed, **if you are caught engaging in unseemly practices, severe action will be taken.**

(f) Please write your pseudocode as you see those written in textbooks. Confusing/inscrutable psuedocodes will be penalized with no room for a regrade.

(g) Please write your mathematical claims rigorously and unambiguously. Again, follow the style laid out in textbook or in the class if you are unsure. Ambiguous or unclear claims/proofs will be penalized. Your TAs are not here to debug your proofs/algorithms. If your writeup does not convince them of the correctness, you will rightfully lose points.

1 (20 pts.) Cycle Cover to Bipartite Perfect Matching

In the cycle cover problem, we have a directed graph G , and our goal is to find a set of directed cycles $C_1, C_2, \ldots, C_k$ in G such that every vertex appears in exactly one cycle (a cycle cannot revisit vertices, e.g. $a \to b \to a \to c \to a$ is not a valid cycle, but $a \to b \to c \to a$ is), or declare none exists. In the bipartite perfect matching problem, we have a undirected bipartite graph (a graph where the vertices can be split into L, R , and there are no edges between two vertices in L or two vertices in R), and our goal is to find a set of edges in this graph such that every vertex is adjacent to exactly one edge in the set, or declare none exists. Give a reduction from cycle cover to bipartite perfect matching.

(**Hint:** In a cycle cover, every vertex has one incoming and one outgoing edge.)

Solution:

The key is to produce a graph G' which has a perfect matching *if and only if* G has a cycle cover. How can we do this?

Let us start with the cycle cover instance G. I will now produce a graph $G' = (L, R, E')$ as follows: Add a copy of v to both L and R for every vertex in v in G. For an edge (u, v) in G, we add an edge (u_L, v_R) in the bipartite graph. We claim that G has a cycle cover if and only if G' has a perfect matching.

If G has a cycle cover, then the corresponding edges in the bipartite graph are a bipartite perfect matching: The cycle cover has exactly one edge entering each vertex so each v_R has exactly one edge adjacent to it, and the cycle cover has exactly one edge leaving each vertex, so each v_L has exactly one edge adjacent to it.

On the other hand, if G' has a perfect matching, then G has a cycle cover, which is formed by taking the edges in G corresponding to edges in G': If we have e.g. the edges $(a_L, b_R), (b_L, c_R) \ldots (z_L, a_R)$ in the perfect matching, we include the cycle $a \to b \to c \ldots z \to a$ in G in the cycle cover. Since v_L and v_R are both adjacent to some edge, every vertex will be included in the corresponding cycle cover.

2 (20 PTS.) A TRYST WITH NP COMPLETENESS

Prove that the following problem is NP-hard Input: A directed graph $G = (V, E)$ with each vertex colored blue or red, i.e., $V = V_{blue} \cup V_{red}$.

Goal: Find a Colored cycle which is a directed cycle through all vertices in G that alternates between blue and red vertices.

(**Hint:** You might want to do a reduction from the Hamiltonian Cycle problem.)

Solution:

We reduce the Hamiltonian Cycle problem to this colored cycle problem and this way we will show it is NP-Complete. Given a directed graph $G = (V, E)$, we construct a new graph $G' = (V', E')$ as follows:

For each $v \in V$, create a blue node v_b with an edge to a red node v_r (in G').[1] For each $(u, v) \in E$, add edge (u_r, v_b) to E'. Another way to view this is that for each node $v \in V$, we are redirecting all its incoming nodes to v_r, and all its outgoing nodes originate from v_b (in G').

Now, let us show that if G has a Hamiltonian Cycle then the graph G' obtained above contains a colored cycle which alternates between red and blue vertices and visits all vertices of G'. Consider what happens in G' if G has a Hamiltonian Cycle denoted

$$v_1 \to v_2 \to v_3 \to \ldots v_n \to v_1.$$

In G', you can start at the blue copy of v_1. You take the upward edge to the red copy of v_1. Thankfully enough, the reduction added an edge from the red copy of v_1 to the blue copy of v_2 from where you can visit the red copy of v_2 and so on. This way, you see that G' has an alternately colored Hamiltonian Cycle

$$v_1 \to v_1 \to v_2 \to v_2 \to \ldots v_{n-1} \to v_n \to v_n \to v_1.$$

Now, let us argue the other side. Suppose G' has a Colored Cycle spanning all vertices which alternates. I must show that G has a Hamiltonian Cycle. Consider starting at a blue vertex in G' in this Hamiltonian Cycle. Denote this cycle as

$$v_1 \to v_1 \to v_2 \to v_2 \to \ldots v_{n-1} \to v_n \to v_n \to v_1.$$

By properties of the reduction, the red vertices only provide a way of going from one blue vertex to another. Indeed, if you replace the edges entering and exiting a red vertex with a single edge, you end up with a copy of G! And this graph clearly has a Hamiltonian Cycle.

3 (30 pts.) More NP Completeness, Mission Impossible Style

Consider the following two decision problems:

- $OR(G, k)$: Either G contains no odd length cycle OR G has a vertex cover[2] $S \subseteq V(G)$ of size $|S| \leqslant k$.
- $AND(G, k)$: G contains no odd length cycle AND G has a vertex cover $S \subseteq V(G)$ of size $|S| \leqslant k$.

One of the problems is NP-Complete and the other one is in P. Your mission – you better choose to accept it – is to figure out which of the above problems is NP-Complete. You should prove that your answer is correct by providing a polynomial time algorithm to solve one problem and by proving that the other problem is NP-Complete.[3]

Solution:

Here, $AND(G, k)$ is the problem which is solvable in polynomial time and $OR(G, k)$ is NP-Complete. Let us begin by recalling that a graph G has no odd cycle *if and only if* it is bipartite. To get some intuition about the problem, let us begin by noting that $AND(G, k)$ problem can indeed be solved in polynomial time. Notice that if G contains an odd cycle (that is it is not bipartite), you can straightway conclude that G is a NO instance for the $AND(G, k)$ problem. So, suppose the graph G passes the first clause – that is, G indeed had no odd cycle and so G is bipartite.

From Konig's theorem, it follows that the size of min vertex cover equals the size of the maximum matching in bipartite graphs. And the maximum matching can be found in biparite graphs by setting up an appropriate max flow problem in polynomial time. Now, let us try to show that $OR(G, k)$ is NP-Complete which should intuitively follow as Vertex-Cover(G, k) is NP-Complete. To see this formally, you can perform the standard reduction from 3-Sat to Vertex Cover and notice that it does not produce a bipartite graph.

[2] if you need a reminder about what vertex cover is, consult last homework

[3] I realize that this problem requires knowledge of max-flow which I had not covered. I changed the order in which I presented the topics and therefore this problem will not be graded.

4 (30 pts.) Even More NP-Completeness, One Punch Man Style

So, the following is a predicament faced by people in City Z which is frequented by villainous monsters. The city has started a Hero Association to deal with these troublemakers. There is a set V of n villains and a set H of m heroes. Each hero $h \in H$ is a resource that can be allocated to fight off exactly one villain. Each villain $v \in V$ has a subset $H_v \subseteq H$ of heroes that must be assigned to capture v. The Hero Association needs you to solve the decision problem $ZHERO(n, m, k)$ which asks whether they can allocate the heroes to villains such that at least k villains get captured.

Let us try to get a feel for how this problem behaves through the following subparts.

4.A. (5 pts.) Assume $k = O(1)$ and m and n are large. Help the Hero Association come up with a polynomial time algorithm to solve $ZHERO(n, m, k)$ in this setting.

4.B. (25 pts.) Assume all of n, m, k are all very large. Answer the following questions.

(a) (5 pts.) Show that $ZHERO(n, m, k)$ is in NP.

(b) (15 pts.) Show that $ZHERO(n, m, k)$ is NP-Hard by giving a reduction from the matching problem in 3-uniform hypergraphs. The matching problem in 3-uniform hypergraphs is as follows: You are given a triple of vertex sets A, B, C where $|A| = |B| = |C| = m$. You are told that a subset $E \subseteq A \times B \times C$ forms the set of hyperedges where each hyperedge spans 3 vertices – one each from $A, BandC$. We want to decide whether there exists a subset $M \subseteq E$ with $|M| \geqslant k$ where no two hyperedges in M share any vertex.

(c) (5 pts.) Assume you have an oracle which solves the decision problem $ZHERO(n, m, k)$. Give an algorithm that calls this oracle a polynomial number of times and finds the biggest subset $S \subseteq V$ of villains that can be safely captured.

Solution:

4.A

For each of the $\binom{n}{k}$ subsets of k villains, check whether any hero is required by more than one villain. This can be done simply by going over each of the k villains v, marking the heroes in H_v as needed, and if any hero is marked twice, then this subset fails. Output "yes" if some subset of k villains can be captured, and "no" otherwise.

The time complexity is $\binom{n}{k} \cdot m$ because there are $\binom{n}{k}$ subsets of size k and we pay $O(m)$ time per subset (because all but one hero will be marked only once). For constant k, this indeed runs in polynomial time.

4.B

(a) The idea behind this problem is the following. Given any certificate, which consists of a subset S of k villains, the certifier just checks whether the sets H_u and H_v are disjoint for all $u, v \in S$. Thus, we indeed have a polynomial time procedure which checks whether any given assignment is a *valid or invalid* allocation of heroes to villains.

(b) Let us outline the high level idea first. Each $(a, b, c) \in E$ becomes a villain that requires heroes $H_{(a,b,c)} = \{h_a, h_b, h_c\}$. Thus $n = |E|, H = A \cup B \cup C$, and $m = |A| + |B| + |C|$. We set k to be the same in both problems. The size of the matching is equal to the number of villains that can be captured because both problems model disjointness: if k villains can be captured, a subset M of size k can be found, and vice versa. The reduction takes polynomial time. In more detail, we will now show the following claims.

Claim 1

If the 3-uniform hypergraph contains a matching of size k then in the derived $ZHERO(n, m, k)$ instance is a YES instance.

Proof of Claim 1

In the reduction above, an edge corresponds to a villain and the vertices of the edge are the heroes required to capture this villain. So, if there is a matching M with $|M| = k$ in the 3-uniform hypergraph, we see that for each $e \in M$ it is possible to capture e.

claim 2

If the derived $ZHERO(n, m, k)$ instance is a YES instance then the 3-uniform hypergraph given as input to the reduction must have had a matching of size k.

Proof of Claim 2

Notice that to capture the villain $\langle a, b, c \rangle$ we need heroes h_a, h_b, h_c. It is possible to capture k villains because the derived instance is a YES instance according to the premise. This means it is possible to allocate heroes to villains so that no hero is allocated to more than one villain. Thus, in the 3-uniform hypergraph we started out with, we have a matching of size k as desired.

(c) The algorithm begins by using the decision oracle to figure out what is the biggest number of villains that can be safely captured. This can be done in many ways. One naive way to do this is described below.

- For $j = n \ldots 1$
- Query $ZHERO(n, m, j)$. If it returns NO continue.
- Else, let k be the current value of j. Break.

The algorithm makes repeated calls to the oracle for $ZHERO(n, m, j)$ starting from $j = n$ down to $j = 0$. It picks the first value k in these calls when the decision oracle answers YES.

Now, we know it is possible to capture k villains (and no more). Now we need to figure out which of the k villains can be captured in some optimal allocation scheme σ. Let $V = \{1, 2, \ldots, n\}$ denote the set of all villains. Let us begin by trying to check whether σ allows us to capture the first villain, 1. This is done by calling the decision oracle on the modified input which has the villain set $\{2, 3, \ldots, n\}$ and the hero set $H' = H \setminus H_1$. Finally, it has $k' = k - 1$ villains. We check whether this is feasible by calling $ZHERO(n', |H'|, k')$ (here $n' = n - 1$). This checks whether we could "spare" heroes involved in (hopeful) capture of 1 and still manage to capture $k - 1$ villains. If the oracle answers YES, we know we can safely capture 1 and recurse. If it answers NO, we know optimal solution does not capture 1. We repeatedly use the oracle this way to identify the k villains we can safely capture.