CSCE 629-601 Analysis of Algorithms

Fall 2022

Instructor: Dr. Jianer Chen
Office: PETR 428
Phone: (979) 845-4259
Email: `chen@cse.tamu.edu`
Office Hours: MWF 3:50pm–5:00pm

Teaching Assistant: Vaibhav Bajaj
Office: EABC 107B
Phone: (979) 739-2707
Email: `vaibhavbajaj@tamu.edu`
Office Hours: T; 2pm-3pm, TR: 4pm-5pm

Quick References on NP-Completeness Theory

Definitions.

1. A problem Q is solvable in **polynomial time** if there is an algorithm that solves the problem Q in time $O(n^c)$, where c is a constant.

2. The class $\mathcal{P}$ consists of all (decision) problems that can be solved in polynomial time. Thus, $\mathcal{P}$ is the collection of all "easy" (or, feasible) problems.

3. $\mathcal{NP}$ is the class of all (decision) problems whose solutions, though perhaps not easy to construct, but can be verified in polynomial time. Formally, a problem Q is in $\mathcal{NP}$ if there is an algorithm $A(x, y)$ on two parameters x and y such that
   (a) If x is a yes-instance of Q, then there is a y such that $A(x, y) = 1$;
   (b) If x is a no-instance of Q, then for all y, $A(x, y) = 0$; and
   (c) $A(x, y)$ runs in time polynomial in $|x|$ (i.e., in time $O(|x|^c)$ for a constant c).

4. For two decision problems Q_1 and Q_2, we say that Q_1 is **polynomial-time reducible to** Q_2, written $Q_1 \leq_m^p Q_2$, if there is a polynomial-time algorithm R such that x is a yes-instance of Q_1 if and only if $R(x)$ is a yes-instance of Q_2.

5. A problem Q is $\mathcal{NP}$**-hard** if for every problem Q' in $\mathcal{NP}$, we have $Q' \leq_m^p Q$. A problem Q is $\mathcal{NP}$**-complete** if it is in $\mathcal{NP}$ and is $\mathcal{NP}$-hard.

6. The Satisfiability problem (SAT): given a CNF formula F, decide if F is satisfiable (i.e., if there is an assignment to F that makes $F = \text{TRUE}$).

7. The Independent-Set problem (IS): given a graph G and an integer k, decide if G contains an independent set I of k vertices (i.e., a set I of k vertices in which no two vertices are adjacent).

8. The Vertex-Cover problem (VC): given a graph G and an integer k, decide if G contains a vertex cover C of k vertices (i.e., a set C of k vertices such that every edge in G has at least one end in C).

9. The Partition problem: given a set $S = \{a_1, a_2, \ldots, a_n\}$ of n integers, can S be partitioned into two sets L and R, i.e., $S = L \cup R$ and $L \cap R = \emptyset$, such that $\sum_{a_i \in L} a_i = \sum_{a_j \in R} a_j$?

10. The problems Satisfiability, Independent-Set, Vertex-Cover, Clique, Partition, Subset-Sum, and Knapsack are all $\mathcal{NP}$-complete.

Some (informal but intuitive and helpful) Statements

1. $\mathcal{P}$ is the collection of all "easy" problems. A problem that cannot be solved in polynomial time (i.e., not in $\mathcal{P}$) is regraded as being hard.

2. When we compare the "hardness" of problems, we compare them "up to polynomial time". Thus, if the complexities of two problems differ by a polynomial factor (i.e., by n^c for a constant c), we would regard them as having the "same" complexity, i.e., they are either both easy or both hard. For example, if problem Q_1 is solvable in time $O(n^2)$ while problem Q_2 is solvable in time $O(n^5)$, then we regard Q_2 as *not* harder than Q_1.

3. $\mathcal{NP}$ is the collection of all (decision) problems whose solutions can be verified "easily." Thus, when we say "a problem Q is in $\mathcal{NP}$," what we really wanted to emphasize is the "easiness" of the problem, i.e., the solutions of the problem are easily verified. The statement has nothing to do with the "hardness" of the problem. In particular, all easy problems (i.e., problems in $\mathcal{P}$) are in $\mathcal{NP}$.

4. $Q_1 \leq_m^p Q_2$ means that Q_1 is not harder than Q_2, or that Q_2 is not easier than Q_1. As a consequence, if Q_1 is hard, then Q_2 is also hard, and if Q_2 is easy, then Q_1 is also easy.

5. A problem is $\mathcal{NP}$-hard if it is not easier than any problems in $\mathcal{NP}$. A problem is $\mathcal{NP}$-complete if it is the hardest problem in $\mathcal{NP}$. There are $\mathcal{NP}$-hard problems that are not in $\mathcal{NP}$ (i.e., they are not $\mathcal{NP}$-complete).

6. The logic of using $\mathcal{NP}$-hardness: Since $\mathcal{NP}$ contains many known problems that seem hard (i.e., we do not know how to solve them in polynomial time), and since an $\mathcal{NP}$-hard problem is not easier than any problem in $\mathcal{NP}$, an $\mathcal{NP}$-hard (including $\mathcal{NP}$-complete) problem is believed to be hard, though there is no formal proof for this.

7. By definition, all (decision) problems in $\mathcal{P}$ are in $\mathcal{NP}$, i.e., $\mathcal{P} \subseteq \mathcal{NP}$. Whether all problems in $\mathcal{NP}$ are easy (i.e., if $\mathcal{P} = \mathcal{NP}$) is the most famous open problem in computer science. It is commonly believed that $\mathcal{P} \neq \mathcal{NP}$. Under this hypothesis, all $\mathcal{NP}$-hard problems are hard (i.e., cannot be solved in polynomial time).

8. To show $Q_1 \leq_m^p Q_2$, you need to construct a polynomial-time algorithm R 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. The algorithm R can be highly non-trivial, and in general heavily depends on the problems Q_1 and Q_2.

9. 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. In most cases, the algorithm A is rather trivial and straightforward.

10. 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 $Q_0 \leq_m^p Q$.

11. 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 $\mathcal{NP}$.

12. Remember the definitions of the following $\mathcal{NP}$-complete problems: Independent-Set, Vertex-Cover, Clique, Partition, Subset-Sum, Knapsack, and Satisfiability.