CSCE 411-502 Design and Analysis of Algorithms

Spring 2025

Instructor: Dr. Jianer Chen
Office: PETR 428
Phone: (979) 845-4259
Email: chen@cse.tamu.edu
Office Hours: MW 1:30 pm–3:00 pm

Senior Grader: William Kang
Phone: 979) 575-9987
Email: rkdvlfah1018@tamu.edu
Questions: via phone and email and by appointments

Assignment #7 Solutions

1. Let Q_1, Q_2, and Q_3 be decision problems. Prove: if $Q_1 \leq_m^p Q_2$ and $Q_2 \leq_m^p Q_3$, then $Q_1 \leq_m^p Q_3$. You should explain why your reduction from Q_1 to Q_3 runs in time polynomial of the length of the instances of Q_1.

Proof. Since $Q_1 \leq_m^p Q_2$ and $Q_2 \leq_m^p Q_3$, by definition, there exist polynomial-time algorithms A_1 and A_2 such that (1) for each x_1, x_1 is a yes-instance of Q_1 if and only if $A_1(x_1)$ is a yes-instance of Q_2; and (2) for each x_2, x_2 is a yes-instance of Q_2 if and only if $A_2(x_2)$ is a yes-instance of Q_3. Let the running time of algorithm A_1 on input x_1 be bounded by $p_1(|x_1|)$, where $p_1(|x_1|)$ is a polynomial of $|x_1|$, and let the running time of algorithm A_2 on input x_2 be bounded by $p_2(|x_2|)$, where $p_2(|x_2|)$ is a polynomial of $|x_2|$.

Now consider the following algorithm A_0: on input x_1, A_0 first calls the algorithm A_1 on x_1 to produce $x_2 = A_1(x_1)$, then calls the algorithm A_2 on x_2 to produce $x_3 = A_2(x_2)$ and makes x_3 as its output. Therefore, we have $A_0(x_1) = A_2(A_1(x_1))$. By the properties we know for the algorithms A_1 and A_2, we have

$$\begin{gathered} x_1 \text{ is a yes-instance of } Q_1 \text{ if and only if} \\ x_2 = A_1(x_1) \text{ is a yes-instance of } Q_2, \text{ which holds true if and only if} \\ x_3 = A_2(x_2) = A_2(A_1(x_1)) = A_0(x_1) \text{ is a yes-instance of } Q_3. \end{gathered}$$

Therefore, x_1 is a yes-instance of Q_1 if and only if $A_0(x_1)$ is a yes-instance of Q_3.

Now we consider the time complexity of the algorithm A_0. On input x_1, the algorithm A_1 runs in time $\leq p_1(|x_1|)$ and produces $x_2 = A(x_1)$. Since each step of the algorithm A_1 can print at most one character to its output, we have $|x_2| \leq p_1(|x_1|)$. On input x_2, the algorithm A_2 runs in time $\leq p_2(|x_2|)$. Therefore, the running time of the algorithm A_0 on input x_1 is bounded by

$$t(|x_1|) = p_1(|x_1|) + p_2(|x_2|) \leq p_1(|x_1|) + p_2(p_1(|x_1|)),$$

where $p_1(|x_1|) + p_2(p_1(|x_1|))$ is a polynomial of $|x_1|$. Thus, the algorithm A_0 is a polynomial-time algorithm, so it is a polynomial-time reduction from the problem Q_1 to the problem Q_3. This completes the proof that $Q_1 \leq_m^p Q_3$.

2. Prove: if an $\mathcal{NP}$-hard prolem is solvable in polynomial time, then $\mathcal{P} = \mathcal{NP}$.

Proof. Suppose that Q_2 is an $\mathcal{NP}$-hard problem that can be solved in polynomial time. Thus, there is an algorithm A_2 that runs in time $p_2(n)$ and solves the problem Q_2, where $p_2(n)$ is a polynomial of n. We prove that under this assumption, $\mathcal{P} = \mathcal{NP}$. Since we already know that $\mathcal{P} \subseteq \mathcal{NP}$, it suffices to prove that $\mathcal{NP} \subseteq \mathcal{P}$, i.e., to prove that every problem in $\mathcal{NP}$ is also in $\mathcal{P}$.

Let Q_1 be any problem in $\mathcal{NP}$. Since Q_2 is $\mathcal{NP}$-hard, we have $Q_1 \leq_m^p Q_2$, i.e., there is a polynomial-time algorithm A_1 that on input x_1 produces an output $x_2 = A_1(x_1)$ such that x_1 is a yes-instance of Q_1 if and only if x_2 is a yes-instance of Q_2. Let the running time of algorithm A_1 on input x_1 be bounded by $p_1(|x_1|)$, where $p_1(|x_1|)$ is a polynomial of $|x_1|$. As we did in Question 1, since each step of the algorithm A_1 can print at most one character to its output, we have $|x_2| = |A_1(x_1)| \leq p_1(|x_1|)$.

Now consider the following algorithm A_0: on input x_1, A_0 first calls the algorithm A_1 on x_1 to produce $x_2 = A_1(x_1)$ in time $p_1(|x_1|)$, then calls the algorithm A_2 on x_2 to determine, in time $p_2(|x_2|)$, if x_2 is a yes-instance of Q_2. The algorithm A_0 returns "yes" if and only if the algorithm A_2 on x_2 returns "yes."

Since A_1 is a polynomial-time reduction from Q_1 to Q_2, x_1 is a yes-instance of Q_1 if and only if $x_2 = A_1(x_1)$ is a yes-instance of Q_2. Therefore, the algorithm A_0 given above correctly determines if the given input x_1 is a yes-instance of Q_1, thus solves the problem Q_1. The running time of the algorithm A_0 is bounded by

$$t(|x_1|) = p_1(|x_1|) + p_2(|x_2|) \leq p_1(|x_1|) + p_2(p_1(|x_1|)),$$

which is a polynomial of the length $|x_1|$ of the input x_1 to the algorithm A_0. As a result, the algorithm A_0 is a polynomial-time algorithm that solves the problem Q_1, i.e., the problem Q_1 is in $\mathcal{P}$.

Since Q_1 is an arbitrary problem in $\mathcal{NP}$, the above proof shows that every problem in $\mathcal{NP}$ is also in $\mathcal{P}$, i.e., $\mathcal{NP} \subseteq \mathcal{P}$, which gives $\mathcal{NP} = \mathcal{P}$.

3. Using the fact that the INDEPENDENT SET problem is $\mathcal{NP}$-complete, prove that the following problem is $\mathcal{NP}$-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?

Proof. To prove that the problem CLIQUE is $\mathcal{NP}$-complete, we need to prove:

(1) CLIQUE is in $\mathcal{NP}$, and (2) CLIQUE is $\mathcal{NP}$-hard.

We first prove that CLIQUE is in $\mathcal{NP}$. To prove this, we need to give an algorithm V_0 such that on input (x, y):

(1) if x is a yes-instance of CLIQUE, then for some y_0, $V_0(x, y_0)$ returns "yes,"
(2) if x is a no-instance of CLIQUE, then for all y, $V_0(x, y)$ returns "no," and
(3) $V_0(x, y)$ runs in time polynomial in $|x|$.

Note that an instance of CLIQUE is of the form (G, k), where G is a graph and k is an integer. Consider the following algorithm:

```
V0(x, y)
1. if (x is not of the form (G, k)) return ('no');
\\ now x is of the form x = (G, k), where G is a graph and k is an integer
2. if (y is not a set of k distinct vertices in G) return ('no');
\\ now y is a set of k distinct vertices in the graph G
3. if (any two vertices in y are not adjacent in G) return ('no');
4. return ('yes').
```

We show that the algorithm $V_0(x, y)$ satisfies the conditions (1)-(3) above.

(1) If x is a yes-instance of CLIQUE, then $x = (G, k)$, where G is a graph in which there is a set C of k vertices such for any two vertices v and w in C, v and w are adjacent in G. In this case, simply let y_0 be the set C. Then on the input (x, y_0), where $x = (G, k)$ and $y_0 = C$, the algorithm V_0 can pass through the tests in steps 1-3, and returns "yes" at step 4.

(2) if x is a no-instance of CLIQUE, then for any y, either x is not in the valid format (G, k) thus the algorithm $V_0(x, y)$ returns "no" at step 1, or y is not a set of k vertices in the graph G thus the algorithm $V_0(x, y)$ returns "no" at step 2. If the input (x, y) passes through the tests in steps 1-2, then $x = (G, k)$ and y is a set of k vertices in G. However, since $x = (G, k)$ is a no-instance of CLIQUE, there is no set of k vertices in which all vertices are adjacent. In particular, there must be a pair v and w of vertices in the set y such that v and w are not adjacent. Thus, the algorithm on (x, y) will returns "no" at step 3. This shows that in this case, the algorithm on (x, y) can never reach step 4, thus it always returns "no.".

(3) Assume that graphs are given in their adjacency matrices. Thus, step 1 of the algorithm runs in time $O(|x|)$ to check if x is of the format (G, k). Step 2 simply checks if y is a set of k distinct integers in $\{1, 2, ..., n\}$, where the number n of vertices in G can be found when we examine the adjacency matrix of the graph G. Finally, with the adjacency matrix of G, step 3 takes time $O(k^2) = O(n^2)$. In conclusion, the algorithm V_0 runs in time polynomial of $|x|$ on input (x, y).

This verifies that the algorithm V_0 satisfies all the conditions (1)-(3) for the problem CLIQUE. By the definition, this proves that the problem CLIQUE is in the class $\mathcal{NP}$.

Now we show that the problem CLIQUE is $\mathcal{NP}$-hard. For this, we pick the known $\mathcal{NP}$-hard problem INDEPENDENT SET (IS) and show IS $\leq_m^p$ CLIQUE. An instance of the problem IS takes the form (G, k) and asks whether there is a set I of k vertices in which no two are adjacent, while an instance of the problem CLIQUE takes the form (G', k') and asks whether there is a set C of k' vertices in which all vertices are adjacent in G'. It is quite to see that the two problems are somehow "complement" each other.

Let $G = (V, E)$ be any graph. We define the *complement graph* G_c of G as the graph $G_c = (V, E')$ that has the same set V of vertices. On the other hand, for any two vertices v and w in V, v and w are adjacent in G_c *if and only if* v and w are not adjacent in G.

The polynomial-time reduction from IS to CLIQUE is as follows: on an instance (G, k) of IS, we construct the complement graph G_c of G, then output (G_c, k) as an instance of CLIQUE. This algorithm obviously runs in polynomial time.

We prove that (G, k) is a yes-instance of IS if and only if (G_c, k) is a yes-instance of CLIQUE.

If (G, k) is a yes-instance of IS, then there is a set I of k vertices in G in which no two vertices are adjacent. By the structure of the complement graph G_c, for any two vertices v and w in this same set I, the vertices v and w are adjacent in the graph G_c. Thus, the set I makes a clique of k vertices in G_c and (G_c, k) is a yes-instance of CLIQUE.

For the other direction, suppose that (G_c, k) is a yes-instance of CLIQUE, then there is a set C of k vertices in G_c such that every two vertices in C are adjacent in G_c. Again by the structure of the complement graph G_c, it implies that for the same set C of vertices in G, no two vertices in C are adjacent in G. Thus, the set C makes an independent set of k vertices in G and (G, k) is a yes-instance of IS.

This completes the proof that IS $\leq_m^p$ CLIQUE. Since IS is $\mathcal{NP}$-hard, this reduction shows that CLIQUE is $\mathcal{NP}$-hard.

Summarizing the above discussion, we conclude that the problem CLIQUE is $\mathcal{NP}$-complete.