ShanghaiTech University

CS240 Algorithm Design and Analysis

Fall 2021

Problem Set 3

Due: 23:59, Nov.25, 2021

1. Submit your solutions to Gradescope (www.gradescope.com).
2. In "Account Settings" of Gradescope, set your FULL NAME to your Chinese name and enter your STUDENT ID correctly.
3. If you want to submit a handwritten version, scan it clearly. Camscanner is recommended.
4. When submitting your homework, match each of your solution to the corresponding problem number.

Note:

To show that any problem A is NP-Complete, we need to show four things:

(1) there is a non-deterministic polynomial-time algorithm that solves A, i.e., $A \in$ NP,

(2) any NP-Complete problem B can be reduced to A,

(3) the reduction of B to A works in polynomial time,

(4) the original problem A has a solution if and only if B has a solution.

Problem 1:

"Given numbers $s_1, s_2..s_n$, is there a subset that adds up to exactly $\frac{\sum_1^n s_i}{2}$?" Show that the problem is NP-Complete.

Solution:

To show that any problem A is NP-Complete, we need to show four things:

(1) there is a non-deterministic polynomial-time algorithm that solves A, i.e., $A \in$ NP,

(2) any NP-Complete problem B can be reduced to A,

(3) the reduction of B to A works in polynomial time,

(4) the original problem A has a solution if and only if B has a solution.

We now show that SET-PARTITION is NP-Complete.

(1) $SET - PARTITION \in NP$: Guess the two partitions and verify that the two have equal sums.

(2) Reduction of SUBSET-SUM to SET-PARTITION: Recall SUBSET-SUM is defined as follows: Given a set X of integers and a target number t, find a subset $Y \subseteq X$ such that the members of Y add up to exactly t. Let s be the sum of members of X. Feed $X' = X \cup \{s - 2t\}$ into SET-PARTITION. Accept if and only if SET-PARTITION accepts.

(3) This reduction clearly works in polynomial time.

(4) We will prove that $\langle X, t\rangle \in$ SUBSET-SUM iff $\langle X'\rangle \in$ SET-PARTITION. Note that the sum of members of X' is $2s - 2t$. $\Rightarrow$: If there exists a set of numbers in X that sum to t, then the remaining numbers in X sum to $s - t$. Therefore, there exists a partition of X' into two such that each partition sums to $s - t$. $\Leftarrow$: Let's say that there exists a partition of X' into two sets such that the sum over each set is $s - t$. One of these sets contains the number $s - 2t$. Removing this number, we get a set of numbers whose sum is t, and all of these numbers are in X.

Problem 2:

Consider the $CLOSESAT$ problem, which is similar to the SAT problem except that you need to satisfy **n-1** clauses instead of **n** clauses, where n is the number of clauses. Notice that we don't set any limits on the number of variables in each clause. Show that the $CLOSESAT$ problem is NP-Complete.

Solution:

Given an assignment, it is in polynomial time to verify if it satisfies $n-1$ clauses in $O(n)$, so $CLOSESAT$ is NP. Then, we reduce from $3-SAT$ to $CLOSESAT$. Let ϕ represents the original CNF in $3-SAT$, then create a new variable α and conjunct it and its negation to ϕ. Now we construct an input $\phi \cap \alpha \cap \bar{\alpha}$ for $CLOSESAT$. Clearl the construction is in polynomial time $O(1)$.

$\Rightarrow$

If the $3-SAT$ has a truth assignment, since one of α and $\bar{\alpha}$ must be true, $\phi \cap \alpha \cap \bar{\alpha}$ only has one false, which satisfy the $CLOSESAT$ problem. So if $3-SAT$ has a solution, then $CLOSESAT$ has a corresponding solution.

$\Leftarrow$

If the $CLOSESAT$ has a solution, since one of α and $\bar{\alpha}$ must be false, so all the rest of clause must be true, which includes truth assignment for original $3-SAT$ problem. Now, $3-SAT \leq_p CLOSESAT$, so the problem is NP-Complete.

Problem 3:

Suppose you are going to schedule courses in SIST and try to make the number of conflicts no more than K. You are given 3 sets of inputs: C = {...}, S = {...}, R = {{...}, {...}, ...}.

C is the set of distinct courses. S is the set of available time slots for all the courses. R is the set of requests from students, consisting of a number of subsets, each of which specifies the courses a student wants to take. A conflict occurs when two courses are scheduled at the same slot(same time) even though a student requests both of them. Prove this schedule problem is NP-complete.

Example:

K = 1; C = {a, b, c, d}, S = {1, 2, 3}, R = {{a, b, c}, {a, c}, {b, c, d}}

An acceptable schedule is:

a - 1; b - 2; c, d - 3;

Here only one conflict occurs.

Solution:

Firstly, for any given schedule as a certificate, we can traverse every student's requests and check whether the courses in his/her requests conflicts and count the number of conflicts, and at last check if the total number is fewer than K, which can be done in polynomial time. Thus the given problem is in NP.

We choose 3-Coloring problem which is a NP-complete problem. For any instance of 3-Coloring problem with graph G, we construct an instance of the

given problem: let every node v becomes a course, thus construct C; let every edge (u, v) becomes a student whose requests is $\{u, v\}$, thus construct R; let each color we use becomes a slot, thus construct S; at last let K equals to 0 .

We now prove G is a yes-instance of 3-Coloring if and only if (C, S, R, K) is a yes-instance of the given problem:

$\Rightarrow$: if G is a yes-instance of 3 Coloring, then schedule the courses according to their color. Since for each edge $(u, v), u$ and v will be painted with different color, then for each student, his/her requests will not be scheduled to the same slot.

$\Leftarrow$: (C, S, R, K) is a yes-instance of the given problem, then painting the nodes in G according to their slots. Since $K = 0$, then for every student, there is no conflict between their requests, which suggests that for every edge $(u, v), u$ and v will not be painted with the same color.

Problem 4:

The binary quadratic programming problem can be stated as follows. Given a martix $A \in \mathbb{Z}^{m \times n}$ and a vector $b \in \mathbb{Z}^m$, is there an $x \in \{0, 1\}^n$ such that $Ax \leq b$? (Note: $x \in \{0, 1\}^n$ means x is a vector with n elements and each element is either 0 or 1) Hint: Reduction from 3-SAT

Solution:

1. Given x as a certifier, we can evaluate Ax in polynomial time and compare it with b. So it is in NP.

2. For a 3-SAT problem with $\{C_1, ..., C_m\}$ and $x_1, ..., x_n$, we can construct A and b as following:

- $A_{ij} = \begin{cases} -1 & x_j \text{ appeared in } C_i \\ 1 & \neg x_j \text{appeared in } C_i \\ 0 & otherwise \end{cases}$
- $b = (k_1 - 1, ..., k_m - 1)$, k_i is the number of negative variables in C_i.

This construction can be done in polynomial time as we can calculate each value in A and b in polynomial time.

"$\Rightarrow$": If we have a no-instance for 3-SAT, then let $x = (x_1, ..., x_n)$,$x_i = 1$ if x_i (in clauses) is true. Assume C_i is not satisfied, then the row corresponding to C_i (named "r_i") will satisfy $r_i x > k_i - 1$, as that:

$$\begin{cases} -1 \times 0 = 0 & \text{happens on } r_i[j]x[j] \text{ if } x_j \text{ is false} \\ 1 \times 1 = 1 & \text{happens on } r_i[j]x[j] \text{ if } \neg x_j \text{ is included(so it must be ture)} \end{cases}$$

Then

$$r_i x = \#\text{ of negative variables} > k_i - 1 = \#\text{ of negative variables} - 1$$

. So it's also a no-instance for this problem.

"$\Leftarrow$":If we have a no-instance for this problem, then we let the 3-SAT instance be $x_i = true$ if $x_i = 1$ in x and $x_i = false$ otherwise. Assume the ith element in Ax is a_i and $a_i > k_i - 1$. "-1×0" and "1×1" situation leads to no-instance of 3-SAT. We only need to prove "1×0" and "-1×1".

If there are no "1×0" and "-1×1", a_i will equals to k_i as the same analysis in "$\Rightarrow$", and:

- "1×0" increase k_i, but a_i keeps the same,
- "-1×1" decrease a_i, but k_i keeps the same.

In other words $a_i \leq k_i$, $a_i = k_i$ happens iff only "1×1", "-1×0" appear.

So "1×0", "-1×1" leads to a contradiction of $a_i > k_i - 1$. And "-1×0", "1×1" leads to no-instance of 3-SAT.

So this problem is also NPC.

Problem 5:

SIST allows students to work as TAs but would like to avoid TA cycles. A TA cycle is a list of TAs (A1, A2,...,Ak) such that A1 works as a TA for A2 in some course, A2 works as a TA for A3 in some course, $\cdots$, and finally Ak works as a TA for A1 in some course. We say a TA cycle is simple if it does not contain the same TA more than once. Given the TA arrangements of SIST, we want to find out whether there is a simple TA cycle containing at least K TAs. Prove this problem is NP-complete.

Solution:

1. We can use a directed graph to show this problem. Given a directed longest simple cycle, we can check it whether include K nodes in this cycle or not, and it will spend in polynomial time, so the given problem is in NP.

2. We reduce Directed-Ham-Cycle to this problem. Given a graph G(V, E) that is instance of Directed-Ham-Cycle, define an instance of longest simple

cycle containing the same graph, and k = $|V|$, the number of vertices in G. This takes in polynomial time.

3. if a graph G contains a Hamiltonian cycle, then this cycle is a simple cycle with $|V|$ vertices, so the given problem is true, there has a TAs cycle with more than k TAs. If there is no Hamiltonian cycle, then there is no simple cycle with $|V|$ or more vertices, so there is not k TAs contain a TAs cycle.

So, the given problem is in NP-hard, of course in NP-complete.

Problem 6:

Given a set E and m subsets of $E, S_1, S_2, ...S_m$, is there a way to select k of the m subsets such that the selected subsets are pairwise disjoint? Show that this problem is NP complete.

HINT: Reduction from Independent Set.

Solution:

1. Given an input $S = S_1, S_2...S_m$, where S_i is subset, we can check each item in S_i with all other sub-item in S to check if there is two same item in $O\left(n^2\right)$. Thus the problem is in NP.

2. We choose independent set. Given an independent set $G = (V, E)$, we construct $PD = (S, K)$ for S_i in $\{s\}$, where S_i = the set of all edges connected to v_i, and K means the number of nodes in independent set G.

The constructing can be done in $O\left(len\left(v\right) * len\left(E\right)\right)$

3. Then prove: G is a yes-instance of independent set iff PD is yes-instance of pairwise disjoint,