CS 3510: Design & Analysis of Algorithms 11/4/2024

Problem Set 8: NP-completeness II

Prof. Abrahim Ladha Due: 11/11/2024 11:59pm

- **Please TYPE your solutions using Latex or any other software. Handwritten solutions *won't* be accepted.**

Steps to write a reduction

To show that a problem is NP-complete, you need to show that the problem is both in NP and NP-hard. In a polynomial-time reduction, we have a *Problem B*, and we want to show that it is NP-complete. These are the steps:

- Demonstrate that *problem B* is in the class NP. This is a description of a procedure that verifies a candidate's solution. It needs to address the runtime. You should give a polytime verifier which takes as input a candidate problem, and a witness, and outputs true or false if the witness is a solution.
- Demonstrate that *problem B* is at least as hard as a problem previously proved to be NP-complete. Choose some NP-complete A and prove $A \leq_p B$. You need to show that *problem B* is NP-hard. This is done via polytime reduction from a known NP-complete *problem A* to the unknown textitproblem B ($A \to B$) as follows:
  1. Choose A. You will have many NP-complete problems to pick from later on, and it is best to pick a similar one.
  2. Give your reduction f and argue that it runs in polynomial time.
  3. Prove that $x \in A \implies f(x) \in B$
  4. Prove that $x \notin A \implies f(x) \notin B$
  5. This is sufficient to show $x \in A \iff f(x) \in B$. Since A was NP- complete, and $A \leq_p B$, we can conclude that B is NP-hard. Note: You don't have to provide a formal proof. You can briefly explain in words both implications and why they hold.

The second bullet point above is prove that *problem B* is NP-hard which combined with the first bullet point yields the NP-complete proof.

Problem 1

(25 points)

For an undirected graph $G = (V, E)$, **a strongly independent set** is a set S of vertices such that $S \subseteq V$ and for any two vertices $u, v \in S$ there is no path of length ≤ 2 between u and v. For any path, its length is equal to the number of edges in that path. Consider the following texttttStrongly-Independent-Set problem:
Input: An undirected graph $G = (V, E)$ and an integer $k > 0$.
Output: Whether a strongly independent set of size k in G exists.
Prove that this problem is NP-Complete.

Solution: 1. Show the problem is in NP:

- We can guess a set of k vertices and verify in polynomial time if they form a strongly independent set, meeting the required conditions.

2. Reduction from Vertex-Cover:

- 2. Reduction from Vertex-Cover:In a *Vertex-Cover* problem, given a graph $G = (V, E)$ and integer k, we look for a subset $S \subseteq V$ of k vertices such that every edge has at least one endpoint in S.
- In an *Independent Set* problem, an independent set of size k is a subset with no edges between vertices.

3. Relationship:

- If S is a vertex cover, then $V \setminus S$ is an independent set, as all edges are covered by vertices in S.
- Conversely, if S' is an independent set, $V \setminus S'$ is a vertex cover, since no edges exist within S'.

4. Reduction:

- Use the same graph G and integer k to transform Vertex-Cover into Strongly-Independent-Set, as finding a vertex cover is equivalent to finding an independent set.

Conclusion

- Since Vertex-Cover is NP-complete and the reduction is polynomial, Strongly-Independent-Set is NP-hard.
- Given it's also in NP, we conclude that Strongly-Independent-Set is NP-complete.

Problem 2
(25 points)

Consider the `Busy-Schedule` problem: **Input:** A list of n classes with multiple meeting times (each meeting time represented by an interval $[a, b]$ where $a < b$ and a, b positive integers), a start time s, an end time e, and an integer k.
Output: Whether there exists a k-sized selection S of the n classes such that at any time between s and e at least one class in S is meeting.
Show that the `Busy-Schedule` problem is NP-complete.

Solution:

Show that Busy-Schedule is in NP:

- Given a subset of k classes, we can verify in polynomial time if every time point between s and e is covered by at least one meeting in the subset. This confirms that Busy-Schedule is in NP.

Show that Busy-Schedule is NP-hard:

- We reduce from the Set Cover problem, which is NP-complete:
  - Set Cover: Given a universe U and a collection of subsets $\{S_1, S_2, \ldots, S_m\}$, determine if k subsets can cover all elements in U.
  - Reduction: Map each element in U to a time point in $[s, e]$ and each subset S_i to a "class" with intervals covering the corresponding time points.
  - A solution to Set Cover (covering all elements with k subsets) implies a solution to Busy-Schedule (covering all times with k classes), and vice versa.

Since Busy-Schedule is in NP and is NP-hard via polynomial reduction from Set Cover, it is NP-complete.

Problem 3
(25 points)

Say we are given n ingredients numbered 1 to n. There exists an $n \times n$ matrix M that represents the *deliciousness* between any two ingredients. Between any two ingredients a, b, the *deliciousness* value $M[a, b]$ ranges from 0.0 to 1.0 (0.0 meaning that a and b do not go well with each other and 1.0 meaning that a and b are a perfect match). M is symmetric ($M[a, b] = M[b, a]$) and has zeros along its diagonal ($M[a, a] = 0$). Here is an example matrix with 3 ingredients:

	1	2	3
1	0.0	0.4	0.2
2	0.4	0.0	0.1
3	0.2	0.1	0.0

We want to make a selection of k of these ingredients S such that the sum of the *deliciousness* between all pairs of ingredients in S is maximized. Consider the `Best-Dish` problem: **Input:** The number of ingredients n, the *deliciousness* matrix M, a threshold m, and an integer k
Output: Whether there exists a k-sized subset of the n ingredients S such that the total *deliciousness* of S is $\geq m$.
Show that the `Best-Dish` problem is NP-complete.

Solution:

Best-Dish is in NP:

- Given a subset S of k ingredients, we can calculate the sum of the deliciousness values between each pair of ingredients in S and check if it meets or exceeds m. This verification can be done in polynomial time, confirming that Best-Dish is in NP.

Best-Dish is NP-hard:

- We can reduce from the *Clique* problem, which is NP-complete:
  - Clique: Given a graph G and an integer k, determine if there exists a subset of k vertices that are all mutually connected form a clique.
  - Reduction: Treat each ingredient as a vertex in a graph, and let each pairwise deliciousness value $M[a, b]$ represent an edge weight. For a clique, all pairs in the subset must be fully connected, which corresponds to high deliciousness scores between chosen ingredients.
  - By setting a threshold m appropriately, finding a clique of size k with high edge weights in the Clique problem would correspond to finding a subset S of ingredients in Best-Dish where the sum of deliciousness values is at least m.

Since Best-Dish is in NP and is NP-hard through a reduction from Clique, it is NP-complete.

Problem 4

(25 points)

Show that the following `FILL-BINS` problem is NP-complete.

Input: You are given m bins with weight capacities c_1 to c_m. You are given n items with weights w_1 to w_n.

Output: Whether or not it is possible to put every item in a bin without exceeding the weight capacity of any bin.

Solution:

Fill-Bins is in NP:

- Given an assignment of items to bins, we can verify in polynomial time whether the total weight in each bin does not exceed its capacity. This verification confirms that Fill-Bins is in NP.

Fill-Bins is NP-hard:

- We can reduce from the Partition problem, which is known to be NP-complete:
  - Partition: Given a set of integers, determine if it can be divided into two subsets with equal sums.
  - Reduction: Treat each integer in the Partition problem as an item with weight, and let the two subsets correspond to two bins with equal capacities. By setting the bin capacities to half the total weight, solving the Partition problem would correspond to finding a way to assign items into two bins without exceeding their capacities.

Since Fill-Bins is in NP and is NP-hard via a reduction from Partition, it is NP-complete.

Problem 5

(*ungraded; but strongly recommended*)

In the `Special-Two-Sum` problem, you're given a set of integers S, and you need to determine if it's possible to split S into two non-overlapping sets A and B such that $A \cup B = S$, $A \cap B = \emptyset$ and

$$\sum_{a \in A} a = \sum_{b \in B} b$$

Show that `Special-Two-Sum` is `NP`-complete.