We may grade a **subset of the assigned questions**, to be determined after the deadline, so that we can provide better feedback on the graded questions.
Unless otherwise stated, each question requires sufficient justification to convince the reader of the correctness of your answer.
For bonus questions, we will not provide any insight during office hours or Piazza, and we do not guarantee anything about the difficulty of these questions.
We strongly encourage you to typeset your solutions in LaTeX.
If you collaborated with someone, you must state their name(s). You must write your own solution for all problems and may not look at any other student's write-up.

0. If applicable, state the name(s) and uniqname(s) of your collaborator(s).

Solution:

1. Suppose you are organizing a face mask distribution operation. You intend to set up a number of centers, and you want every household in the area to be near a mask distribution center. To model this, consider an undirected, *connected* graph $G = (V, E)$. Represent households by vertices. You may place a distribution center at any vertex, and you would like to place at most k centers, such that every vertex is at most 1 edge away from a distribution center.

   Define

$$\text{MDC} = \{\langle G = (V, E), k\rangle : \exists V' \subseteq V \text{ s.t. } |V'| \leq k, \text{ and } \forall v \in V, \exists c \in V' \text{ s.t. } dist(v, c) \leq 1\}$$

   We can show that MDC is NP-Hard through a reduction with Vertex-Cover-Connected, defined below. Note that Vertex-Cover $\leq_p$ Vertex-Cover-Connected.

$$\text{Vertex-Cover-Connected} = \{\langle G, k\rangle : G \text{ is an undirected, } \textit{connected} \text{ graph with a vertex cover of size at most } k\}$$

   Consider this mapping:

Input: Undirected, connected graph $G = (V, E)$, natural number k
function $f(G = (V, E), k)$
  if $|V| = 1$ **then**
    return $\langle G, 1\rangle$
  initialize new graph $G' = (V', E')$ and set $V' = V$, $E' = E$
  for every edge $(u, v) \in E$ **do**
    create a new vertex w_{uv} in V'
    add edges (w_{uv}, v) and (w_{uv}, u) to E'
  return $\langle G', k\rangle$

   (a) Fill in the blanks: f shows that _____ $\leq_p$ _____.

Solution:

(b) Show that the mapping f is computable in polynomial time.

Solution:

(c) Show that $\langle G, k\rangle \in$ Vertex-Cover-Connected $\iff \langle G', k\rangle \in$ MDC.

Solution:

(d) **Optional:** Show that 3SAT $\leq_p$ MDC.

Solution:

2. Suppose you have a set S of integers. You want to determine if it is possible to partition S into two sets A and B ($A \cup B = S$ and $A \cap B = \varnothing$) such that $\sum_{a\in A} a = \sum_{b\in B} b$.

   The corresponding language is

$$L = \{\langle S\rangle : \text{where } S \subset \mathbb{Z} \text{ and } \exists A, B \subset S \text{ such that } A \cap B = \varnothing, A \cup B = S, \text{ and } \sum_{a\in A} a = \sum_{b\in B} b\}$$

   Give a search-to-decision reduction for L. Given a polynomial-time *decider* for L and a set S where $\langle S\rangle \in L$, find the actual partition (A, B) in polynomial time.

   Hint: Given an instance with $|S| = n$ where S is accepted by a decider, how could you obtain a smaller instance S' with $|S'| = n - 1$, such that S' is still accepted by a decider?

Solution:

3. An important problem in operating systems is figuring out the best way to assign jobs to processors. Suppose we have a set of n execution times $\{t_1, \ldots, t_n\}$, each corresponding to a different job, and m processors. For a given schedule, *make-span* is the total time for all processors to complete their jobs. The goal is to find a schedule with the minimum make-span.

   We claim that the following algorithm A is a 2-approximation of the scheduling problem.

Input: Execution times $\{t_1, ..., t_n\}$, positive integer m
1: **function** A($\{t_1, ..., t_n\}, m$)
2: sort the jobs so that $t_1 \geq \cdots \geq t_n$
3: for each job, assign that job to the machine with the lowest load
4: **return** the final job schedule

(a) Give an example of a set of n execution times $\{t_1, ..., t_n\}$ and a positive integer m such that the output of algorithm A is not optimal.

Solution:

(b) Let OPT be the make-span of an optimal allocation. Describe the relationship of the quantities $\max\{t_1, t_2, \ldots, t_n\}$ and $\frac{1}{m} \cdot \sum_{j=1}^{n} t_j$ to OPT.

Solution:

(c) Prove that A is a 2-approximation.

Hint: Prove that the processor which finishes last takes at most twice the optimal time. Consider how the jobs must have been allocated before this processor's last job was allocated.

Solution:

4. Suppose you are trying to move apartments. You have n items that you wish to pack, represented as a set A containing their sizes, and you have access to bins of capacity v. The goal is minimize b, the number of bins you use.

Formally, we define

$$\textsc{Bin-Packing} = \left\{ \langle v, A, b\rangle : \begin{array}{c} v \text{ is the capacity of each bin, } A \text{ is the set of sizes} \\ \text{for } n \text{ items, and } b \text{ is a natural number where the } n \\ \text{items can be packed using at most } b \text{ bins} \end{array} \right\}.$$

Since Bin-Packing is NP-Complete, it may be very inefficient to try to find the optimal value of b for a very large number of items. Instead, you try using the following greedy algorithm, hoping that you can get close enough to the optimal:

On input $\langle v, A\rangle$:

i. Initialize the set of used bins as $S = \emptyset$

ii. For each item $i \in \{1, \ldots, n\}$, iterate over S and place the item in the first bin that has enough space to fit it.

iii. If no suitable bins are found in the previous step, then get a new bin ($S = S \cup \{s_{\text{new}}\}$) and place the item in this bin.

iv. Return S and $b = |S|$.

Prove that this algorithm is a 2-approximation for Bin-Packing. That is, prove that the value obtained by this algorithm is no more than twice the minimum number of bins, b^*, needed to contain all n items.

(a) Show that $b^* \geq (\sum_{i=1}^{n} A[i])/v$.

Solution:

(b) Show that if $b \leq 1$, then $b = b^*$.

Solution:

(c) Show that if $b > 1$, then $b/2 \leq (\sum_{i=1}^{n} A[i])/v$.
Hint: Think about what it means for two bins to be simultaneously filled to at most half their capacity.

Solution:

(d) Using the above parts, conclude that our algorithm is a 2-approximation.

Solution: