CSCE-629 Analysis of Algorithms

Fall 2019

Instructor: Dr. Jianer Chen
Office: HRBB 338C
Phone: 845-4259
Email: `chen@cse.tamu.edu`
Office Hours: T,Th 10:50 am–12:30 pm

Teaching Assistant: Qin Huang
Office: HRBB 309D
Phone: (979) 402-6216
Email: `huangqin@email.tamu.edu`
Office Hours: MWF 3:30 pm–4:30 pm

Assignment # 6
(Prepared with the TA Qin Huang)

1. A vertex cover in an undirected graph G is a set C of vertices in G such that every edge in G has at least one end in C. Consider the following two versions of the Vertex-Cover problem:

> VC-D: Given a graph G and an integer k, decide whether G contains a vertex cover of at most k vertices.
>
> VC-O: Given a graph G, construct a minimum vertex cover for G

Prove: VC-D is solvable in polynomial time if and only if VC-O is solvable in polynomial time.

Solutions. Suppose there is a polynomial time algorithm $\mathcal{A}$ for VC-O problem. We can construct a polynomial time algorithm for VC-D as follows: given an instance (G, k) of the VC-D problem, run $\mathcal{A}$ on G to obtain an optimal cover $\mathcal{C}$, if $|\mathcal{C}| \leq k$, then (G, k) is a yes-instance; otherwise, it's a no-instance.

Conversely, assume there is a polynomial time algorithm $\mathcal{B}$ for the VC-D problem. We can construct a polynomial time algorithm for VC-O is as follows:

Algorithm 1 Pseudocode for VC-O in Problem 1

```
for i = 0 to n do
    if B(G, i) returns yes then
        break;
let C = ∅;
while i > 0 do
    for each vertex v ∈ G do
        if B(G − v, i − 1) return yes then
            add v into C;
            let G = G − v; i − −;
            break;
```

Obviously, if the algorithm $\mathcal{B}$ runs in polynomial time, then Algorithm 1 is also of polynomial time and correctly constructs a minimum vertex cover $\mathcal{C}$. $\square$

2. Prove that the VC-D problem given in Question 1 is in NP.

Solutions. To show that VC-D is in NP, for a given instance $(G = (V, E), k)$, the certificate we choose is the vertex cover $V' \subseteq V$ itself. The verification algorithm affirms that $|V'| \le k$, and then it checks, for each edge $[u, v] \in E$, that $u \in V'$ or $v \in V'$. We can easily verify the certificate in polynomial time. Hence, VC-D is in NP. $\square$

3. Using the fact that the independent set problem is NP-complete, prove that the following problem is NP-complete:

> Clique: Given a graph G and an integer k, is there a set C of k vertices in G such that for every pair v and w of vertices in C, v and w are adjacent in G?

Solutions. We first show that Clique is in NP. Suppose we are given a graph $G = (V, E)$ and an integer k. The certificate is the clique $V' \subseteq V$ itself. The verification algorithm affirms that $|V'| = k$, and then it checks, for each pair $u, v \in V'$, that $[u, v] \in E$. We can verify the certificate in polynomial time.

Now, we show that Independent Set problem $\le_p$ Clique. This reduction relies on the notion of the "complement" of a graph. Given an undirected graph $G = (V, E)$, we define the complement of G as $\overline{G} = (V, \overline{E})$, where $\overline{E} = \{[u, v] : u, v \in V, u \neq v, \text{ and } [u, v] \notin E\}$.

The reduction algorithm takes as input an instance $\langle G, k\rangle$ of the Independent Set problem. It computes the complement $\overline{G}$, which we can easily do in polynomial time. The output of the reduction algorithm is the instance $\langle \overline{G}, k\rangle$.

Obviously, G has an independent set of size k if and only if $\overline{G}$ has a clique of size k. Therefore, Clique is NP-complete. $\square$

4. Prove: if the problem VC-O is solvable in polynomial time then P = NP. *Hint:* you may use the result in Question 1.

Solutions. If VC-O is solvable in polynomial time, then by Question 1, the decision problem VC-D is also solvable in polynomial time. Thus, there is an algorithm A_1 that solves the VC-D problem in time $O(n^{c_1})$, where c_1 is a constant. Now let Q be any problem in NP. Since the problem VC-D is NP-complete, we have $Q \le_m^p$ VC-D. That is, there is an algorithm A_2 that on an instance x of Q produces an instance y of VC-D in time $O(|x|^{c_2})$, where c_2 is a constant, such that x is a yes-instance for Q if and only if y is a yes-instance for VC-D.

Now consider the following algorithm A_3 for the problem Q in NP: on an instance x of Q, first apply the algorithm A_2 to produce an instance y for VC-D, then apply the algorithm A_1 for VC-D on the instance y to decide if y is a yes-instance of VC-D, which will directly give a decision on the instance x of Q. Note that the algorithm A_2 on x runs in time $O(|x|^{c_2})$ while the algorithm A_1 on y runs in time $O(|y|^{c_1}) = O(|x|^{c_1 c_2})$ (note that the length $|y|$ of y cannot be larger than $O(|x|^{c_1})$ because y was produced by the algorithm A_2 that runs in time $O(|x|^{c_1})$). Therefore, the algorithm A_3 solves the problem Q in time $O(|x|^{c_1} + |x|^{c_1 c_2})$, which is a polynomial of $|x|$. Thus, the problem Q is solvable in polynomial time, thus, is in P. Since Q is an arbitrary problem in NP, this shows that P = NP. $\square$