CS 121 Homework 6: Fall 2020

Some policies: (See the course policy at `http://madhu.seas.harvard.edu/courses/Fall2020/policy.html` for the full policies.)

- **Collaboration:** You can collaborate with other students that are currently enrolled in this course (or, in the case of homework zero, planning to enroll in this course) in brainstorming and thinking through approaches to solutions but you should write the solutions on your own and cannot share them with other students.
- **Owning your solution:** Always make sure that you "own" your solutions to this other problem sets. That is, you should always first grapple with the problems on your own, and even if you participate in brainstorming sessions, make sure that you completely understand the ideas and details underlying the solution. This is in your interest as it ensures you have a solid understanding of the course material, and will help in the midterms and final. Getting 80% of the problem set questions right on your own will be much better to both your understanding than getting 100% of the questions through gathering hints from others without true understanding.
- **Serious violations:** Sharing questions or solutions with anyone outside this course, including posting on outside websites, is a violation of the honor code policy. Collaborating with anyone except students currently taking this course or using material from past years from this or other courses is a violation of the honor code policy.
- **Submission Format:** The submitted PDF should be typed and in the same format and pagination as ours. Please include the text of the problems and write **Solution X:** before your solution. Please mark in gradescope the pages where the solution to each question appears. Points will be deducted if you submit in a different format.
- **Late Day Policy:** To give students some flexibility to manage your schedule, you are allowed a net total of **eight** late days through the semester, but you may not take more than **two** late days on any single problem set. No exceptions to this policy.

By writing my name here I affirm that I am aware of all policies and abided by them while working on this problem set:

Your name: (Write name and HUID here)

Collaborators: (List here names of anyone you discussed problems or ideas for solutions with)

No. of late days used on previous psets (not including Homework Zero):
No. of late days used after including this pset:

Special notes for this homework: For your convenience, here's a list of some problems you can assume are in **P**...

1. Linear Programming: given a set of linear inequalities (like $2x_1 - 3x_2 \leq 5$), is there a real solution to all of them?
2. 2-SAT: given a 2-CNF formula (an AND of clauses each an OR of at most 2 variables or their negations), is there an assignment of variables that makes it true?
3. 2-coloring: given a graph G, is there a way to color its vertices with 2 colors so that adjacent vertices have different colors?
4. Shortest Path: given a graph G, vertices s and t, and an integer k, is there a path of length at most k from s to t?
5. Min Cut: given a graph G and an integer k, is there a nonempty subset $S \subsetneq V$ with at most k edges from S to $\overline{S}$?

...and here's a list of some problems you can assume are **NP − complete**:

1. Integer Programming: given a set of linear inequalities (like $2x_1 - 3x_2 \leq 5$), is there an integer solution to all of them?
2. 3-SAT: given a 3-CNF formula (an AND of clauses each an OR of at most 3 variables or their negations), is there an assignment of variables that makes it true?
3. 3-coloring: given a graph G, is there a way to color its vertices with 3 colors so that adjacent vertices have different colors?
4. Longest Path: given a graph G, vertices s and t, and an integer k, is there a path of length at least k from s to t? (A path is a sequence of *distinct* vertices with each consecutive pair adjacent.)
5. Max Cut: given a graph G and an integer k, is there a nonempty subset $S \subsetneq V$ with at least k edges from S to $\overline{S}$?

Questions

Please solve the following problems. Some of these might be harder than the others, so don't despair if they require more time to think or you can't do them all. Just do your best. Also, you should only attempt the bonus questions if you have the time to do so. If you don't have a proof for a certain statement, be upfront about it. You can always explain clearly what you are able to prove and the point at which you were stuck. You can always simply write **"I don't know"** and you will get 15 percent of the credit for the problem. If you are stuck on this problem set, you can use Piazza to send a private message to all staff.

Note on reading the textbook: If you are stuck on some of the problems, try consulting the book to 1) understand the concepts the question is referencing, and 2) review the way similar theorems are proved in the book.

Problem 0 (5 points): True or False: I have completed the midterm 2 feedback survey. (True worth 5 points, False, or I don't know worth 0 points.)

Problem 1.1 (10 points): Suppose that you are in charge of scheduling courses in computer science in University S. In University S, computer science students wake up late, and have to work on their startups in the afternoon, and take long weekends with their investors. So you only have two possible slots: you can schedule a course either Monday-Wednesday 11am-1pm or Tuesday-Thursday 11am-1pm.

Let $SCHEDULE_S : \{0,1\}^* \to \{0,1\}$ be the function that takes as input a list of courses L and a list of *conflicts* C (i.e., list of pairs of courses that cannot share the same time slot) and outputs 1 if and only if there is a "conflict free" scheduling of the courses in L to the two slots, where no pair in C is scheduled in the same time slot. (We model the course list L as just a list of strings, with the conflict lists C as a list of pairs of strings.)

Prove that $SCHEDULE_S \in \mathbf{P}$. As usual, you do not have to provide the full code to show that this is the case, and can describe operations as a high level, as well as appeal to any data structures or other results mentioned in the book or in lecture. Note that to show that a function F is in $\mathbf{P}$ you need to **(1)** present an algorithm A, **(2)** prove that A computes F in polynomial time, and **(3)** prove that A runs in polynomial time. See footnote for hint.[1]

Solution 1.1:

Problem 1.2 (10 points): Consider the same question but now at university H where there is a third course slot on Monday-Wednesday from 11pm till 1am. Let $SCHEDULE_H : \{0,1\}^* \to \{0,1\}$ be the function as above that takes as input a list of courses L and a list of *conflicts* C and outputs 1 if and only if there is a "conflict free" scheduling of the courses in L to the three slots of H. Prove that $SCHEDULE_H$ is $\mathbf{NP}$-complete.

Solution 1.2:

Problem 2 Recall that $\mathbf{P}$, $\mathbf{NP}$, and other complexity classes are sets of *Boolean* functions $f : \{0,1\}^* \to \{0,1\}$; for instance, 3SAT returns 1 iff there exists a satisfying assignment to its input formula. In this problem, we want a polynomial-time algorithm to *find* a satisfying assignment.

Problem 2.1 (5 points) If $\phi(x_0, x_1, \ldots, x_{n-1})$ is a 3-CNF formula and $i \in \{0,1\}$, let $\phi_i(x_1, \ldots, x_{n-1}) = \phi(i, x_1, \ldots, x_{n-1})$. Show that, given $3SAT(\phi_0)$ and $3SAT(\phi_1)$, you can calculate $3SAT(\phi)$.

Problem 2.2 (15 points) Suppose you have a polynomial-time algorithm M that computes 3SAT. Give a polynomial-time algorithm M' that, on input a 3-CNF formula $\phi(x_0, x_1, \ldots, x_{n-1})$, returns a satisfying assignment if one exists, or 0 if none does.

Problem 2.3 (Bonus, 0 points (hard)) Give an algorithm M' with the following property: If it is true that 3SAT $\in P$, then, on input a satisfiable 3-CNF formula $\phi(x_0, x_1, \ldots, x_{n-1})$, M' returns a satisfying assignment in polynomial time. (If no satisfying assignment to its input formula exists, M' may not halt in polynomial time.)

Problem 2.4 (25 points) Suppose you have a polynomial-time algorithm M that computes Longest Path. Give a polynomial-time algorithm M' that, on input a graph G, vertices s and t, and an integer k, returns a path of length at least k from s to t if one exists, or 0 if none does.

[1] Try to see how you model the setting mathematically, and whether it amounts to questions about objects we have looked at before.

Question 3.1 (15 points): k-1S-POSITIVE-3SAT is the following function: the input is a CNF formula φ where each clause is the OR of one to three variables (*without negations*), and a number $k \in \mathbb{N}$. For example, the following is a valid input: $(\varphi = (x_5 \vee x_2 \vee x_1) \wedge (x_1 \vee x_3) \wedge (x_2 \vee x_4 \vee x_0), k = 2)$. The output is 1 if and only if there exists a satisfying assignment to φ in which exactly k of the variables get the value 1. For example, for the formula φ above, k-1S-POSITIVE-3SAT$(\varphi, 2) = 1$ since the assignment $(1, 1, 0, 0, 0, 0)$ satisfies all the clauses. However k-1S-POSITIVE-3SAT$(\varphi, 1) = 0$ since there is no single variable appearing in all clauses.

Prove that k-1S-POSITIVE-3SAT is **NP**-complete. See footnote for hint.[2]

Solution 3.1:

Question 3.2 (15 points): In the *employee recruiting problem* we are given a list of potential employees, each of which has some subset of m potential skills, and a number k. We want a team of k employees such that for every skill there is at least one member of the team with it.

For example, if Alice has the skills "C programming", "NAND programming" and "Solving Differential Equations", Bob has the skills "C programming" and "Solving Differential Equations", and Charlie has the skills "NAND programming" and "Coffee Brewing", then if we want a team of $k = 2$ people that covers all $m = 4$ skills, we could hire Alice and Charlie.

Define the function EMP s.t. on input the skills L of all potential employees (in the form of a sequence L of n lists $L_1, \ldots, L_n$, each containing distinct numbers between 0 and m), and a number k, $EMP(L, k) = 1$ if and only if there is a subset S of k potential employees such that for every skill j in $[m]$, there is at least one employee in S that has the skill j.

Prove that EMP is **NP**-complete. You can use the result of the previous subquestion even if you didn't prove it. See also footnote.[3]

Solution 3.2:

Problem 4 (20 points): Prove that the "balanced variant" BMC of the maximum cut problem is **NP**-complete: given a graph G and an integer k, $BMC(G, k) = 1$ iff there is a subset $S \subsetneq V$ of size $|V|/2$ with at least k edges from S to $\overline{S}$.

Solution 4:

Problem 5.1 (20 points): The input to the CubicSAT function is m polynomials $P_0, \ldots, P_{m-1}$ on n variables $x_0, \ldots, x_{n-1}$. Each polynomial is a cubic equation in the variables, i.e., the degree of every monomial is at most 3. Also, the coefficients are integers in the range $\{-n, \ldots, n\}$. (Each such polynomial can be expressed as a list of $O(n^3)$ integers - why?). CubicSAT$(P_0, \ldots, P_{m-1}) = 1$ iff there exist integers $a_0, \ldots, a_{n-1}$ such that for every $j \in [m]$, $P_j(a_0, \ldots, a_{n-1}) = 0$. Prove that CubicSAT is NP-hard.

---

[2] **Hint:** You can use a direct reduction from $3SAT$ or a reduction from a function that was shown to be **NP**-complete in either the textbook, lecture, or section. If you go via the direct reduction route, you might want to try to transform a SAT instance on n variables $x_0, \ldots, x_{n-1}$ to a k-1S-POSITIVE-3SAT instance on $2n$ variables $w_0, \ldots, w_{2n}$ so that for every $i \in [n]$, the variable w_{2i} will correspond to x_i and the variable w_{2i+1} will correspond to $\neg x_i$. You can try to find ways to constrain a weight n solution so that exactly one of the variables w_{2i} and w_{2i+1} will equal 1 for every $i \in [n]$.

[3] You can assume that lists (some of which could potentially be empty) are represented as strings in some reasonable way, and for every skill j there is some employee with the skill j, and so in particular both n and m are smaller than the length of the input as a string of bits.