CS 170
Spring 2017

Efficient Algorithms and Intractable Problems
Prasad Raghavendra and Sanjam Garg

Homework 9

Instructions: You are welcome to form small groups (up to 4 people total) to work through the homework, but you **must** write up all solutions by yourself. List your study partners for homework on the first page, or "none" if you had no partners.

If using LaTeX (which we recommend), you may use the homework template linked on this Piazza post to get started.

Begin each problem on a new page. Clearly label where each problem and subproblem begin. The problems must be submitted in order (all of P1 must be before P2, etc). For questions asking you to give an algorithm, respond in what we will refer to as the *four-part format* for algorithms: main idea, pseudocode, proof of correctness, and running time analysis.

Read the Homework FAQ Piazza post on Piazza before doing the homework for more explanation on the four-part format and other clarifications for our homework expectations.

No late homeworks will be accepted. No exceptions. This is not out of a desire to be harsh, but rather out of fairness to all students in this large course. Out of a total of approximately 12 homework assignments, the lowest two scores will be dropped.

Special Questions:

- *Shortcut questions*: Shortcut questions are usually easy questions that give you opportunities to practice basic materials. However, we understand that some of you are very familiar with the topics and do not want to spend too much time on easy questions. Therefore, we try to design shortcut questions for this purpose. A shortcut question usually has multiple parts that build upon each other and are ordered by their difficulty level. You can work on those in order or start from wherever you like. However you only need to submit the last part you are able to solve. For example, if a question has 5 parts (a, b, c, d, e), you are confident about part e, you should submit part e without any of the previous four parts. If you are confident about d but not sure about e, you should submit d for grading purposes. Please clearly indicate in your submission which part you are submitting.
- *Redemption questions*: It is important that you carefully read the posted solutions, even for problems you got right. To encourage this, you have the option of submitting a redemption file, a few paragraphs in which you explain, for each problem you choose to cover, what you did wrong and what the right idea was in your own words (not cutting and pasting from the solution!), and appending it to your homework. For example, suppose that as you review your solutions to HW1, you realize you had misunderstood question 3 and answered it incorrectly. You would write down what you just learned, and then submit it in your HW2 assignment the following week. Because these are mainly for your benefit, feel free to format them however is most useful for you.
- *Extra credit questions*: We might have some extra credit questions in the homework for people who really enjoy the materials. However, please note that you should do the extra credit problems only if you really enjoy working on these problems and want an extra challenge. It is likely not the most efficient manner in which to maximize your score.

Due Tuesday April 18, at 11:59am

1. (★ level) Basic Complexity Concepts

a. Suppose we reduce a problem A to another problem B. This means that if we have an algorithm that solves ___, we immediately have an algorithm that solves ___.

b. Again, suppose we show a (polynomial-time) reduction from A to B. This implies that up to polynomial factors, ___ cannot be any easier than ___.

c. Define the class **NP** in one sentence. *There are several correct characterizations. One of them is the most intuitive, and the definition we use in this class. Give this definition.*

d. Define the class **NP**-hard in one sentence.

e. Define the class **NP**-complete in one sentence.

f. Show that for any problem Π in **NP**, there is an algorithm which solves Π in time $O(2^{p(n)})$, where n is the size of the input instance and $p(n)$ is a polynomial (which may depend on Π).

2. (★★ level) Proving NP-completeness by generalization.

Proving **NP**-completeness by generalization. For each of the problems below, prove that it is **NP**-complete by showing that it is a generalization of some **NP**-complete problem we have seen in the lecture or in the book. You only need to show the reduction part in this problem (there is no need to show that the problem is in **NP**).

a. SUBGRAPH ISOMORPHISM: Given as input two undirected graphs G and H, determine whether G is a subgraph of H (that is, whether by deleting certain vertices and edges of H we obtain a graph that is, up to renaming of vertices, identical to G), and if so, return the corresponding mapping of $V(G)$ into $V(H)$.

b. LONGEST PATH: Given a graph G and an integer g, find in G a simple path of length g.

c. MAX SAT: Given a CNF formula and an integer g, find a truth assignment that satisfies at least g clauses.

d. DENSE SUBGRAPH: Given a graph and two integers a and b, find a set of a vertices of G such that there are at least b edges between them.

e. SPARSE SUBGRAPH: Given a graph and two integers a and b, find a set of a vertices of G such that there are at most b edges between them.

3. (★★★ level) Reduction, Reduction, and Reduction

Instruction: In this problem, you will prove three problems are **NP**-Complete. Please be aware that to prove a problem is **NP**-Complete, you need to argue it is in **NP** and then briefly justify a reduction that shows it is **NP**-hard.

1. Alice and Bob go out on a date at a nice restaurant. At the end of the meal, Alice has eaten c_A dollars worth of food, and has in her wallet a set of bills $A = \{a_1, a_2, \ldots, a_n\}$. Similarly, Bob owes the restaurant c_B dollars and has bills $B = \{b_1, b_2, \ldots, b_m\}$. Now, Alice and Bob are very calculating people, so they agree that each of them should pay their fair share (c_A and c_B, respectively).

   One thing they don't mind doing, however, is fairly trading bills. That is, Alice can exchange a subset $A' \subseteq A$ of her bills for for a subset $B' \subseteq B$ of Bob's bills, so long as $\sum_{a \in A'} a = \sum_{b \in B'} b$. Under the above conditions, Alice and Bob wish to find, after trading as many times as desired, subsets of their bills A^*, B^* such that $\sum_{a \in A^*} a = c_A$ and $\sum_{b \in B^*} b = c_B$.

Show that FAIR DATE is **NP**-complete.

Hint: You may assume SUBSET SUM is **NP**-complete.

2. Consider the following problem PUBLIC FUNDS:

   > You are looking to build a new fence for your mansion, to keep out pesky people protesting profligate purchases. You have m bank accounts at your disposal to use to pay for your fence; each account i has a balance of b_i. You must choose one of n options for your fence; each fence j costs c_j dollars. You would like to withdraw from at most k of the bank accounts to build the fence, and due to peculiar UC accounting rules, if you use a particular bank account, you must use the whole balance (all b_m dollars.)
   > Determine whether it is possible to exactly pay for some fence j; that is, whether there is a j between 1 and n such that you can withdraw exactly c_j dollars given the bank account balances $b_1, \ldots, b_m$, the fence costs $c_1, \ldots, c_n$, and k, and if so return the corresponding choice of fence and set of bank accounts that you withdraw from.

   Show that PUBLIC FUNDS is **NP**-Complete.

3. Consider the search problem MAX-ACYCLIC-INDUCED-SUBGRAPH:

   INPUT: A *directed* graph $G = (V, E)$, and a positive integer k.

   OUTPUT: A subset $S \subseteq V$ of size k such that the graph G_S obtained from G by keeping exactly those edges both whose endpoints are in S is a DAG.

   Show that MAX-ACYCLIC-INDUCED-SUBGRAPH is **NP**-complete.

4. (★★★★ level) Graphs and Matrix Multiplication.

Consider the MATRIX MULTIPLICATION problem:

> *Input:* Two $n \times n$ matries M_1 and M_2.
> *Output:* The matrix product $M_1 M_2$.

Suppose there is an algorithm which solves MATRIX MULTIPLICATION in time $t(n)$.

(a) Show that given M and an integer $p > 0$, you can compute M^p in time $O(t(n) \log p)$. You only need to prove a main idea and show how to satisfy the time complexity requirement.

(b) Given a directed graph G in adjacency matrix representation and two nodes s and t, you want to know whether there is a path from s to t. This is called the GRAPH REACHABILITY problem. Show that you can solve GRAPH REACHABILITY in time $O(t(n) \log n)$, where n is the number of nodes in the graph. You need to first give the main idea, and justify the time complexity requirement. Also you need to prove that your algorithm is correct.

Hint 1: It may help to use part (a).

Hint 2: In matrix multiplication, what does $(A^k)_{ij}$ mean?

(c) Now, suppose you know that graph reachability cannot be solved in time $O(T(n) \log n)$. Show that MATRIX MULTIPLICATION cannot be solved in time $O(T(n))$. This should be a very simple proof.

5. (★★★★★ level) Finding Zero(s)

Consider the problem INTEGER-ZEROS.

INPUT: A multivariate polynomial $P(x_1, x_2, x_3, \ldots, x_n)$ with integer coefficients, specified as a sum of monomials.

OUTPUT: Integers $a_1, a_2, \ldots a_n$ such that $P(a_1, a_2, a_3, \ldots, a_n) = 0$.

Show that 3-SAT reduces in polynomial time to INTEGER-ZEROS. (You do not need to show that INTEGER-ZEROS is in **NP**: in fact, it is known *not* to be in **NP**).

Hint 1: Given a 3-SAT formula ϕ in the variables $x_1, x_2, \ldots, x_n$, your reduction f will produce a polynomial P in the same variables such that satisfying assignments correspond to $0,1$ valued zeros of P.

Hint 2: If your polynomial constructed above has exponential amount of terms, it is not optimal! You need to reduce it to polynomial amount. Think about the equality $a^2 + b^2 = 0$ if and only if $a = b = 0$ when a, b are real to achieve it. [1]

[1] Fun fact: This problem INTEGER-ZEROS is *really* hard: the decision version of the problem is *undecidable*. This means that there is provably no algorithm which, given a multivariate polynomial, will decide correctly whether or not it has integer zeros. However, if we relax the problem to ask if there are *real* numbers at which the given polynomial vanishes, then the problem surprisingly becomes decidable! For Blue and Gold jingoists: the above decidability result was proven by Alfred Tarski, a Berkeley professor, in 1949. The undecidability of INTEGER-ZEROS was proven by Yuri Matiyasevich (at the ripe old age of 23!) in 1970, building upon previous work of another Berkeley professor, Julia Robinson.