ShanghaiTech University

CS101 Algorithms and Data Structures

Fall 2019

Homework 13

Due date: 23:59, December 26th, 2019

1. Please write your solutions in English.
2. Submit your solutions to gradescope.com.
3. Set your FULL Name to your Chinese name and your STUDENT ID correctly in Account Settings.
4. If you want to submit a handwritten version, scan it clearly. Camscanner is recommended.
5. When submitting, match your solutions to the according problem numbers correctly.
6. No late submission will be accepted.
7. Violations to any of above may result in zero score.
8. In this homework, all the proofs need three steps. The demand is on the next page. If you do not answer in a standard format, you will not get any point.

Demand of the NP-complete Proof

When proving problem A is NP-complete, please clearly divide your answer into three steps:

1. Prove that problem A is in NP.
2. Choose an NP-complete problem B and for any B instance, construct an instance of problem A.
3. Prove that the yes/no answers to the two instances are the same.

0. Proof Example

Suppose you are going to schedule courses for the SIST and try to make the number of conflicts on more than K. You are given 3 sets of inputs: $C = \{\cdots\}, S = \{\cdots\}, R = \{\{\cdots\}, \{\cdots\}, \cdots\}$. C is the set of distinct courses. S is the set of available time slots for all the courses. R is the set of requests from students, consisting of a number of subsets, each of which species the course a student wants to take. A conflict occurs when two courses are scheduled at the same slot even though a student requests both of them. Prove this schedule problem is NP-complete.

1. Firstly, for any given schedule as a certificate, we can traverse every student's requests and check whether the courses in his/her requests conflicts and count the number of conflicts, and at last check if the total number is fewer than K, which can be done in polynomial time. Thus the given problem is in NP.
2. We choose 3-coloring problem which is a NP-complete problem. For any instance of 3-coloring problem with graph G, we can construct an instance of the given problem: let every node v becomes a course, thus construct C; let every edge (u, v) becomes a student whose requests is $\{u, v\}$, thus construct R; let each color we use becomes a slot, thus construct S; at last let K equals to 0.
3. We now prove G is a yes-instance of 3-coloring problem if and only if (C, S, R, K) is a yes-instance of the given problem:
   - "$\Rightarrow$": if G is a yes-instance of 3-coloring problem, then schedule the courses according to their color. Since for each edge (u, v), u and v will be painted with different color, then for each student, his/her requests will not be scheduled to the same slot, which means the given problem is also a yes-instance.
   - "$\Leftarrow$": if (C, S, R, K) is a yes-instance of the given problem, then painting the nodes in G according to their slots. Since $K = 0$, then for every student, there is no conflict between their requests, which suggests that for every edge (u, v), u and v will not be painted with the same color. It is also a yes-instance of 3-coloring problem.

Therefore, the given problem is NP-complete.

1. (2*1') True or False

(a) Suppose a NP problem is proved to be solved by an algorithm in polynomial time, then it indicates that NP=P.

(a)
F

(b) For any NP problem, the NPC problem can be reduced to that problem since it is the definition of NPC.

(b)
T

2. (2*4') NP

Show the following problems are in NP.

(a) Given a graph with n nodes and a number k, are there k nodes that form a clique?(vertices in a clique are all connected to each other)
Part(A): Construct the verifier(1')

Denote the given graph as G. Given a subgraph G_1 that is a clique of size m, can we find a vertex belonging to the G while not belonging to G_1 that is connected to all the vertices of G_1?

Part(B): If the instance x has a solution, show that your verifier works(1')

1. Initialize the number of nodes in a clique with 2.
2. For a subgraph G_1 with two connected vertices v1 and v2, if we can find a vertex connected to both v1 and v2, add the vertex and the according edges to the subgraph, and m $\leftarrow m+1$.
3. If we can find a vertex connected to all the three nodes in the subgraph, add the vertex and the according edges to the subgraph, and m $\leftarrow m+1$.
4. Apply the above method recursively.
5. If at some time m has equaled to k, then the answer is yes (there are k nodes that form a clique).

Part(C): If the instance x has no solution, show that your verifier works(1')

1. Initialize the number of nodes in a clique with 2.
2. For a subgraph G_1 with two connected vertices v1 and v2, if we can find a vertex connected to both v1 and v2, add the vertex and the according edges to the subgraph, and m $\leftarrow m+1$.
3. If we can find a vertex connected to all the three nodes in the subgraph, add the vertex and the according edges to the subgraph, and m $\leftarrow m+1$.
4. Apply the above method recursively.

5. If at some time we cannot find a vertex connected to all the vertices of the subgraph, and k is smaller than k, remove the previous added node and change another that meets the verifier if we can find one and apply the method above. If we cannot find one, remove the previous node and change another node that meets the verifier. Apply the method above recursively. If at last we have removed all the nodes, then the answer is no (we cannot find a clique withh size k).

Part(D): Show that verifier works in polytime(1')

To find a clique with size k we iterate the same method $O(k)$ times. To determine a proper node added to the subgraph costs $O(n)$ time. Hence, it takes $O(n^k)$ times in the worst case.

(b) Given a set of n cities, and distances between each pair of cities, is there a path vist each city exactly once, and has distance at most D, for a given D?
Part(A): Construct the verifier(1')

The n cities can be seen as n vertices and they form a complete weighted graph. For a given threadlike subgraph(with total weight w), can we add one end of a minimum edge(with weight w1) that we can find to one end of the subgraph and w+w1 $<= D$.

Part(B): If the instance x has a solution, show that your verifier works(1')

(a) Find the minimum edge in the graph and pick one of its end nodes.
(b) If we can find another edge conected to the picked node and w+w1 $<= D$, add the node and the edge to the subgraph.
(c) Apply the above method recursively and at last if all the nodes are added to the subgraph and the total weight is no larger than D, the instance has a solution.

Part(C): If the instance x has no solution, show that your verifier works(1')

(a) Find the minimum edge in the graph and pick one of its end nodes.
(b) If we can find another edge conected to the picked node and w+w1 $<= D$, add the node and the edge to the subgraph. w is updated to w+w1
(c) Apply the above method recursively, at last if we cannot find an edge to ensure that w+wn is no larger than D while not all the nodes are included in the subgraph, the instance has no solution.

Part(D): Show that verifier works in polytime(1')
Determining a proper node and the corresponding edge added to the subgraph costs $O(n)$ time. One time of verifier costs $O(1)$. Hence, the verifier works in $O(n)$ time.

3. (★ 10') Knapsack Problem

Consider the Knapsack problem. We have n items, each with weight a_j and value $c_j (j = 1, ..., n)$. All a_j and c_j are positive integers. The question is to find a subset of the items with total weight at most b such that the corresponding profit is at least k (b and k are also integers). Show that Knapsack is NP-complete by a reduction from Subset Sum. (Subset Sum Problem: Given n natural numbers $w_1, \cdots, w_n$ and an integer W, is there a subset that adds up to exactly W?)

1. Firstly, for any given subset as a certificate, we can traverse the subset to check whether the total weight of the items is no larger than b and the total profit is no less than k. Traversing costs $O(n)$, the total time of the verifier is $O(n)$, which is a polynomial time.
2. We choose Subset Sum Problem which is a NP-complete problem. For an instance of Subset Sum Problem with graph G, we can construct an instance of the given problem: let every node v becomes an item with value, thus construct c_j; let every edge (u,v) becomes the weight, thus construct a_j; let b and k equals to 0.
3. We now prove G is a yes instance of Subset Sum Problem if and only if (c_j, a_j) is a yes instance of the given problem:
   - "⇒": if G is a yes-instance of Subset Sum Problem, then the sum of a_j is equal to b, the sum of c_j is equal to k, which means the given problem is also a yes-instance.
   - "⇐": if (c_j, a_j) is a yes-instance of the given problem, then the sum of a_j is equal to b, the sum of c_j is equal to k. It is also a yes-instance of Subset Sum Problem.

   Therefore, the given problem is NP-complete.

4. (★★ 10') Zero-Weight-Cycle Problem

You are given a directed graph G = (V,E) with weights w_e on its edges $e \in E$. The weights can be negative or positive. The Zero-Weight-Cycle Problem is to decide if there is a simple cycle in G so that the sum of the edge weights on this cycle is exactly 0. Prove that this problem is NP-complete by a reduction from the Directed-Hamiltonian-Cycle problem.

1. Firstly, for any given sequence of node which forms a cycle as a certificate, we can traverse the cycle to check whether the total weight of the edges is equal to 0. Traversing costs $O(n)$, the total time of the verifier is $O(n)$, which is a polynomial time.
2. We choose Directed-Hamiltonian-Cycle problem, which is a NP-complete problem.
3. A Zero-Weight-Cycle graph is a simple circle, so it is a Hamiltonian Cycle.
    - "$\Rightarrow$": if G is a yes-instance of Zero-Weight-Cycle Problem, then it is also a Hamiltonian Cycle since it is a simple graph.
    - "$\Leftarrow$": If it is a Hamiltonian Cycle $c_1, ...c_{n-1}, c_1$, let the weight of edge (c_1, c_2) be 1-n and other edges with weight 1, then it is a Zero-Weight graph.

   Therefore, the given problem is NP-complete.

5. (★★★ 10') Subgraph Isomorphism

Two graphs $G = (V, E)$ and $G' = (V', E')$ are said to be isomorphic if there is a one-to-one mapping $f : V \to V'$ such that $(v, w) \in E$ if and only if $(f(v), f(w)) \in E'$. Also, we say that G' is a subgraph of G if $V' \subseteq V$,and $E' = \{(u, v) \in E | u, v \in V'\}$. Given two graphs G and G',show that the problem of determining whether G' is isomorphic to a subgraph of G is NP-complete. (K-clique Problem: Given a graph with n nodes, whether there exist k nodes that are all connected to each other?)

1. Firstly, for any given graph G' and a sub graph G" of G as a certificate, we can traverse G' to check whether it is isomorphic to G". The total time of the verifier is $O(|V|^2 * |E|^2)$, which is a polynomial time.
2. We choose K-clique Problem, which is a NP-complete problem.
3.
   - "$\Rightarrow$": if S is a yes-instance of K-clique Problem, then all the isomorphism of the subgraph of S's k-clique is also isomorphic to the subgraph of S.
   - "$\Leftarrow$": If it is an instance of Subgraph Isomorphism, then there exists a solution 1 of G' to k-clique problem

   Therefore, the given problem is NP-complete.