Tutorial 2

1. Suppose you are helping out a university to design their exam correction schedule. The exam has two parts: objective type questions, to be corrected by a computer and subjective type questions to be corrected by the teachers. The university has as many teachers as the number of answer sheets (say n), but only one computer. It has been decided that each teacher i will correct the second part of the ith answer sheet. The second part is to be corrected only after the first part has been corrected.

   For each answer sheet $i \in [n]$, you are given two durations $t_{i,1}$ and $t_{i,2}$, where

   - $t_{i,1}$ indicates the time taken by the computer to correct the first part of the ith answer sheet and
   - $t_{i,2}$ indicates the time taken by the ith teacher to correct the second part of the ith answer sheet.

   The computer can correct only one answer sheet at a time. The teacher i (for any $i \in [n]$) can correct the second part of the ith answer sheets immediately after the computer finishes the first part and she can do so independently of any other teacher.

   Give an algorithm that designs an ordering of the answer sheets to be sent to the computer so that the overall correction time (i.e. the time by which both the parts of all the answer sheets are corrected) is minimised.

2. Consider the following version of the interval scheduling problem. There are n jobs. Each job is described by its start and finish times, i.e. $\forall i \in [n]$, $J_i = (s(i), f(i))$. Each job has a weight assigned to it, say $w(i)$. The problem is to maximise the total weight of the jobs that can be scheduled on a single machine.

   If we set $w(i) = 1$ for each $i \in [n]$, it basically is the interval scheduling problem we saw in class.

   (a) Show that the algorithm designed in class does not give an optimal solution for this version of the problem.

   (b) Suppose a greedy algorithm picks the largest weight job with earliest finish time among the available jobs at each step. Will this strategy work? If so, prove that it works. If not, give a counterexample.

   (c) We will say that a job J overlaps with a job J', denoted as $J||J'$, if $J \cap J' \neq \emptyset$. For a job J_i, we use O_i to denote the quantity $O_i = \sum_{i':J_i||J_{i'}} w(i')$. We call O_i the overlap-weight of J_i. Consider a greedy algorithm that schedules a job with smallest overlap-weightamong the available jobs at each step. Will such a greedy algorithm give the correct answer for the problem? If so, prove that it works. If not, give a counterexample.

3. You are helping Virat Kohli's personal nutritionist with an algorithm to come up with Kohli's diet plan. Due to his current training routine he can eat at most C calories per day. The nutritionist has figured out items $I_1, I_2, \ldots, I_n$ that Kohli can eat (based on his likes/dislikes diet restrictions etc.) which have calories $c_1, c_2, \ldots, c_n$. The likelihood of Kohli enjoying a certain item is given by $\ell_1, \ell_2, \ldots, \ell_n$. You are required to fit as many items as possible in his day to maximise his overall happiness with the diet plan, while keeping the overall intake of calories to at most C.

   Consider the following greedy algorithm.

for $i = 1$ to n **do**
  $v_i \leftarrow \ell_i / c_i$.
end for
Sort them in the non-increasing order to obtain the list $\text{val}_1, \ldots, \text{val}_n$.
If l_i is very large and c_i is small, then val_i will be high.
Let Limit $\leftarrow C$ Plan $\leftarrow \emptyset$.
for $i = 1$ to n **do**
  if $c_i \leq$ Limit **then**
    Limit $\leftarrow$ Limit $- c_i$
    Plan $\leftarrow$ Plan $\cup \{i\}$
  end if
end for
Output Plan

(a) Show that the above algorithm cannot find the optimal diet plan.

(b) Can you suggest any other greedy strategy to come up with the diet plan in the above setting?

(c) * Suppose, items $I_1, \ldots I_n$ are not exactly units, but can be broken into smaller fractions. For example, say I_1 is 200gm rice then one can take 50gm of it, i.e. 1/4th fraction etc. Accordingly the calories in f_ith fraction of item I can be assumed to be $c_i \times f_i$. Consider the following updated greedy algorithm.

for $i = 1$ to n **do**
  $v_i \leftarrow \ell_i / c_i$.
end for
Sort them in decreasing order to obtain the list $\text{val}_1, \ldots, \text{val}_n$.
If l_i is very large and c_i is small, then val_i will be high.
Let Limit $\leftarrow C$ and Plan $\leftarrow \emptyset$.
for $i = 1$ to n **do**
  if $c_i \leq$ Limit **then**
    Limit $\leftarrow$ Limit $- c_i$
    Plan $\leftarrow$ Plan $\cup \{i\}$
  else if **then**
    $f \leftarrow$ Limit$/c_i$.
    Plan $\leftarrow$ Plan $\cup$ f-fraction of I_i.
  end if
end for
Output Plan

Show that the above algorithm gives an optimal solution.

\+ Given a set of n intervals, design greedy algorithms to find

(a) The smallest size subset of intervals such that every interval not in the subset overlaps with at least one interval in the subset.

(b) The smallest size subset of intervals such that every interval is contained in the union of the intervals in the subset.

(c) The largest size subset of intervals such that every interval in the subset overlaps with at most k other intervals in the subset.

4. Let $G = (V, E)$ be an undirected graph. A collection of edges $M \subseteq E$ is called *a matching* if no two edges of M share a common vertex. A matching is called *a perfect matching* if M spans all the vertices. Consider the following greedy algorithm.

Let $M \leftarrow \emptyset$. Let $V(M)$ be the vertices in M.
for each $(u, v) \in E$ **do**
  if $V(M) \cap \{u, v\} = \emptyset$ **then**
    $M \leftarrow M \cup \{(u, v)\}$.
  end if
end for
Output M

(a) Give a graph G such that the above algorithm may not output a perfect matching in G even if it has a perfect matching.

(b) A matching M is called *a maximal matching* if it is the maximal collection of edges that forms a matching. It is easy to see that every perfect matching is also a maximal matching. Give a graph such that has a perfect matching M and a maximal matching M' such that $M' \neq M$.

(c) Prove that the above greedy algorithm finds a maximal matching.

5. Prove or disprove that when all edges in an undirected graph have distinct costs then the minimum spanning tree in the graph is unique.

6. Let $G = (V, E)$ be an undirected graph with edge costs defined by $c : E \to \mathbb{Z}^{>0}$. Let T be a minimum spanning tree of the graph and let P be a minimum cost path between two specific vertices, say s and t. Suppose we square the costs of each edge to form a new graph G' (no other changes are made to G). Prove whether the following statements are true or false.

   (a) T will continue to be the minimum spanning tree of G'.

   (b) P will continue to be the minimum cost path between s and t in G'.