Name: ____________________

Entry number: ____________________

- Always try to give algorithm with best possible running time. The points that you obtain will depend on the running time of your algorithm. For example, a student who gives an $O(n)$ algorithm will receive more points than a student who gives an $O(n^2)$ algorithm.
- You are required to give proofs of correctness whenever needed. For example, if you give an algorithm using network flow for some problem, then you should also give a proof why this algorithm outputs optimal solution.
- You may use any of the following known NP-complete problems to show that a given problem is NP-complete: 3-SAT, INDEPENDENT-SET, VERTEX-COVER, SUBSET-SUM, 3-COLORING, 3D-MATCHING, SET-COVER, CLIQUE.
- **Use of unfair means will be severely penalized.**

There are 8 questions for a total of 30 points.

---

(4) 1. Recall the deterministic algorithm for finding the k^{th} smallest number in an array containing distinct numbers:

> *The algorithm considers groups of 5 elements. It picks the middle element from each group and then finds the median element of these middle numbers. Let p denote the median of the middle numbers. It then partitions the array into two parts: left part consisting of numbers $< p$ and right part consisting of numbers $> p$. It then looks for the appropriate number in either the left or right part depending on the size of these parts.*

Suppose the algorithm considers groups of 3 elements instead of groups of 5 elements. Analyze the running time of this algorithm?

(2) 2. Consider the following problem:

FACTOR: Given integers N, x, determine if N has a non-trivial factor less than x.

It is known that FACTOR $\in NP$ but it is not known if FACTOR is NP-complete. State whether the following statements are true or false or unknown with reasons:

(a) If P $\neq$ NP, then FACTOR cannot be solved in polynomial time.

(b) If P $=$ NP, then any 1024 bit number can be factored within an hour.

(3) 3. Solve the following linear program using the simplex algorithm. Give the value of the variables that maximizes the objective function.

Maximize $3x_1 + 2x_2 + x_3$,
Subject to:

$$\begin{aligned}
x_1 - x_2 + x_3 &\leq 4\\
2x_1 + x_2 + 3x_3 &\leq 6\\
-x_1 \qquad + 2x_3 &\leq 3\\
x_1 + x_2 + x_3 &\leq 8\\
x_1, x_2, x_3 &\geq 0.
\end{aligned}$$

(4) 4. Given n integers $x_1, ..., x_n$ and an integer P, a set $S = \{(i, j) : i < j \text{ and } x_i + x_j \geq P\}$ is said to be *valid pair set* if each i is present in at most one pair in S. Design an algorithm that outputs a valid pair set with largest cardinality.

The Makespan problem Recall the minimum makespan problem that we discussed in class. Consider the following variant of the problem:

> k-MAKESPAN: Given n jobs with integer durations $d_1, ..., d_n$ and an integer D, determine if these jobs can be scheduled on k machines such that the maximum finishing time of any job is $\leq D$.

Let us start analyzing the above problem for different values of k. We note that the 1-MAKESPAN and n-MAKESPAN problems are very easy. The next question asks you to show that the 3-MAKESPAN problem in NP-complete.

(5) 5. Show that 3-MAKESPAN is NP-complete.

Now that we have proved that 3-MAKESPAN is NP-complete, we know that a polynomial time algorithm is unlikely. However, we know that all instances of the problem might not be hard. For example, what about when D is small? The next question asks to give a polynomial time algorithm for problem instances where D is small.

(4) 6. Suppose you are given problem instances of 3-MAKESPAN where $D = O(n)$. Give a polynomial time algorithm for solving the problem. Discuss the running time of your algorithm.

Now consider the optimization version of the problem.

> MIN-3-MAKESPAN: Given n jobs with duration $d_1, ..., d_n$, determine a schedule of these n jobs on 3 machines that minimizes the maximum finishing time of any job.

The optimization version of a problem is usually harder than the decision version. The next question asks you to show this formally.

(2) 7. Show that MIN-3-MAKESPAN is NP-hard.

We have seen that one way to deal with NP-hard problems is to give efficient approximation algorithms. In the lectures we looked at a greedy algorithm that gives 2-approximation. The next question asks you to analyze the approximation factor of a variant of this algorithm.

(6) 8. Consider the following greedy algorithm for the MIN-3-MAKESPAN problem: *Sort the jobs in decreasing order of their duration. Consider jobs in this order and schedule a job on a machine with the smallest current load.* Show that this algorithm gives a (3/2) approximation factor.