

CSE 830: Design and Theory of Algorithms
Sample Questions for Final Exam
Fall 2011

Satisfiability

1. Draw the graph that results from the reduction of 3-SAT to independent set for the expression

$$\{x, \neg y, z\}, \{\neg x, y, \neg z\}, \{\neg x, y, z\}, \{x, \neg y, \neg z\}$$

Include the maximum independent set that would be found if we solved the problem, and explain what this means for this instance of the 3-SAT problem. What certificate would be returned?

2. There are many possible variants of the 3-SAT problem, some of which are NP-Complete, while others can be easily solved in polynomial time. It is not always very easy to tell them apart. For each of the following variants, either prove that it is NP-Complete or else describe a polynomial time algorithm that will solve it. (Yes, one of these is *very* hard to prove).

- (a) **Strong 3-SAT**: At least *two* of the three literal in each clause must be true.
- (b) **Reflexive 3-SAT**: For each clause in this problem, a corresponding clause containing the negation of each literal also exists. For example, if one clause is $\{x, \neg y, z\}$, then the clause $\{\neg x, y, \neg z\}$ must also be present.
- (c) **Exclusive 3-SAT**: In each clause, one and only one literal may be true; the other two must both be false.
- (d) **Relaxed 3-SAT**: We no longer mandate that all clauses must contain a true literal; in this variant, we only require that at least *half* of the clauses have a true literal in them.
- (e) **Balanced 3-SAT**: Each variable must appear in the same number of clauses as its negation.

3. In one of your homework problems, you proved that 2-SAT can always be solved in polynomial time. Prove that the 2-SAT problem becomes NP-Complete if we add an extra integer k and ask if a solution exists where at most k variables are marked as true.

4. In the **Majority-Rule k -SAT** problem, a clause is only considered true if *at least half* of the literals included in that clause are true. Prove that this problem is NP-Complete for any k greater than or equal to 4.

5. The normal SAT problem is conjunctive normal form (CNF) – every clause must be satisfied (the clauses are linked by logical *and*) and to satisfy a clause you only need to make one of its literals true (the literals are linked by *or*). **Disjunctive normal form (DNF)** is the opposite: only one clause must be satisfied, but to satisfy that clause, all of its literals must be true.

- (a) Describe an algorithm that will solve DNF-SAT in polynomial time.

- (b) Any logical expression in CNF can be converted to DNF. Prove that this conversion is NP-Hard. Does this mean that the conversion back must be NP-Hard?
- (c) Show that the conversion from CNF to DNF is *not* in NP.

General Reductions

6. Prove that the vertex cover problem remains NP-Complete even when all of the vertices in the graph are restricted to have even degree.
7. You are put in charge of scheduling classes at a High School. For each student, you are given a list of classes that that student is required to take. You must make sure that, in the final schedule, no two classes required by any student occur in the same time slot. Given n students and m classes, design a polynomial-time algorithm that produces a schedule using the minimum number of distinct time slots *or* prove that this problem is NP-Complete.
8. A zoo wants to add a large new exhibit with as many animals as possible, but they don't want the animals to eat each other. Given a Directed Acyclic Graph wherein each vertex represents a type of animal they have access to, and each edge from A to B indicates that A will eat B, design a polynomial-time algorithm to determine the largest number of animals that they can keep together such that they won't eat each other *or* prove that this problem is NP-Complete.
9. In the classic knapsack problem, you are given n items each with a weight w_i , and a value v_i . The goal is to take a collection of items such that their total weight is no greater than k , and their total value is maximized. This problem is NP-Complete in this most general form.
 - (a) Prove that the inverse problem, where we must take *at least* weight k and we want to *minimize* the total value is also NP-Complete.
 - (b) Give a polynomial time algorithm that will solve knapsack if we are allowed to go over total weight k as long as the removal of any single item will drop us below weight k .
10. The **partition problem** asks: Given n positive integers, is it possible to partition them into two disjoint subsets with the same sum? Given that the partition problem is NP-Complete, use it to prove that the Knapsack problem is also NP-Complete.
11. Prove that the **Steiner Tree** problem is NP-Complete using a reduction from vertex cover. The Steiner tree problem asks: given a graph, a collection of key vertices, and an integer k , is it possible to find a sub-tree with only k vertices that connects all of the key vertices?
12. Show that the Subset Sum problem is NP-Complete. In Subset Sum, you are given a set S of n numbers and a target number C , you want to know what subset of S sums to exactly C .

You may assume that you know the Set Partition problem to be NP-Complete. In the Set Partition problem, you are given a set S of n numbers and you want to know if the set can be divided into two sets so that the sum of all elements in the first set is equal to the sum of all elements in the second set.

13. Alexander Harris is planning to travel from Sunnydale California to New York City for a vacation. He has collected k different coupons for various hotel chains, so these are the only hotels that he wants to stay at during his trip. He draws a graph where the n vertices are cities that are connected by m edges indicating pairs of cities he is willing to travel between in a day. The problem of finding the shortest path to his destination is easily solvable in polynomial time by using Dijkstra's algorithm. However, Mr. Harris has an added constraint that he has *limited* coupons, and is only willing to stop in a hotel for which he still has a coupon. Once he runs out of coupons for Holiday Inn (for example) he doesn't want to stop at any more of them for the rest of the trip. Is finding the shortest route, subject to this restriction, an NP-Complete problem? If so, prove it; if not, give a polynomial time algorithm to solve this problem.

Approximation Algorithms

14. You are a time-traveler who travels into the past with an array with n values representing the price that a particular stock will have in the stock market for each of the next n days.

- (a) Design an algorithm that will determine the best day to buy a huge chunk of stock, and the best day to sell, to maximize your profit.
- (b) If you were able to buy and sell an unlimited number of times, design an algorithm that will maximize your profit in this case (assume that there are no overhead costs or taxes)
- (c) Now, assume that you were only allowed to buy and sell a maximum of k times. Design a polynomial time algorithm to solve this problem exactly *or* give a good, polynomial-time approximation algorithm if the problem is NP-Complete. Make sure to tell me which you are doing.

15. Imagine that you constructed an approximation algorithm for the Traveling Salesman Problem that could always calculate a solution that is correct within a factor of $1/k$ of the optimal tour in $O(n \cdot 2^k)$ time. Would you be able to use this approximation algorithm to obtain a "good" solution to all other NP-Complete problems? Explain why or why not.

Algorithmic Theory

16. Give two reasons why it may be preferable to implement a $O(n^2)$ algorithm over an $O(n)$ algorithm that solves the same problem.

17. If you optimize the performance of a program by tuning up the most heavily-used parts of the code, but do not alter the underlying algorithm, would you expect the efficiency gain to be bounded by a constant or could the increase be proportionally greater as the problem size increases? Justify your answer.

18. By using virtual memory, a programmer can build programs and data structures that are larger than the actual RAM available on a machine. Since hard disks are much cheaper than memory, does this mean that the amount of storage used by an algorithm is not important in practice? Give two reasons to justify your answer.

19. Given that you find a polynomial-time reduction from problem A to problem B, which of the following are true?

- a** – if A can be solved in polynomial time, so can B.
- b** – if B can be solved in polynomial time, so can A.
- c** – if A is NP-Hard so is B.
- d** – if B is NP-Hard so is A.

20. Suppose that you are analyzing the Slithy Toves problem, trying to get some idea of its complexity. You are able to reduce Slithy Toves to the Sorting problem by calling your sort() sub-routine a total of $O(n)$ times and also do $O(n)$ additional work. What does this tell you about the algorithmic complexity of the Slithy Toves problem?