Homework 5

Zicong Mo

February 1, 2018

15. We show that Set-Cover reduces to Nearby Electromagnetic Observation (NEO). Since set-cover is a NP-complete problem, NEO is also a NP-complete problem.

    Suppose we have a set of k subsets, whose collective union forms the universe U. We want to find the minimum number of subsets needed whose union is the U. For each subset S_i, we let F_i be the set of all elements in U not in S_i. We can view U as the set of all frequencies that we want to listen to, i as a particular location, and F_i as the set of all frequencies blocked at that particular location. Therefore we can reduce our set cover problem into finding a set of locations such that all frequencies (the universe) can be observed, given information about which locations have which frequencies blocked. But this is exactly the Nearby Electromagnetic Observation problem, which we can call as a sub-routine to solve the set-cover problem. Therefore NEO is a NP-complete problem.

    For example, suppose we want to find a set cover of $U = \{1, 2, 3, 4, 5\}$ from $S_1 = \{1, 3, 4\}$, $S_2 = \{2, 4\}$, and $S_3 = \{1, 3, 5\}$. This problem is equivalent to finding a sufficient set of locations with:

    $$(F_1, L_1) = (\{2, 5\}, \{\ell_1\})$$
    $$(F_2, L_2) = (\{1, 3, 5\}, \{\ell_2\})$$
    $$(F_3, L_3) = (\{2, 4\}, \{\ell_3\})$$

    Since ℓ_2 has frequencies 2 and 4 unblocked, and ℓ_3 has frequencies 1, 3, and 5 unblocked, $\{\ell_2, \ell_3\}$ forms a sufficient set. These two locations correspond to the sets S_2 and S_3, which form a set cover of U.

18. We show that vertex cover reduces to the decisive subset problem. Suppose we are given a graph $G = (V, E)$. Consider a committee of $|V| = n$ members voting on $|E| = m$ issues. For each edge (u, v) in G, we let committee members u and v be the only members that voted yes on that issue. Every other member abstains from voting on that issue. Thus, we have a total of m issues, each of which has two affirmative votes, and $n - 2$ abstained votes.

    Consider a decisive subset of this panel. Since every issue votes positive, a decisive subset must contain a set of people that vote positive on every issue. However, since the only two people that vote positive on any given issue are members u and v, at least one of these members must be on the decisive subset. Therefore, each issue m has at least one of its members who vote affirmative on the subset. However, because the decisive subset was created such that edges only exist between people that vote affirmative, and we showed that every edge has at least one vertex in it, this is precisely the solution to the vertex cover.

    Therefore Vertex Cover $\leq_p$ Decisive Subset, and the decisive subset problem is NP-complete.

22. If the provided graph $G = (V, E)$ is connected, then we can simply call the algorithm on G to determine if it contains an independent set of size k. We notice that if G is not connected, then it will have an independent set of at least size k if it sizes of the independent sets of the connected components sum to at least k. Therefore, if we can find an algorithm that breaks down G into n connected sub-graphs $G_1, ...G_n$, we can find the largest independent set of G by calling the algorithm on $G_1, ..., G_n$.

    To do this, we can start at a vertex of G, and "follow" each edge it is connected to until there are no more connected edges, and repeat for each of the vertices it is connected to. Once that subsection has no more new vertices, we can pick a new vertex of G that is not in the subsection, and repeat until there are no more vertices. This can be done in linear time, since each vertex v and edge e only has to be checked or traversed once.

    Finally, to find the size of the largest independent set in G_i, we do a binary search from 0 to $|V_i|$, calling the black-box algorithm each time. The binary search component can be done with $\log |V_i|$ calls to the black-box algorithm, which is still polynomial time.

    Since each of these subcomponents runs in either polynomial time or better, this final algorithm to find if there exists an independent set for any arbitrary graph G runs in polynomial time.

```
# Returns True if there exists an IS of at least size k in G, False if
    there does not, "not connected" if G is not connected
def exists_IS(k, G)

# Returns connected components of G
def get_components(G):

# Returns size of largest independent set in G
def get_IS_size(G):

# Returns True/False for arbitrary graph G
def problem_22(k, G):
    b = exists_IS(k, G)
    if b != "not connected":
        return b
    sub_graphs = get_components(G)
    total_size = 0
    for graph in sub_graphs:
        total_size += get_IS_size(graph)
    return total_size >= k
```

31. We show that vertex cover reduces to the undirected feedback set problem. For any undirected graph $G = (V, E)$, construct the graph $G' = (V, E')$ by doubling every edge in E. That is, for any edge $e = (u, v)$ in G, add the edge $e' = (v, u)$ to G'. This operation takes $O(|E|)$. Note that every cycle of length 2 in G' can be described by $[(u, v), (v, u)]$.

    Let F be a feedback set of G'. Consider every cycle of length 2 in G'. By definition, the set $G' - F$ contains no cycles, meaning it cannot contain any cycle of length 2 either. Because every cycle of length 2 can be described by $[(u, v), (v, u)]$, $G' - F$ cannot contain both u and v. But since G' contains both u and v, for any edge in the graph, F must contain at least one of its vertices. Therefore, F is a vertex cover of G'. But since G' and G have the same set of vertices, and G' has strictly more edges than G, F is also a vertex cover of G.

    Since Vertex Cover $\leq_p$ Undirected Feedback Set, Undirected Feedback Set is NP-complete.

36. We show that Hamiltonian path reduces to the Daily Special Scheduling problem. Suppose we have a graph $G = (V, E)$ that we want to find a Hamiltonian path for, if it exists. Let $n = |V|$, and $m = |E|$. We start by checking if G is connected. This check takes $O(n + m)$, using breadth first search. If G is not connected, then there cannot be a Hamiltonian path.

    Otherwise, let $\{R_1, ..., R_n\}$ be a set of n recipes, one for each vertex in G. Let there by a total of m ingredients. If vertices (u, v) are connected, then we say that recipes R_u and R_v share an ingredient. Suppose that each ingredient comes in units of 2 grams, each ingredient lasts 2 days, and the price of each ingredient is \$1 per unit. Finally, let the total budget of the restaurant be $k = \$(2m - n + 1)$. We show that G has a Hamiltonian path if there exists a recipe schedule that keeps the budget under $\$k$.

    We show that every time we jump from one recipe to another, meaning we don't move along connected recipes, the cost of cooking the recipes increases. Therefore, the lowest possible cost is that in which no jumps between recipes are made. Suppose we jump from one recipe to another instead of moving along a connected recipe. Since we don't use the ingredient we bought previously, the ingredient will expire the next day, and we will have to rebuy it eventually. Additionally, we are jumping to a new recipe, we don't have an existing set of relevant ingredients, so we will have to pay $\deg(R)$ dollars instead of the $\deg(R) - 1$ dollars we would have spent if we had an ingredient buffer.

    Finally, we show that the budget requires a Hamiltonian path. We showed previously that if we jump from one recipe to another, then we have to spend more money than if we traversed through recipes that share ingredients. Therefore, it suffices to show that a path through all vertices costs $\$k$ dollars. Suppose we start on a recipe R_i. Since the recipe is connected to $\deg(i)$ other recipes, this recipe requires $\deg(i)$ ingredients. Therefore, we must initially spend $\$\deg(i)$. Suppose that we continue on with a recipe R_j. Since ingredients last two days, we still have the ingredient we bought the previous day, therefore we only have to spend $\$\deg(j) - 1$. If we try to move to a recipe R_k that shares an ingredient with the first recipe R_i, note that we still have to buy the ingredient, since the original purchase of the ingredient has both expired and ran out. Therefore, the price needed to buy any recipe R_r after the initial recipe costs $\$deg(R_r) - 1$. Since we must cook every recipe, the total cost is

$$\deg v_1 + (\deg v_2 - 1) + ... + (\deg v_v - 1) = \sum_{i=1}^{v} \deg(v_i) - (n - 1) = 2m - n + 1.$$

    If there is ever a situation where we have to jump from one recipe to the next, then we have to spend more money than this budget provides us. Therefore, if there exists a recipe schedule that meets the budget, then there exists a Hamiltonian path for that graph.

    Since we have shown that Hamiltonian Path $\leq_p$ Daily Special Scheduling, Daily Special Scheduling is NP-complete.