CSCE-629 Analysis of Algorithms

Spring 2019

Instructor: Dr. Jianer Chen
Office: HRBB 315C
Phone: 845-4259
Email: chen@cse.tamu.edu
Office Hours: T,Th 11:00 am–12:30 pm

Teaching Assistant: Qin Huang
Office: HRBB 315A
Phone: 402-6216
Email: huangqin@email.tamu.edu
Office Hours: MWF 3:00 pm–4:00 pm

Assignment # 6
(Due April 30, 2019)

(The homework is due April 30, at 5:00pm. Please submit it to HRBB 315C)

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.

2. Prove that the VC-D problem given in Question 1 is in $\mathcal{NP}$.

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?

Definitions.

1. $\mathcal{P}$ is the collection of all (decision) problems that can be solved in polynomial time. Thus, $\mathcal{P}$ is the collection of all "easy" problems.

2. $\mathcal{NP}$ is the collection of all (decision) problems whose solutions, though perhaps not easy to construct, but can be checked in polynomial time. $\mathcal{NP}$ contains many problems that are not known to be in $\mathcal{P}$. Examples include TRAVELING SALESMAN, SATISFIABILITY, and INDEPENDENT SET. Huge amount of efforts has been paid trying to develop polynomial-time algorithms for these problems, but all failed. A common belief is that these problems are hard and do not belong to $\mathcal{P}$, i.e., $\mathcal{P} \neq \mathcal{NP}$.

3. $Q_1 \leq_m^p Q_2$ means that up to polynomial-time computation, Q_1 is not harder than Q_2.

4. $\mathcal{NP}$-hard problems are those that are not easier than any problems in $\mathcal{NP}$ (up to polynomial-time computation). Based on the common belief given in 2, an $\mathcal{NP}$-hard problem cannot be solved in polynomial time.

Some thing you may want to remember.

1. To show $Q_1 \leq_m^p Q_2$, you need to construct a polynomial-time algorithm that computes a function f such that x is a yes-instance of Q_1 if and only if $f(x)$ is a yes-instance of Q_2.

2. To prove that a problem Q is in $\mathcal{NP}$, you need to construct a polynomial-time algorithm $A(x, y)$ such that for any yes-instance x_1 of Q, there is a y_1 such that $A(x_1, y_1) = 1$, and for any no-instance x_2 of Q, $A(x_2, y) = 0$ for all y.

3. To prove that a problem Q is $\mathcal{NP}$-hard, you need to pick a problem Q_0 that is known to be $\mathcal{NP}$-hard, and show $Q_0 \leq_m^p Q$.

4. To prove that a problem Q is $\mathcal{NP}$-complete, you need to prove both that Q is $\mathcal{NP}$-hard and that Q is in $\mathcal{NP}$.

5. You should remember of the definitions of at least the following $\mathcal{NP}$-complete problems: SATISFIABILITY, INDEPENDENT SET, VERTEX COVER, PARTITION, and SUBSET-SUM.