Problem 1. Let S be a set of n intervals $\{[s_i, f_i] | 1 \leq i \leq n\}$, satisfying $f_1 \leq f_2 \leq \ldots \leq f_n$. Denote by S' the set of intervals in S that are disjoint with $[s_1, f_1]$. Prove: if $T' \subseteq S'$ is an optimal solution to the activity selection problem on S', then $T' \cup \{[s_1, f_1]\}$ is an optimal solution to the activity selection problem on S.

Proof. Suppose the otherwise that $T' \cup \{[s_1, f_1]\}$ is not an optimal solution to the activity selection problem on S. Then, we must have $T \subseteq S$ being the optimal solution and $|T| > |T' \cup [s_1, f_1]|$. Since T must include $[s_1, f_1]$ and the intervals in $T \backslash \{[s_1, f_1]\}$ must be disjoint with $[s_1, f_1]$, we must have $|T'| < |T \backslash \{[s_1, f_1]\}|$, which is a contradiction as $|T \backslash \{[s_1, f_1]\}|$ must now be the optimal solution on S'. □

Problem 2. Describe how to implement the activity selection algorithm discussed in the lecture in $O(n \log n)$ time, where n is the number of input intervals.

Solution. Sort all the intervals by their finishing time in ascending order, using $O(n \log n)$ time. Maintain a set S for the intervals selected. During the scan, we also maintain a variable, *last*, for recording the finishing time of the last interval in S (having the maximum finishing time). Initially $S = \{[s_1, f_1]\}$ and $last = f_1$. Then, in $O(1)$ time, we can check comparing *last* and the starting time s_i of the interval $[s_i, f_i]$ $(i \in [2, n])$ currently being scanned if the interval is overlapping with the activities in S. Add the interval to S if there is no conflict and update *last* accordingly. The scanning process can thus be done in $O(n)$ time. Hence, the overall time complexity is $O(n \log n)$ as required.

```
procedure ActivitySelection(S)
    T = {[s_1, f_1]}, last = f_1
    for i ← 2 to n do
        if last < s_i then
            T ← T ∪ {[s_i, f_i]}
        end if
    end for
    return T
end procedure
```

Problem 3. Prof. Goofy proposes the following greedy algorithm to "solve" the activity selection problem. Let S be the input set of intervals. Initialize an empty T, and then repeat the following steps until S is empty:

- (Step 1) Add to T the interval $I = [s, f]$ in S that has the smallest s-value.
- (Step 2) Remove from S all the intervals overlapping with I (including I itself).

Finally, return T as the answer.
Prove: the above algorithm does not guarantee an optimal solution.

Proof. Consider the set $S = \{[1, 20], [2, 3], [4, 5]\}$. The algorithm returns $\{[1, 20]\}$ while the optimal solution should be $\{[2, 3], [4, 5]\}$. □

Problem 4. Prof. Goofy just won't give up! This time he proposes a more sophisticated greedy algorithm. Again, let S be the input set of intervals. Initialize an empty T, and then repeat the following steps until S is empty:

- (Step 1) Add to T the interval I in S that overlaps with the fewest other intervals in S.
- (Step 2) Remove from S the interval I as well as all the intervals that overlap with I.

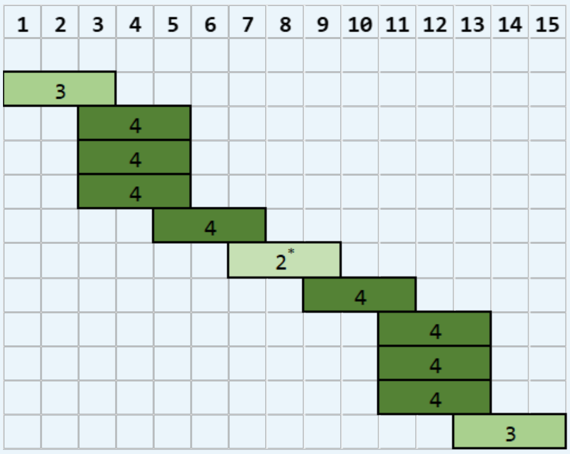
Finally, return T as the answer.
Prove: the above algorithm does not guarantee an optimal solution.

Proof. Let $S = \{[1,5],[2,7],[3,8],[4,9],[6,20],[19,30],[29,40],[30,50],[31,51],[32,52],[41,60]\}$.

Interval $[19,30]$ must be selected since it is the only interval overlapping with 2 other intervals, which is the fewest. This then eliminates $[6,20]$ and $[29,40]$ since they are overlapping with $[19,30]$. This then breaks S into two disjoint sets of intervals, and in the best case only one of each can be selected. This gives a solution of 3 intervals. Instead, the optimal answer should be a set of 4 intervals by selecting $\{[1,5],[6,20],[29,40],[41,60]\}$.

A clearer construction can be given if we allow a multiset of intervals:

$$S = \{[1,3],[3,5],[3,5],[3,5],[5,7],[7,9],[9,11],[11,13],[11,13],[11,13],[13,15]\}.$$



□

Problem 5. **(Fractional Knapsack)**. Let $(w_1, v_1), (w_2, v_2), \ldots, (w_n, v_n)$ be n pairs of positive real values. Given a real value $W \leq \sum_{i=1}^{n} w_i$, design an algorithm to find $x_1, x_2, \ldots, x_n$ to maximize the objective function

$$\sum_{i=1}^{n} \frac{x_i}{w_i} \cdot v_i$$

subject to:

- $0 \leq x_i \leq w_i$ for every $i \in [1, n]$;
- $\sum_{i=1}^{n} x_i \leq W$

Remark: You can imagine, for each $i \in [1, n]$ that the value w_i is the 'weight' of a certain item, and v_i is the item's 'value'. The goal is to maximize the total value of the items we collect, subject to the constraint that all the items must weight no more than W in total. For each item, we are allowed to take only a fraction of it, which reduces its weight and value by proportion.

Solution.

```
procedure FractionalKnapsack(S)
    Sort the n items (w_i, v_i) by descending value-cost rate v_i/w_i
    for i ← 1 to n do
        x_i ← min{W, w_i}, W ← W − x_i
    end for
    return x_1, x_2, ..., x_n
end procedure
```

We prove the correctness of this algorithm.

Observe that we must have $\sum_{i=1}^{n} x_i^* = W$, otherwise we may add a fraction of any remaining item(s) with weight equal to $W - \sum_{i=1}^{n} x_i^*$ to our knapsack which increases the objective function, as the values are all positive.

Without loss of generality, assume that $v_1/w_1 \geq v_2/w_2 \geq \ldots \geq v_n/w_n$. Consider an arbitrary optimal solution $x_1^*, x_2^*, \ldots, x_n^*$.

We may constructively find a better objective value (or no worse than the value given by $x_1, x_2, \ldots, x_n$) and reach a contradiction. Let t be the smallest integer with $x_t^* \neq x_t$. We must have $x_t^* < x_t$ justified by our algorithm, so we let $\Delta = x_t - x_t^*$, and hence $\sum_{i=t+1}^{n} x_i^* = \sum_{i=t+1}^{n} x_i + \Delta$. Thus, we may increase x_t^* to x_t, and reduce a total amount of Δ arbitrary from $x_{t+1}^*, x_{t+2}^*, \ldots, x_n^*$. Noting that $v_i/w_i \geq v_j/w_j$ for any $i > j$, we have

$$\frac{x_t^* + \Delta}{w_t} \cdot v_t + \sum_{i=t+1}^{n} \frac{x_i^* - \Delta_i}{w_i} \cdot v_i \geq \sum_{i=t}^{n} \frac{x_i^*}{w_i} \cdot v_i + \Delta \frac{v_t}{w_t} - \Delta \frac{v_{t+1}}{w_{t+1}} \geq \sum_{i=t}^{n} \frac{x_i^*}{w_i} \cdot v_i.$$

By repeating the above argument, we may adjust the optimal solution to match our algorithm's solution without worsening the objective value. This shows the optimality of our algorithm.

Problem 6. If we run the activity-selection algorithm taught in the class on the following input:

$$S = \{[1, 10], [2, 22], [3, 23], [20, 30], [25, 45], [40, 50], [47, 62], [48, 63], [60, 70]\},$$

then what is the set of intervals returned?

Solution. S is already sorted by ascending finishing time. Therefore, the set of intervals returned is $\{[1, 10], [20, 30], [40, 50], [60, 70]\}$.

Problem 7. The following is another greedy algorithm for the activity selection problem. Initialize an empty T, and then repeat the following steps until S is empty:

- (Step 1) Add to T the interval I with the shortest length.
- (Step 2) Remove from S the interval I, and all the intervals overlapping with I.

Finally, return T as the answer.
Prove: the above algorithm does not always return an optimal solution.

Proof. Consider $S = \{[1, 3], [3, 4], [4, 6]\}$. While the algorithm returns $\{[3, 4]\}$, the optimal solution should be $\{[1, 3], [4, 6]\}$. □

Problem 8. Let $(w_1, v_1), (w_2, v_2), \dots, (w_n, v_n)$ be n pairs of positive real values. Given a real value $W \le \sum_{i=1}^{n} w_i$, we want to find $x_1, x_2, \dots, x_n$ to maximize the objective function

$$\sum_{i=1}^{n} \frac{x_i}{w_i} \cdot v_i$$

subject to:

- $0 \le x_i \le w_i$ for every $i \in [1, n]$;
- $\sum_{i=1}^{n} x_i \le W$

Without loss of generality, assume that $v_1 \ge v_2 \ge \dots \ge v_n$. Consider the algorithm that works as follows:

for $i \leftarrow 1$ to n **do**
  $x_i \leftarrow \min\{W, w_i\}$
  $W \leftarrow W - x_i$
end for

Prove: the above algorithm does not always returns an optimal solution.

Proof. Consider the $n = 3$ pairs $(1, 101), (1, 101), (2, 201)$ and $W = 2$. While the algorithm returns $(x_1, x_2, x_3) = (0, 0, 2)$, but the optimal solution should be $(x_1, x_2, x_3) = (1, 1, 0)$ instead. □

Problem 9. Suppose that there are n gold bricks, where the i-th piece weighs p_i bounds and is worth d_i dollars. Given a positive integer W, our goal is to find a set S of gold bricks such that

- the total weight of the bricks in S is at most W;
- the total value of the bricks in S is maximized (among all the sets S satisfying the first condition).

Assuming $d_1 \geq d_2 \geq \ldots \geq d_n$, let us consider the following greedy algorithm:

$S = \emptyset$
for $i \leftarrow 1$ to n **do**
  if $p_i \leq W$ **then**
    $S \leftarrow S \cup \{p_i\}$
    $W \leftarrow W - p_i$
  end if
end for

Prove: the above algorithm does not guarantee finding the desired set S.

Proof. Suppose we have four items represented in weight-value pairs $(8, 12)$, $(3, 11)$, $(3, 11)$, $(3, 11)$ and $W = 9$. Then, the algorithm returns $\{(8, 12)\}$ with total value 12 while the optimal solution is $\{(3, 11), (3, 11), (3, 11)\}$ with total value 33. $\square$