

HOMEWORK 08, DUE MARCH 18 (AND MARCH 25)

You must prove your answer to every question.

Problem 1 (Problem 7.33 of Sipser). (10) Describe the error in the following fallacious “proof” that $P \neq NP$. Consider an algorithm for SAT: “On input ϕ , try all possible assignments to the variables. Accept if any satisfy ϕ .” This algorithm clearly requires exponential time, thus SAT has exponential time complexity. Therefore SAT is not in P. Because SAT is in NP, it must be true that P is not equal to NP.

Solution. This argument only shows that a particular algorithm for solving SAT requires exponential time. In order to prove $P \neq NP$, we will have to show that no other algorithm for solving SAT runs in polynomial time either.

Problem 2. (10) Consider the following problem: given a finite set of integers in binary form, one has to partition this set into two subsets such that the sum of numbers within each subset is the same. Prove that this problem is NP-complete. You can use the fact that SUBSET-SUM is NP-complete.

Solution. Note first that the partition problem is in NP. Indeed, the partition (say, the 0-1 sequence showing whether each number is in the first set or the second one) serves as a certificate.

To show that the problem is NP-complete, we will construct a reduction from an instance of SUBSET-SUM to the partition problem. Let (a_1, \dots, a_n, b) be an instance of the subset sum problem: we want to know whether there is a subset X of $\{1, 2, \dots, n\}$ with $\sum_{i \in X} a_i = b$. The idea is to add some extra numbers in such a way that the partition problem for the new set of numbers has a solution if and only if the original instance of the subset-sum problem has. The numbers we add should force the breakup of the sequence a_1, \dots, a_n in such a way that the sum of one part is b . Let $a = a_1 + \dots + a_n$. If $a < b$ we know there is no solution without any reduction, so assume $a \geq b$. Consider the sequence of numbers

$$S = (a_1, \dots, a_n, a - b, b).$$

If there is a sum $\sum_{i \in X} a_i = b$ then $a - b + \sum_{i \in X} a_i = a$, and $b + \sum_{i \notin X} a_i = a$, so the set S can be partitioned. In order for this to be a reduction, we would also have to show that every partitioning of S leads to a sum $\sum_{i \in X} a_i = b$. But this is not true since there is a trivial partitioning: just put $a - b$ and b on one side and everything else on the other side. To prevent this, we take the sequence

$$S' = (a_1, \dots, a_n, a - b + 1, b + 1).$$

In each partitioning of this sequence, the last two numbers must be on different sides, therefore it forces the existence of a sum $\sum_{i \in X} a_i = b$

Problem 3. (10) Show that if TAUTOLOGY is in NP then $NP = Co-NP$.

Solution. Suppose that TAUTOLOGY is in NP. Then \overline{SAT} (non-satisfiability) is also in NP: indeed, a certificate for $\neg\phi$ being a tautology is a certificate for ϕ being in \overline{SAT} . Let L be an arbitrary NP problem, then it can be reduced to SAT via a polynomial function f . So, $w \in \overline{L}$ if and only if $f(w) \in \overline{SAT}$. Therefore if there

are certificates for $f(w) \in \overline{\text{SAT}}$, these are also certificates for $w \in \overline{L}$, and so \overline{L} is in NP.

Problem 4. (25) This problem is due only on March 25.

Write a C (or C++ or Java or C[#]) function $I(A, n)$ (not pseudocode!) that takes, as input, the adjacency matrix A of a graph of size n ($A[i, j] = 1$ if there is an edge from i to j and 0 otherwise) and returns a maximum-size independent set of G (in a dynamic array). Estimate the number of steps of your function.

Write a driving program for your function and test it. Submit the program on paper only. You must also explain well, how it works. [Hint: make the function $I(A, n)$ (or a function it uses) recursive.]