Problem Set 6

CMSC 27230: Honors Theory of Algorithms

Assigned February 20, Due February 27

Problem 1: NP (40 points)

For each of the following yes/no problems, either show that the problem is in NP or explain why we should believe that it is not in NP. Hint is available.

(a) Fixed-length path: Given a directed graph G, two vertices $s, t \in V(G)$, and a natural number $k \in \mathbb{N}$, is there a path of length exactly k from s to t? Note that a path is not allowed to visit any vertex more than once.

(b) Longest Increasing Subsequence: Given a sequence of distinct numbers $x_1, \ldots, x_n$ and a natural number $k \in \mathbb{N}$, is the length of the longest increasing subsequence of numbers equal to k?

For example, if we have the sequence of numbers $8, 5, 9, 2, 6, 3, 4$ and $k = 2$ then the answer would be no because the longest increasing subsequence is $2, 3, 4$ which has length 3.

(c) Chromatic Number: Recall that the chromatic number $\chi(G)$ of a graph G is the minimum $k \in \mathbb{N}$ such that it is possible to give each vertex $v \in V(G)$ a color such that no two adjacent vertices have the same color. Given a graph G and a natural number $k \in \mathbb{N}$, is $\chi(G) = k$?

(d) Balanced Teams: Given $2n$ players with skill levels $x_1, \ldots, x_{2n} \in [2^{100n}]$, is it possible to divide the players into teams of size n such that the total skill level of each team is the same? In other words, can you find $A, B \subseteq [2n]$ such that $|A| = |B| = n$, $A \cap B = \emptyset$, and $\sum_{i \in A} x_i = \sum_{j \in B} x_j$?

Note: For each of these problems which you believe is not in NP, you should explain why you believe the problem is hard and why it is difficult to verify a yes answer.

Remark 0.1. *One way to give strong evidence that a problem is not in NP is to show that the problem is co-NP-hard. You do not need to do this, though you are welcome to do so.*

Problem 2: Job Distribution (30 points)

You are organizing a big convention and there is plenty of work to do. There are a total of m jobs which need to be done, each of which is of one of k types. Fortunately, you have n volunteers, but each volunteer is only willing to do a certain number of each type of job (doing the same thing over and over again gets boring) and a certain number of jobs in total. How can you assign your volunteers to the jobs to satisfy all of these constraints?

Example 0.2. *Consider the case when you have three volunteers, Margaret, Nancy, and Owen, there are* 10 *jobs of type* 1*,* 15 *jobs of type* 2*, and* 20 *jobs of type* 3*, and your volunteers' willingness to work is as follows:*

1. *Margaret is willing to do* 8 *jobs of type* 1*,* 10 *jobs of type* 2*,* 5 *jobs of type* 3*, and a maximum of* 15 *jobs overall.*
2. *Nancy is willing to do* 3 *jobs of type* 1*,* 3 *jobs of type* 2*,* 12 *jobs of type* 3*, and a maximum of* 16 *jobs overall.*
3. *Owen is willing to do* 1 *job of type* 1*,* 3 *jobs of type* 2*,* 14 *jobs of type* 3*, and a maximum of* 16 *jobs overall.*

If so, we can complete all of the jobs by having Margaret do 6 *jobs of type* 1 *and* 9 *jobs of type* 2*, having Nancy do* 3 *jobs of type* 1*,* 3 *jobs of type* 2*, and* 9 *jobs of type* 3*, and having Owen do* 1 *job of type* 1*,* 3 *jobs of type* 2*, and* 11 *jobs of type* 3*.*

We can express this problem mathematically as follows. We are given non-negative integers $\{a_i : i \in [k]\}$, $\{b_{ij} : i \in [k], j \in [n]\}$, and $\{c_j : j \in [n]\}$ where a_i is the number of jobs of type i that need to be done, b_{ij} is the maximum number of times volunteer j is willing to do a job of type i, and c_j is the maximum number of jobs volunteer j is willing to do. We want to find non-negative integers $\{x_{ij} : i \in [k].j \in [n]\}$ such that:

1. For all $i \in [k]$, $\sum_{j=1}^{n} x_{ij} = a_i$ (all of the jobs get done).
2. For all $i \in [k]$ and $j \in [n]$, $x_{ij} \leq b_{ij}$ (no volunteer does too many jobs of a given type).
3. For all $j \in [n]$, $\sum_{i=1}^{k} x_{ij} \leq c_j$ (no volunteer does too many jobs).

Give a $poly(k, n, m)$ time algorithm to solve this problem (where $m = \sum_{i=1}^{k} a_i$ is the total number of jobs which need to be done), explain why your algorithm is correct, and analyze its runtime.

Problem 3: Event Supervision (30 points)

You are organizing a convention where there n events. You have m volunteers, each of whom will be at some subset of the events. Is it possible to promote k of the volunteers to head volunteers so that every event has at least one head volunteer?

Example 0.3. *Let's say that $k = 2$, you have four volunteers, Wendy, Xavier, Yasmin, and Zachary, there are 6 events, and the following volunteers will volunteer for each event:*

1. *Xavier, Yasmin, and Zachary will volunteer for event 1.*
2. *Xavier and Yasmin will volunteer for event 2.*
3. *Wendy and Zachary will volunteer for event 3.*
4. *Wendy and Yasmin will volunteer for event 4.*
5. *Xavier and Zachary will volunteer for event 5.*
6. *Wendy and Xavier will wolunteer for event 6.*

In this case, you can promote Wendy and Xavier to head volunteers and there will be a head volunteer at each event.

Mathematically, we can express this problem as follows: Given subsets $P_1, \ldots, P_m$ of $[n]$ (where $j \in P_i$ if volunteer i is at event j), is there a subset $I \subseteq [m]$ such that:

1. $|I| \leq k$ (at most k volunteers are head volunteers)
2. $\cup_{i \in I} P_i = [n]$ (together, the head volunteers cover all of the events).

Prove that this problem is NP-complete. Hint is available.

Hints

1. For this problem, you should think about the following questions.

   (a) Is there an easy way to find a path of length exactly k or confirm that a given sequence of edges is a path of length k?

   (b) Is there an easy way to find the longest increasing subsequence or confirm that the longest increasing subsequence has length k?

   (c) Is there an easy way to find $\chi(G)$ or confirm that $\chi(G) = k$?

   (d) Is there an easy way to find balanced teams or to confirm that a given pair of teams A and B are balanced?

3. Reducing vertex cover to this problem works well.