Jason Lai
204995126
Dis 1B

CS 180
April 18, 2019

Homework 2

1 Problem 15

Nearby Electromagnetic Observation Problem: Given frequencies, locations, interference sources, and a parameter k, is there a sufficient set of size at most k?

To show that the *Nearby Electromagnetic Observation Problem* is NP-complete, a reduction using *Vertex Cover* is used.

$$Vertex\ Cover \leq_p Nearby\ Electromagnetic\ Observation$$

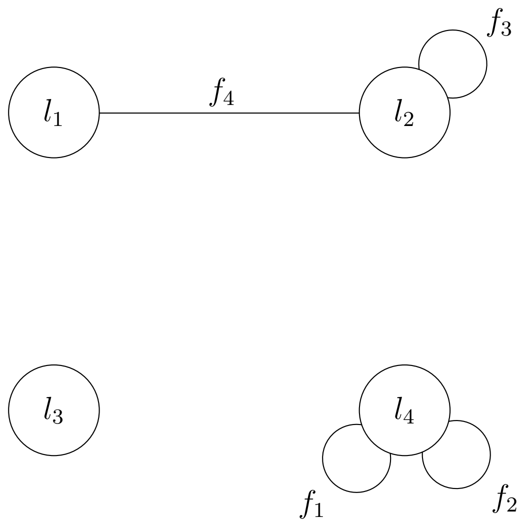
To transform the input of the *Nearby Electromagnetic Observation Problem* into a valid input for *Vertex Cover*, the locations will be used as the nodes for an undirected graph G, and the frequencies will be used as the edges between nodes in G, where the edge means that the frequency that the edge represents is not blocked by the locations represented by the nodes that it connects. To find these "unblocking" locations for certain frequencies, this is just taking the set of locations and subtracting the set of interference source locations for each frequency; if some frequencies do not have "unblocking" locations, then at this time, it can be said that there is no sufficient set of at most size k. Should a frequency only be permitted at one location, then a loop should be drawn at that location node, representing that only that location doesn't block that frequency. Also, frequencies that aren't blocked by more than 2 locations should have edges connecting all of those location nodes. Note that disconnected graphs are entirely possible. This process should take, given there are n frequencies and m locations, approximately $O(n^2)$, since each location node must be drawn and "unblocking" locations for each frequency must be found and their subsequent edges drawn.

Once an undirected graph G is obtained, representing all locations and any unblocked frequencies, *Vertex Cover* can then be called as a subroutine to solve the *Nearby Electromagnetic Observation Problem*, using G and k as input. If *Vertex Cover* returns that there exists a set of nodes of at most size k, which can be used to reach every edge in the graph, and all frequencies are represented by an edge in the graph, then it can be said that there exists a sufficient set of at most size k, thereby satisfying the problem.

Verifying the solution is a matter of simply checking that the certificate: a set of nodes of at most size k, covers the entire set of frequencies (edges) in the graph, which can be done in polynomial time, so the *Nearby Electromagnetic Observation Problem* is in NP. From the aforementioned transformation, it was shown that the NP-complete problem *Vertex Cover* is polynomial time reducible to the *Nearby Electromagnetic Observation Problem.* Therefore, because the the *Nearby Electromagnetic Observation Problem* is in NP and can be reduced to by some other NP-complete problem in polynomial time, then the *Nearby Electromagnetic Observation Problem* is also NP-complete.

Homework 2

Here is an example of a disconnected graph for input to *Vertex Cover* produced by the reduction, using the example outlined by the problem in the book. It can be seen that the sufficient set is made up of locations l_2 and l_4, as these are the nodes which cover every edge (l_1 is not included, as it is more intuitive to use l_2 instead, since it covers 2 edges).



CS 180
April 18, 2019

Jason Lai
204995126
Dis 1B

Homework 2

2 Problem 22

Show how, using calls to A, you could then solve the Independent Set Problem in polynomial time: Given an arbitrary undirected graph G, and a number k, does G contain an independent set of size at least k?

Given an arbitrary undirected graph $G = (V, E)$, a number k, and an algorithm A that runs in polynomial time in the size of G and k, then it is understood that

$$A(G, k) = \begin{cases} \text{"}G\text{ is not connected" if }G\text{ is not connected} \\ \text{"yes" if }G\text{ is connected and has an independent set of size at least }k \\ \text{"no" if }G\text{ is connected and does not have an independent set of size at least }k \end{cases}$$

To solve the *Independent Set Problem* with calls to algorithm A, looking at the returns of A, there is an issue. Assuming there are no constraints on the graph G outlined by the *Independent Set Problem*, then the algorithm A does not account for graphs that are not connected, as the return will only say *G is not connected* if a disconnected graph is passed to it. However, even if a graph is disconnected, an independent set for that graph still exists, so they are still valid in the *Independent Set Problem.* To solve this, an additional algorithm is needed.

The proposed additional algorithm must process a disconnected graph into input that is valid for algorithm A. The initial, naive solution would be to separate a disconnected graph into smaller, connected subgraphs, and use these as input to algorithm A, however, there's a problem with this approach, as it does not account for the additional constraint of the independent set needing to be at least size k for A to return *yes.* So, rather than separating the disconnected graph G into smaller, connected subgraphs, another algorithm could take all of the smaller subgraphs and connect them to make G connected. To do this, an additional node must be added to G; this node must then be connected to all other nodes in G (this ensures that this additional node would not be included in any possible independent sets since it shares an edge with every node). Given there are n nodes in G, this algorithm would run in approximately $O(n)$, since, using an algorithm like breadth first traversal or depth first traversal, every node must be reached, then an edge connecting that node to the newly added node must be added.

Once the additional algorithm has returned, the newly formed, connected graph G can then be passed to algorithm A with the independent set size constraint k. To see if G contains an independent set of size at least k, then the algorithm A can just be called with incrementing $n = 1, 2, ..., k$ as the size constraint until it returns *no.* If $n = k$ is reached without A returning *no*, then there does exist an independent set of at least size k. This would run in approximately $O(n(\textit{complexity of } A))$ which is still polynomial time. However, the total running time of the solution to the *Independent Set Problem* must also include the running time of the additional algorithm used to change a possible disconnected graph G into a connected graph, so the total running time is approximately $O(n(\textit{complexity of } A) + n)$, which is still polynomial time. So the solution satisfies the given problem in polynomial time.

3 Problem 36

Daily Special Scheduling Problem: Given data on ingredients and recipes as above, and a budget x, is there a way to schedule the k daily specials so that the total money spent on ingredients over the course of all k days is at most x? Prove that Daily Special Scheduling is NP-complete.

To show that the *Daily Special Scheduling Problem* is NP-complete, a reduction using *Directed Hamiltonian Path* is used.

$$\textit{Directed Hamiltonian Path} \leq_p \textit{Daily Special Scheduling}$$

To transform the input of the *Daily Special Scheduling Problem* into a valid input for *Directed Hamiltonian Path*, treat the k daily specials as the nodes of a directed graph G and the edges of the G as the ingredients; consequently, each node also represents a day. For each daily special node, each incoming edge on that node is an ingredient that is in the recipe of that node. If an ingredient is only used for one special, then no edge is drawn since that ingredient is a one time purchase and the shelf life of the leftovers is irrelevant, since it won't be used. All possible connections between recipes that share a common ingredient should be included in the directed graph G. For n ingredients and k daily specials, this transformation runs in approximately $O(n^2)$, since each daily special node must be drawn, and the daily specials which include a certain ingredient must be found and the subsequent edges drawn between common recipes.

Once a directed graph G is found, *Directed Hamiltonian Path* can then be called as a subroutine to find a schedule that reaches every daily special node with every ingredient edge of those nodes included. However, a caveat to this algorithm is that multiple Hamiltonian Paths can be found, since, in the aforementioned transformation, it was stated that all nodes that share a certain ingredient are connected, so there are many different possible paths to and from a node for a that ingredient. In order to narrow down the number of Hamiltonian Paths, extra constraints must be defined by the algorithm. These constraints will be as such: each ingredient must be bought in excess of what's needed in a given recipe; for example, if a recipe requires 3 grams of fresh basil, then 5 grams of basil will be purchased. This extra constraint allows for the algorithm to examine the opportunity to save money, since the idea here is that the proposed schedule should fully utilize the shelf life of the leftovers of a certain ingredient that is bought fresh, but without having to compromise on the remainders of other ingredients that have limited shelf lives. Then, this constraint is applied to Hamiltonian Paths by finding the path that utilizes more of the leftovers effectively than the other paths; this path is then used as the final schedule for the restaurant's k daily specials.

Verification of the solution would be to check that the certificate: the given schedule of daily specials, buys and uses enough ingredients to satisfy the recipes of each daily special, without exceeding the budget x. It would be simple to just add up the total cost, in dollars per unit, of all instances where ingredients are bought in the schedule, and to check that the grand total is less than or equal to x; this can be done in polynomial time, so the *Daily Special Scheduling Problem* is in NP. Furthermore, it was shown above that the NP-complete problem *Directed Hamiltonian Path* is polynomial time reducible to the *Daily Special Scheduling Problem*, so, because the *Daily Special Scheduling Problem* is both in NP and reducible to by a different NP-complete problem, the *Daily Special Scheduling Problem* is also NP-complete.