CSE 6140 / CX 4140 Assignment 3
due Oct 16, 2020 at 11:59pm on Canvas

Shasha Liao

1 Dominating set [12 pts]

You're configuring a large network of workstations, which we'll model as an undirected graph G; the nodes of G represent individual workstations and the edges represent direct communication links. The workstations all need access to a common core database, which contains data necessary for basic operating system functions.

You could replicate this database on each workstation; this would make look-ups very fast from any workstation, but you'd have to manage a huge number of copies. Alternately, you could keep a single copy of the database on one workstation and have the remaining workstations issue requests for data over the network G; but this could result in large delays for a workstation that's many hops away from the site of the database.

So you decide to look for the following compromise: You want to maintain a small number of copies, but place them so that any workstation either has a copy of the database or is connected by a direct link to a workstation that has a copy of the database. In graph terminology, such a set of locations is called a *dominating set*.

Thus we phrase the *Dominating Set Problem* as follows. Given the network G, and a number k, is there a way to place k copies of the database at k different nodes so that every node either has a copy of the database or is connected by a direct link to a node that has a copy of the database?

Show that Dominating Set is NP-complete. Follow all steps we have outlined in class for a complete proof. *Hint*: consider the Vertex Cover problem.

Solution:

- Step 1: Show that *Dominating Set Problem* is in NP.
  A potential solution would be $L_k = [v_1, v_2, ..., v_k]$, which is a list of k vertices in the graph G that was placed a copy of the database. To check if L_k is a correct solution, we can loop through all the vertices in the L_k, store their neighbors in a hashset, and then check if the hashset has a length equal to $|V|$, the number of vertices in G. If we use a hashset to store L_k, then the worst runtime for checking a potential solution is $O(k|E|)$, where E is the number of edges in G. Therefore, *Dominating Set Problem* is in NP.

- Step 2: Choose an NP-complete problem X.
  Vertex Cover: Given a graph $G = (V, E)$ and an integer k, dose there exist a subset of vertices $S \subseteq V$ with $|S| \leq k$ such that each edge in E has at least one endpoint in X?
  We know the *Vertex Cover* problem is NP-complete.

- Step 3: Prove that *Vertex Cover* $\leq_p$ *Dominating Set*.
  - Given a *Vertex Cover* instance $G = (V, E)$ and k, we construct a *Dominating Set* instance $G' = (V', E')$ and k' that has a dominating set of size k' iff G has a vertex cover of size k. For each edge $e = (a, b)$ in E, it has at least one endpoint in S. We add a new vertex v_{ab} between vertices a and b and connecting them with edges, i.e., we add two new edges (a, v_{ab}) and (v_{ab}, b). In this way, we constructed our new graph $G' = (V', E')$. Also, note that if G has isolated vertices $I = \{v_i \in V | v_i \text{ is isolated in } G\}$, then these isolated vertices will not be included in a cover set of G since they don't belong to any edge. However, a dominating set in G will have to contain all the isolated vertices since there is no way for them to have a neighbor in the dominating set. Therefore, we need to set $k' = k + |I|$. Obviously, reducing an instance of *Vertex Cover* to an instance of *Dominating Set* only requires a time complexity of $O(|E|)$. Now we are ready to prove that $G = (V, E)$ has a cover set of size k iff $G' = (V', E')$ has a dominating set of size k'.
  - "$\Rightarrow$" Let $X \subseteq V$ be a vertex cover of size k in G. Then $X \cup I$ is a dominating set of size k' in G'. To show this, we prove by contradiction. Since $X \cup I$ has size k' and every vertex in I is for sure in the dominating set $X \cup I$. So the only way for $X \cup I$ not to be a dominating set in G' is that there exists some vertex u in $V' - I$ such that $u \notin X$ and all the neighbors of u are also not in X. The way G' was constructed ensures that u has at least one neighbor $v \in V$. If $u \in V$, then the edge (u, v) is not covered by X, which contradicts with the fact that X is a vertex cover in G. If $u \notin V$, then u was added between two vertices $a, b \in V$, which means u has only two neighbors a and b and neither a nor b are in X. Since there is an edge between a and b in G, we know this edge is not covered by X. Again, we get the contradiction. So $X \cup I$ is a dominating set of size k' in G'.
  - "$\Leftarrow$" Let $X \cup I$ be a dominating set of size k' in G'. If X is not a vertex cover set of G, then there exists an edge $(a, b) \in E$ such that $a \notin X$ and $b \notin X$. Then the vertex v_{ab} added between a and b dose not has a neighbor in $X \cup I$, which contradicts with the fact that $X \cup I$ is a dominating set in G'.

  This completes the proof of *Vertex Cover* $\leq_p$ *Dominating Set*.

2 Frenemies [12 pts]

Assume you are planning a dinner party and going to invite a set of friends. However, among them, there are some pairs of persons who are enemies. You need to create a seating plan and you are wondering if it is possible to arrange this set of n friends of yours around a round table such that none of the two enemies will seat next to each other. Given the set of the n friends and the set of the pairs of enemies, prove that this problem is NP-Complete. Remember to follow the steps from lecture to prove NP-completeness.

You can use the fact that `Hamiltonian Cycle (HC)` is NP-complete.

Solution:

- Step 1: Show that `Frenemies` is in NP.
  A potential solution would be $L = \{a_1, a_2, ..., a_n\}$, which should be a permutation of numbers $1, 2, ..., n$. To check if L is a correct solution, we can loop through L and check if the numbers in L are unique using a hashset and if $(a_i, a_{(i+1)\%n})$ is in the set of the pairs of enemies. This procedure takes $O(n)$ time.

- Step 2: Choose an NP-complete problem: `Hamiltonian Cycle`.
  `Hamiltonian Cycle`: Given an undirected graph $G = (V, E)$, does there exist a simple cycle that contains every node in V?

- Step 3: Prove that `Hamiltonian Cycle` $\leq_p$ `Frenemies`.
  - Given a `Hamiltonian Cycle` instance $G = (V, E)$, we construct a `Frenemies` instance. Suppose there are n vertices in G, we consider n friends in `Frenemies`. For each edge (i, j) missing in G, i.e., $(i, j) \in E_c$, we construct a pair of enemies (i, j) in `Frenemies`. Since there are $\frac{n(n-1)}{2}$ in a complete graph with n vertices, the procedure of constructing all the pairs of enemies takes polynomial time. Now suppose $v_1, v_2, ..., v_n, v_1$ is a Hamiltonian Cycle of G, we can construct a solution to `Frenemies` be arranging the friends in the order of $v_1, v_2, ..., v_n, v_1$ around a table. This takes linear time. Now we are ready to prove that the two problems are equivalent.
  - "$\Rightarrow$" Suppose $v_1, v_2, ..., v_n, v_1$ is a Hamiltonian Cycle of G, then arranging friends in the order of $v_1, v_2, ..., v_n, v_1$ around a table avoids two enemies sitting next to each other. Suppose not, say friends v_i and v_{i+1} are enemies but they sit next to each other. This is a contradiction because there will be no edge between v_i and v_{i+1} in G if friends v_i and v_{i+1} are enemies. Therefore, arranging friends in the order of $v_1, v_2, ..., v_n, v_1$ gives a solution to `Frenemies`.
  - "$\Leftarrow$" Suppose arranging friends in the order of $v_1, v_2, ..., v_n, v_1$ is a solution to `Frenemies`. Since for sure $v_1, v_2, ..., v_n, v_1$ contains all the vertices in G, we just need to prove that it is a cycle in G. If not, there exists an edge $(v_i, v_{i+1}) \notin E$, i.e., $(v_i, v_{i+1}) \in E_c$, which means friends v_i and v_{i+1} are enemies and they are sitting next to each other. This contradicts with the fact that arranging friends in the order of $v_1, v_2, ..., v_n, v_1$ is a solution to `Frenemies`. So $v_1, v_2, ..., v_n, v_1$ is a Hamiltonian cycle in G.

Therefore, we proved that `Frenemies` is NP-complete.

3 Let's go hiking [26 pts]

Alex and Baine go hiking together. They bring a bag of items and want to divide them up. For the following scenarios, decide whether the problem can be solved in polynomial time. If yes, give a polynomial-time algorithm; otherwise prove the problem is NP-complete.

- (8 pts) The bag contains n items of two weights: 1lb and 2lb. Alex and Baine want to divide the items evenly so that they carry the same amount of weight.

Solution: This problem can be solved in constant time. Suppose there are n_1 items of 1lb and n_2 items of 2lb with $n_1 + n_2 = n$. There are three different cases:

- Case 1: n_1 is odd.
  There is no way for Alex and Baine to divide the items evenly since the total number of weights is odd.
- Case 2: n_1 and n_2 are both even.
  Alex and Baine can divide the items evenly simply by each of them taking $n_1/2$ items of 1lb and $n_2/2$ items of 2lb.
- Case 3: n_1 is even but n_2 is odd.
  If $n_1 = 0$, there is no way to divide the items evenly. Otherwise, they can divide the items evenly by treating 2 items of 1lb as 1 item of 2lb and reduce the problem to Case 2.

- (9 pts) The bag contains n items of different weights. Again they want to divide the items evenly.

Solution: Denoting this problem by Y, we can prove that it is NP-complete by showing `Subset Sum` $\leq_p Y$.

- Step 1: Prove that Y is in NP.
  Suppose the bag contains n items of weights $\{w_1, w_2, ..., w_n\}$. A potential solution would be that Alex carries a subset of items with weights $\{w_{k_1}, w_{k_2}, ..., w_{k_p}\}$ where $0 < p < n$, and Baine carries the rest of the items. To check if it is a correct solution, we only need to check if $\sum_{i=1}^{p} w_{k_i} = \frac{\sum_{i=1}^{n} w_i}{2}$, which takes $O(n)$ time.
- Step2: Choose an NP-complete problem: `Subset Sum`.
  `Subset Sum`: given natural numbers $w_1, w_2, ..., w_n$ and a target W, dose there exist a subset of $\{w_1, w_2, ..., w_n\}$ that adds up to exactly W?
- Step 3: Prove that `Subset Sum` $\leq_p Y$.
  - Reduction: given an instance $(w_1, w_2, ..., w_n, W)$ of `Subset Sum`, we can create an instance of Y as this: the bag contains $n+1$ items of weights $\{w_1, w_2, ..., w_n, w_{n+1}\}$ with
  $$w_{n+1} = \sum_{i=1}^{n} w_i - 2W.$$
  We will show that if the `Subset Sum` problem has a solution $\{w_{k_1}, w_{k_2}, ..., w_{k_p}\}$, then we can divide $n+1$ items of weights $\{w_1, w_2, ..., w_n, w_{n+1}\}$ evenly by taking one part as $\{w_{k_1}, w_{k_2}, ..., w_{k_p}, w_{n+1}\}$, and the rest as the other part.
  - "$\Rightarrow$" If $\{w_{k_1}, w_{k_2}, ..., w_{k_p}\}$ is a solution to the `Subset Sum` problem, i.e., $\sum_{i=1}^{p} w_{k_i} = W$. Then we have
  $$\sum_{i=1}^{p} w_{k_i} + w_{n+1} = W + (\sum_{i=1}^{n} w_i - 2W) = \sum_{i=1}^{n} w_i - W = \frac{\sum_{i=1}^{n} w_i + w_{n+1}}{2}.$$
  So $\{w_{k_1}, w_{k_2}, ..., w_{k_p}, w_{n+1}\}$ is a solution to our problem Y.
  - "$\Leftarrow$" If $\{w_{k_1}, w_{k_2}, ..., w_{k_p}, w_{n+1}\}$ is a solution to our problem Y, then
  $$\sum_{i=1}^{p} w_{k_i} + w_{n+1} = \frac{\sum_{i=1}^{n} w_i + w_{n+1}}{2}.$$
  So
  $$\sum_{i=1}^{p} w_{k_i} = \frac{\sum_{i=1}^{n} w_i + w_{n+1}}{2} - w_{n+1} = \frac{\sum_{i=1}^{n} w_i - w_{n+1}}{2} = W.$$
  This proved that $\{w_{k_1}, w_{k_2}, ..., w_{k_p}\}$ is a solution to the `Subset Sum` problem.

Therefore, `Subset Sum` $\leq_p Y$ and Y is NP-complete.

- (9 pts) The bag contains n items of different weights. They want to divide the items such that the weight difference of items they carry is less than 10lbs.

Solution: Denoting this problem by Z, we can prove that it is NP-complete by showing `Subset Sum` $\leq_p Z$.

- Step 1: Prove that Z is in NP.
  Suppose the bag contains n items of weights $\{w_1, w_2, ..., w_n\}$. A potential solution would be that Alex carries a subset of items with weights $\{w_{k_1}, w_{k_2}, ..., w_{k_p}\}$ where $0 < p < n$, and Baine carries the rest of the items. To check if it is a correct solution, we only need to check if $\frac{M-10}{2} < \sum_{i=1}^{p} w_{k_i} < \frac{M+10}{2}$ with $M = \sum_{i=1}^{n} w_i$, which takes $O(n)$ time.

- Step2: Choose an NP-complete problem: problem Y in the previous part.
  Y: The bag contains n items of different weights with each weight being a natural number. Alex and Baine want to divide the items evenly.

- Step 3: Prove that $Y \leq_p Z$.
  * Reduction: given an instance $\{w_1, w_2, ..., w_n\}$ of Y, we can create an instance of Z as this: the bag contains n items of weights $\{w'_1, w'_2, ..., w'_n\}$ with

  $$w'_i = 11 \times w_i.$$

  We will show that if problem Y has a solution $\{w_{k_1}, w_{k_2}, ..., w_{k_p}\}$ iff problem Z has a solution $\{w'_{k_1}, w'_{k_2}, ..., w'_{k_p}\}$.

  * "$\Rightarrow$" If $\{w_{k_1}, w_{k_2}, ..., w_{k_p}\}$ is a solution to problem Y, i.e., $\sum_{i=1}^{p} w_{k_i} = \frac{M}{2}$ with $M = \sum_{i=1}^{n} w_i$. Then we have

  $$\sum_{i=1}^{p} w'_{k_i} = \sum_{i=1}^{p} 11 w_{k_i} = \frac{11M}{2} = \frac{\sum_{i=1}^{n} 11 w_i}{2} = \frac{M'}{2}$$

  with $M' = \sum_{i=1}^{n} w'_i = 11M$. Since $\sum_{i=1}^{p} w'_{k_i} = \frac{M'}{2} \in (\frac{M'-10}{2}, \frac{M'+10}{2})$, we know $\{w'_{k_1}, w'_{k_2}, ..., w'_{k_p}\}$ is a solution to our problem Z.

  * "$\Leftarrow$" If $\{w'_{k_1}, w'_{k_2}, ..., w'_{k_p}\}$ is a solution to our problem Z, then

  $$\sum_{i=1}^{p} w_{k_i} = \frac{1}{11} \sum_{i=1}^{p} w'_{k_i} \in (\frac{M'-10}{22}, \frac{M'+10}{22}) = (\frac{M}{2} - \frac{10}{22}, \frac{M}{2} + \frac{10}{22}).$$

  So

  $$\sum_{i=1}^{p} w_{k_i} = \frac{M}{2}.$$

  This proved that $\{w_{k_1}, w_{k_2}, ..., w_{k_p}\}$ is a solution to problem Y.

Therefore, $Y \leq_p Z$ and Z is NP-complete.

Hint: Recall Subset Sum problem: given a set X of integers and a target number t, find a subset $Y \subset X$ such that the members of Y add up to exactly t.