Problem (Show that being greedy w.r.t. different measures may not yield optimal solutions to the activity-selection problem.)

(a) Consider the problem instance:

\[
\begin{array}{c|c}
\text{a} & \text{b} \\
\hline
1 & 2 \\
\end{array}
\quad
\begin{array}{c|c}
\text{c} & \text{d} \\
\hline
3 & 3 \\
\end{array}
\]

The optimal solution is \( \{a, c, d\} \).
Being greedy by shortest duration chooses \( \{a, b\} \), which is suboptimal.

(b) Consider the problem instance:

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c|c}
\text{a} & \text{b} & \text{c} & \text{d} & \text{e} & \text{f} & \text{g} & \text{h} & \text{i} & \text{j} \\
\hline
\text{b} & \text{d} & \text{e} & \text{f} & \text{g} & \text{h} & i & j & k \\
\end{array}
\]

The overlap counts are:

\[
\begin{array}{ccccccccccc}
a & b & c & d & e & f & g & h & i & j & k \\
\hline
3 & 4 & 4 & 4 & 4 & 2 & 4 & 4 & 4 & 3 \\
\end{array}
\]

The optimal solution is \( \{a, e, g, k\} \).
Being greedy by fewest remaining overlaps chooses \( \{f, a, k\} \), which is suboptimal.
(b) Consider the problem instance:

\[ a \quad -\quad b \quad -\quad c \]

The optimal solution is \[ \{b, c\} \].

Being greedy with respect to increasing start time chooses \[ \{a\} \], which is suboptimal.
Exercise  
(Gas refueling with the fewest stops)

Problem  
Given a fixed route from point A to point B on a map, with known distances between gas stations along the route, determine where to refuel, starting with a full tank, so as to minimize the number of stops. The car can travel n miles on a full tank.

Remarks  
A solution exists if successive gas stations are always at most n miles apart, so we assume this holds.

![Diagram of gas stations and route]

...  
A  g1  g2  g3  ...  gm  B
  g0  s0  s1  s2  ...  sn  gm+1

There is an optimal solution that always fills the tank when it stops, so we represent a solution simply by the subset of stations it stops at.

Greedy procedure

Scan the stations in order from A to B, starting with a full tank, treating A and B as stations.

At station i, if station i+1 can be reached on the current tank, do not stop;

otherwise, stop at station i and fill up.

Analysis

This takes Θ(m) time if the stations are given in order.
Exercise cont!

Correctness

Lemma

For any $1 \leq i \leq m$, let $G$ be the subset of stations $g_i, \ldots, g_i$ chosen by the greedy procedure, and let $g_k$, where $k$, be the next greedy stop.

Suppose there is an optimal solution that agrees with $G$ on $g_i, \ldots, g_i$.

Then there is an optimal solution that agrees with $G \cup \{g_k\}$ on $g_i, \ldots, g_k$.

Proof: Let $G^* \supseteq G$ be an optimal solution that agrees with $G$ on $g_i, \ldots, g_i$.

Let $G'$ be $G^*$ with its steps between $g_i$ and $g_k$ replaced by $g_k$.

$G'$ is a feasible solution that agrees with $G \cup \{g_k\}$ on $g_i, \ldots, g_k$; since it stops no more than $G^*$, it is optimal.

Theorem The greedy procedure finds an optimal solution.

Proof Straightforward from the lemma, using induction on the number of iterations.
(Greedy algorithms) Suppose we have a collection of \( n \) tasks that must be performed. For each task \( i \) we know \( t_i \), the length of time it takes to perform task \( i \). We can perform a task at any point in time that we choose, and we can perform them in any order, but we can only perform one task at a given moment.

The completion time of a task is the time at which we finish performing it. Design an efficient greedy algorithm that finds a sequence in which to perform the tasks that minimizes the average completion time for the \( n \) tasks. More formally, if \( c_i \) is the completion time of task \( i \) for a given sequence, the solution value for that sequence is

\[
\frac{1}{n} \sum_{1 \leq i \leq n} c_i.
\]

Analyze the running time of your algorithm, and prove that it finds an optimal solution using a greedy augmentation lemma of the type given in class.

**Algorithm**

We sort the tasks by increasing running time.

Rename the sorted tasks so that

\[ t_1 \leq t_2 \leq \ldots \leq t_n. \]

We then execute the tasks in this order \( 1, 2, \ldots, n \), starting them at times

\[ 0, t_1, t_1 + t_2, \ldots, t_1 + t_2 + t_3, \ldots, t_1 + \ldots + t_n. \]

(Equivalently, this greedy procedure executes next that task \( i \) that has the smallest \( t_i \) of all tasks not yet executed.)

**Analysis**

Sorting the tasks and determining their start times takes a total of \( O(n \log n) \) time for \( n \) tasks.

**Correctness**

Let a partial solution be a prefix of the listing of tasks in their order of execution.

A partial solution is contained in a complete solution if it is a prefix of the complete solution.
Correctness, cont. 

Lemma

Suppose tasks 1, 2, ..., i form a partial solution contained in an optimal solution.

Let task i+1 be the next task executed by the greedy procedure.

Then partial solution 1, 2, ..., i, i+1 is contained in an optimal solution.

Proof

Let $S^*$ be an optimal solution that contains the partial solution 1, 2, ..., i.

If the next task $S^*$ executes is i+1, the lemma holds.

Suppose instead $S^*$ executes next task $j > i+1$.

Let $k$ be the position in the ordering at which $S^*$ executes task i+1.

Form a new solution $\tilde{S}$ by exchanging the positions of tasks i+1 and j, as follows.

$S^*$

\[
\begin{array}{ccccccc}
1 & 2 & \cdots & i & i+1 & k & n \\
\end{array}
\]

$\tilde{S}$

\[
\begin{array}{ccccccc}
i & 2 & \cdots & i & i+1 & j & \cdots \\
\end{array}
\]

Notice that the average completion time $c(S)$ of a schedule $S$ is,
Proof, cont'd.

\[ c(S) = \frac{1}{n} \sum_{i \in \mathbb{U}} \sum_{j \in S[i]} t \]
\[ = \frac{1}{n} \sum_{i \in \mathbb{U}} (n - i + 1) t_{\leq i} . \]

Since schedules \( S^* \) and \( S \) only differ at positions \( i+1 \) and \( k, \)

\[ c(S^*) - c(S) = \frac{1}{n} \left( ((n-i) t_j + (n-k+1) t_{i+1}) - ((n-i) t_{i+1} + (n-k+1) t_j) \right) \]
\[ = \frac{1}{n} \left( (k - (i+1)) t_j - (k - (i+1)) t_{i+1} \right) \]
\[ = \frac{1}{n} \left( k - (i+1) \right) (t_j - t_{i+1}) \]
\[ > 0 \quad \text{since} \quad t_1 \leq \ldots \leq t_n \]

which implies \( c(S) \leq c(S^*). \)

Thus \( S \) is an optimal solution that contains the partial solution \( 1, 2, \ldots, i+1. \)

\[ \square \]

Theorem. The greedy procedure finds an optimal schedule.

Proof. By the lemma, using induction on the number of iterations. \[ \square \]
Exercise  (Scheduling activities among the fewest halls)

Problem  Given \( n \) activities, with activity \( i \) for \( 1 \leq i \leq n \) described by time interval \([s_i, f_i]\), and \( n \) available halls, assign activities to halls so that

- two activities whose time intervals intersect are never assigned to the same hall, and
- the total number of halls used is minimized.

Definitions

- Let \( N := \{1, 2, \ldots, n\} \).
- A function \( H : A \rightarrow N \) is a partial assignment if
  - \( A \subseteq N \), and
  - \( \forall \) distinct \( i, j \in A \) \((H(i) = H(j) \Rightarrow [s_i, f_i] \cap [s_j, f_j] = \emptyset)\).
- A total assignment is a partial assignment with \( A = N \).
- A total assignment \( H : N \rightarrow N \) extends a partial assignment \( G : A \rightarrow N \) if \( H \) restricted to domain \( A \) agrees with \( G \).

Remarks

- A solution to our problem is a total assignment \( H : N \rightarrow N \) that minimizes \(|H(N)|\). (Here \( H(N) := \{H(i) : i \in N\} \).)
- Our greedy procedure repeatedly extends a partial assignment until it is total.
Exercise contd.

Greedy procedure

(1) Sort the start times $s_1, s_2, \ldots, s_n$ and finish times $f_1, f_2, \ldots, f_n$.

(2) Merge the sorted lists of start and finish times into one sorted list of events, placing start events before finish events in case of ties. Record with each event its activity number and whether it is a start or finish event.

(3) Build a min-heap $H$ of halls $\{1, 2, \ldots, n\}$ prioritized by hall number. ($H$ stores the currently available halls.)

(4) Scan the merged list of events from earliest to latest. For each event $(e, i)$ of type $e$ caused by activity $i$, do the following:

   (a) If $e$ is a start event, set $h(i) := \text{Extract}(H)$ (i.e. assign to $i$ the lowest-numbered available hall).

   (b) If $e$ is an end event, do $\text{Insert}(h(i), H)$ (i.e. free up hall $h(i)$).

(5) Return the assignment $h$.

Analysis

Step (1) takes $O(n \log n)$ time worst-case.
Steps (2) and (3) take $\Theta(n)$ time.
Parts (a) and (b) of Step (4) each take $O(\log n)$ time,
so Step (4) takes $O(n \log n)$ total time.
Thus the whole procedure takes $O(n \log n)$ time.
Exercise cont'd

Correctness

Lemma

Number the activities 1, 2, \ldots, n in order of increasing start time, and let $H$ be the partial assignment of activities $\{1, 2, \ldots, i\}$ obtained by the greedy procedure. Suppose $H$ has an extension to an optimal total assignment. Then $H$ together with the greedy assignment for activity $i+1$ has an extension to an optimal total assignment.

Proof

Let $H^*$ be an optimal total assignment that extends $H$, $h^*$ be $H^*(i+1)$, and $h$ be the greedy assignment for activity $i+1$. If $h^* = h$, the lemma holds.

If $h^* \neq h$, change $H^*$ into $\tilde{H}$ by exchanging halls $h^*$ and $h$ as follows. Whenever $H^*(j) = h^*$ for an activity $j \in \{i+1, i+2, \ldots, n\}$, set $\tilde{H}(j) := h$. Whenever $H^*(j) = h$ for an activity $j \in \{i+1, \ldots, n\}$, set $\tilde{H}(j) := h^*$.

$\tilde{H}$ is a total assignment that extends $H$ together with the greedy assignment of activity $i+1$, and it uses no more halls than $H^*$, so it is optimal.

Theorem

The greedy procedure finds an optimal assignment.

Proof

By the lemma using induction on the number of activities.
Exercise cont'd

Remark

* Note that the approach that repeatedly finds a
  maximum-cardinality non-intersecting subset of activities,
  assigns them to a hall, removes them from the input,
  and iterates, is not correct, as shown by the
  following counterexample:

```
        1
          2
            3
              4
```

The above approach partitions the activities into 3 halls:

- $\{1,3\}$, $\{2\}$, $\{4\}$.

But an optimal solution partitions them into 2 halls:

- $\{1,4\}$, $\{2,3\}$. 