(a) Consider the problem instance:

\[
\begin{array}{c}
\text{a} \rightarrow 1 \leftarrow \text{b} \\
\rightarrow 2 \leftarrow \text{c} \\
\leftarrow 3 \rightarrow \text{d} \\
\leftarrow 3 \rightarrow 1
\end{array}
\]

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

(b) Consider the problem instance:

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

The overlap counts are:
\[
\begin{array}{cccccccc}
\text{a} & \text{b} & \text{c} & \text{d} & \text{e} & \text{f} & \text{g} & \text{h} & \text{i} & \text{j} \\
3 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 & 3
\end{array}
\]

The optimal solution is \( \{ a, e, g, k \} \).
Being greedy by fewest remaining overlaps chooses \( \{ f, a, k \} \), which is suboptimal.
Problem cont'd

(b) Consider the problem instance:

\[
\begin{array}{c}
\text{a} \\
\text{b} \\
\text{c}
\end{array}
\]

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 iff successive gas stations are always at most n miles apart, so we assume this holds.

![Diagram](A \rightarrow g_1, g_2, g_3 \ldots \rightarrow g_m \rightarrow B)

- 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 $\Theta(m)$ time if the stations are given in order.
Exercise cont!

Correctness

Lemma

For any 1 ≤ i ≤ m, let \( G \) be the subset of stations \( g_1, \ldots, g_i \) chosen by the greedy procedure, and let \( g_k \), where \( k \neq i \), be the next greedy step. Suppose there is an optimal solution that agrees with \( G \) on \( g_1, \ldots, g_i \).

Then there is an optimal solution that agrees with \( G \cup \{g_k\} \) on \( g_1, \ldots, g_k \).

Proof Let \( G^* \supseteq G \) be an optimal solution that agrees with \( G \) on \( g_1, \ldots, g_i \).

If the first stop in \( G^* \) after \( g_i \) is at \( g_k \), the lemma holds. So suppose \( G^* \) does not stop at \( g_k \).

Then it must stop at least once before \( g_k \). (If its first stop after \( g_i \) occurs after \( g_k \), the greedy procedure could have gone farther than \( g_k \) without stopping — a contradiction.)

Let \( G' \) be \( G^* \) with its stops between \( g_i \) and \( g_k \) replaced by \( g_k \). \( G' \) is a feasible solution that agrees with \( G \cup \{g_k\} \) on \( g_1, \ldots, g_k \); 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**

1. Sort the tasks by increasing running time.
2. Rename the sorted tasks so that $t_1 \leq t_2 \leq \ldots \leq t_n$.
3. We then execute the tasks in this order 1, 2, ..., n, starting them at times 0, $t_1$, $t_1 + t_2$, ..., $\sum_{1 \leq i \leq n} t_i$.

   (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 S by exchanging the positions of tasks i+1 and j, as follows.

\[
S^* = \begin{array}{cccccc}
1 & 2 & \cdots & i & i+1 & k & n \\
\end{array}
\]

\[
S = \begin{array}{cccccc}
1 & 2 & \cdots & i & i+1 & j & \cdots \\
\end{array}
\]

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

\[ c(\bar{S}) = \frac{1}{n} \sum_{i \in \mathcal{S}} \sum_{j \in i} t_{ij} \]

\[ = \frac{1}{n} \sum_{i \in \mathcal{S}} (n - i + 1) t_{S[i]} \]

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

\[ c(S^*) - c(\bar{S}) = \frac{1}{n} \left( \left( (n-i) t_j + (n-k+1) t_{i+n} \right) - \left( (n-i) t_{i+1} + (n-k+1) t_j \right) \right) \]

\[ = \frac{1}{n} \left( (k - (i+1)) t_j - (k - (i+1)) t_{i+1} \right) \]

\[ = \frac{1}{n} \left( k - (i+1) \right) \left( t_j - t_{i+1} \right) \]

\[ \geq 0, \quad > 0 \quad > 0 \quad \text{since} \quad t_1 \leq \ldots \leq t_n \]

which implies \( c(\bar{S}) \leq c(S^*) \).

Thus \( \bar{S} \) is an optimal solution that contains the partial

solution 1, 2, ... , \( i + 1 \).

\[ \square \]

Theorem: The greedy procedure finds an optimal schedule.

Proof: By the lemma, using induction on the number of iterations. 

\[ \square \]