CS180, Winter 2018
Homework 3
Problems 8.15 8.18 8.22 8.31 8.36

Derek Xu

February 2, 2018

1 Problem 8.15

Proof of NP:
The proof of NP is trivial. Given a certificate which includes or can be transformed into a set of locations that can cover all frequencies, we linear scan through these location, and for each location, linear scan through the set of interference sources, to obtain a set of frequencies that the location set covers. Assuming that there are at most m locations, and b sources, the run time of this is polynomial: O(mb). We can then scan through this set and a set of frequencies, to see if this set encompasses the set of all frequencies, again in polynomial time.

Proof of NP-Complete by reduction:
Claim: Vertex Cover $\leq_p$ Nearby EM Observation
Let us define vertex cover in the following way: Given a graph $G = (V, E)$, let us find a subset $V' \epsilon V$, of size k, such that the set of edges connected to V', $E' \epsilon E$, consists of all edges in E.
Now, assign to each vertex in the graph, a location, $l_j \epsilon L$; and assign to each edge in the graph a frequency, $f_j \epsilon F$. For each frequency (or edge) in the graph, let us define an interference source, i_j s.t. i_j consists of the frequency f_j, and all nodes in the graph which are not connected to the edge representing f_j. Doing this makes it such that in order to satisfy the frequency, f_j, we must pick one of the nodes that is connected to the corresponding edge. Given that we can find a set of locations that can cover all of these frequencies, we can also find a set of vertexes that are connected to all the edges in the graphs. Given that the Nearby EM Observation question returns whether this set is of size k, we see that, in this context, the Nearby EM Observation is equivalent to that of Vertex Cover.
$\therefore$ Vertex Cover $\leq_p$ Nearby EM Observation

Because Vertex Cover is NP-Complete, Nearby EM Observation is NP-Complete.

2 Problem 8.18

Proof of NP:
The proof of NP is trivial. Given a certificate which includes or can be transformed into a subset of committee members whose votes would reflect the votes of the committee, we perform a linear scan through each issue, and, at each issue, a linear scan through the subset, to obtain the voting results of that issue. Assuming that there are at most n members, and t issues, the run time of this is polynomial: O(nt). We can then compare this set of results with the results of the whole committee to check if they are indeed the same, again in polynomial time.

Proof of NP-Complete by reduction:
Claim: Set Cover $\leq_p$ Decisive Subset.
Let us define Set Cover in the following way: Given a universal set $U = \{u_1, u_2, \ldots, u_n\}$, and subsets: $S_1 = \{u_{i1}, \ldots, u_{m1}\}, S_2 = \{u_{i2}, \ldots, u_{m2}\}, \ldots, S_k = \{u_{ik}, \ldots, u_{mk}\}$, find a set of subsets of size j s.t. the elements present in the subsets span all the elements in the universal set.
Now, assign to each element in the universal set an issue, $i_v \epsilon I$, and assign to each subset in the graph a committee member, $c_w \epsilon C$. For each committee member, we will interpret his or her subset of issues as the issues that she or he voted '*yes*'. We will assign every other issue that the committee member voted on as '*abstain*'. Doing this makes it such that if we can find a set of subsets where all of the issues in those subsets span all the issues for the universal set of issues, we obtain an answer to Set Cover. Since all of the issues in the universal set of issues are represented as issues warranting a yes vote by the subsets of committee members, we interpret that the committee as a whole decides that all issues in this set are passed, thus it must be that we need to find a subset of committee members that will also pass all these issues (which is equivalent to set cover.)
$\therefore$ Set Cover $\leq_p$ Decisive Subset.

Because Set Cover is NP-Complete, Decisive Subset is NP-Complete as well.

3 Problem 8.22

To solve this, we first notice 2 cases:
1. The graph is connected.
2. The graph is not connected.

Case 1 is trivial, as A would automatically solve our problem. For Case 2, we must ensure that the original graph is fully connected. This can be easily accomplished by adding a node and connecting it to every other node in the graph. Observe: this added node will be connected to every other node, thus it cannot be part of an independent set as long as $k > 1$. This is trivial because if $k = 1$, only a graph of the null set (no vertices) would make the answer of A on the altered graph (yes) differ from the actual answer (no). Depending on how we choose our graph, we can check for this condition at the beginning of the algorithm. Since this extra node would not matter in our new graph, we simply run A after adding the extra node and the result A returns is the result our algorithm returns. A very short pseudo-code:

Let $G = (V, E)$ be the graph of interest
If A returns "not a connected graph" on input G
    Let node u be a node
    for each node v_i in V in G :
        connect v_i to u with an edge e_i
    add u to the set V in G
    //put special conditions mentioned above here
return A on the input G

4 Problem 8.31

Proof of NP:
The proof of NP is trivial. Given a certificate which includes or can be transformed into the set X of feedback loops in the graph, we may remove X from S, which give us $S-X$. By definition, the $S-X$ graph should have no loops, which in other works makes $S-X$ a tree. We can confirm this by traverse $X-S$ using BFS or DFS. We keep track of all visited nodes in a certain tree, and if we revisit a node, then the certificate is false. Because DFS and BFS algorithms are both polynomial or better, this verifier is polynomial in time.

NOTE:
The key thing to realize here is that the perfect assembly problem finds a set of vertexes that destroys cycles. Thus we need to equate the minimal set of vertexes needed to destroy of cycles to finding the minimal set of vertexes needed to cover all of the edges in the graph. We do this by making the edges themselves cycles.

Proof of NP-Complete by reduction:
Claim: Vertex Cover $\leq_p$ Undirected Feedback Set.
Let us define vertex cover in the following way: Given a graph $G = (V, E)$, let us find a subset $V' \epsilon V$ such that the set of edges connected to V', $E' \epsilon E$, consists of all edges in E.
Now let us alter the graph G by turning each of the edges in G into cycles. We do this by adding a vertex, v_i, for each edge $e_i \epsilon G$. We connect the two vertexes that are connected to by e_i to v_i, effectively creating a 3 vertex cycle for each edge that exists in G. Now, we need to find the minimal set of vertexes that can destroy these cycles. We do this by testing for Undirected Feedback Set and decrementing how many vertexes we want to destroy by 1 each time until we find the minimal amount of vertexes needed to destroy all of the cycles. For sanity, we remove all extra nodes that we added to the graph from this minimal set of vertexes returned by Undirected Feedback. The remaining vertexes are equivalent to finding a minimal set of vertexes needed to be connected to all of the edges in G, i.e. Vertex Cover.
$\therefore$ Vertex Cover $\leq_p$ Undirected Feedback Set.

Because Set Cover is NP-Complete, Undirected Feedback Set is NP-Complete.

5 Problem 8.36

Proof of NP:
The proof of NP is trivial. Given a certificate which includes or can be transformed into k daily specials, costs, and ingredients, we can linear scan through each of the k daily specials and keep track of each ingredient and in which specials and days are used in a graph or array of linked list. We then use the previous data structure to find the optimal cost by maximizing reused ingredients and compare it to the budget, x. This verifier runs in polynomial time.

NOTE:
The key thing to realize here is that the cost of the specials be reduced by reusing ingredients, which allows the budget to be in a tighter bound. To model this phenomenon, we use a graph, where each node is a special dish and each edge represent ingredients that can be reused between different specials.

For more clarity, we express Daily Scheduling in maths:
Let set I represent a set of all ingredients.
Observe ingredient $i_j \epsilon I$:
$s(i_j)$ returns the minimum units of food that can be bought at once.
$c(i_j)$ returns the cost of buying $s(i_j)$
$t(i_j)$ returns the time it takes for i_j to expire.
Let the Set D represent a set of all daily specials.
Let $\varphi(I')$ represent the minimum cost of buying a subset of ingredients I', s.t. we account for the ingredients left over from previous specials.
We map special, $d' \epsilon D$, to a subset of ingredients needed to make it, I'. We characterize this subset by 2 more subsets $I'_1 and I'_2$ where I'_1 as a set of ingredients that need to be bought the day of, and I'_2 as a set of ingredients that can be bought whenever. The question asks to find a set of d' of size k such that the budget is below x.
In other words, the Daily Scheduling Problem finds a set of I' such that $(\sum \varphi(I'_i)|D \rightarrow I'_i) < x$

Proof of NP-Complete by reduction:
Claim: Hamiltonian Path $\leq_p$ Daily Special Scheduling.
Let us define Hamiltonian Path in the following way: Given a graph $G = (V, E)$, we need to find a path (as defined by a sequence of vertexes $P = \{v_1, v_2, \ldots, v_n\}$ where each adjacent vertex is connected by an edge, $e \epsilon E$) that contains all vertexes in V of graph G.
Before reducing the Hamiltonian graph problem, we first define the following parameters: Each ingredient is bought in bundles of 2 units of food. Each ingredient expires in 2 days (1 day after being bought). Each bundle costs 1 unit of money. Following this we define the graph, G, from the Hamiltonian Path in the following way: we let each of the vertexes, $v_i \epsilon V$, be equivalent to specials on the menu, we let the edges, $e_i \epsilon E$, be equivalent to ingredients that can be shared. In other words, we let each special (or vertex) have 2 ingredients (1 unit of food each). If the v_1 is connected to v_2 by 1 edge, then they share 1 ingredient, otherwise no 2 specials require the same ingredients. Notice, this implies the following: if there are k nodes in the graph G, then there are k specials on the menu; if there are k specials and we wish to buy fresh produce for each meal, we will buy $2k$ ingredients; if there are k specials, we can reuse $k - 1$ ingredients; all ingredients can only be reused once due to the expiration date. With these facts we can come to the following conclusion: if our graph can propagate 1 used ingredient through all the specials (i.e. if there is an edge connecting every node in the graph with a path), then there is a way to plan the specials such that our budget is less than: $2k$–$(k - 1) = k + 1$. We can see this effective transforms the Daily Special Scheduling Problem into the Hamiltonian Path Problem.
$\therefore$ Hamiltonian Path $\leq_p$ Daily Special Scheduling.

Because Hamiltonian Path is NP-Complete, Daily Special Scheduling is NP-Complete as well.