This is a practice problem (with solution appended) for Homework 3. It is similar to the actual homework problem, but this problem is only for practice. Do not submit your solutions to it.

Problem 1. The INTERVAL COVERING problem is defined as follows:

input: A pair $I = (S, [a, b])$ where $S = \{S_1, S_2, \ldots, S_n\}$ is a set of open intervals and $[a, b]$ is a closed interval such that $[a, b] \subseteq \bigcup_{S_i \in S} S_i$.

output: A minimum-size subset C of S such that $[a, b] \subseteq \bigcup_{S_i \in C} S_i$.

Each S_i is an *open interval*, denoted (s_i, f_i), so $S_i = \{x \in \mathbb{R} : s_i < x < f_i\}$ contains the points that lie strictly between the start s_i and the finish f_i. For this problem we say that interval S_i *covers* each point $x \in S_i$. A set C is a *cover* for $(S, [a, b])$ if $C \subseteq S$ vand each point in $[a, b]$ is covered by some interval in C. Cover $C \subseteq S$ is *optimal* for $[a, b]$ if it has minimum possible size (among all covers for $(S, [a, b])$). (The size $|C|$ is the number of intervals in C.) So, the problem is to find an optimal cover for $(S, [a, b])$. (Note that C does not have to be pairwise disjoint.)

You are to design (and prove correct) a fast greedy algorithm for INTERVAL COVERING. Given I, the algorithm should choose intervals from S one by one, until the chosen intervals form a cover C for $(S, [a, b])$. It should return the cover C, which should be optimal.

(a) To warm up your intuition, consider the following rule for choosing the first interval: *pick an interval S_i of maximum length $f_i - s_i$.* Describe an instance $I = (S = \{S_1, S_2, S_3\}, [0, 1])$ containing just three open intervals for which this rule picks an interval that is not in any optimal cover.

(b) To get started, precisely describe a simple rule that, given any instance $I = (S, [a, b])$ with $a \le b$, picks an interval $S_i \in S$ that *must* be in some optimal cover. (It must be clear that your rule can easily be computed in polynomial time. E.g., the rule "Choose any S_i that's in an optimal cover." is cheating!)

(c) Prove your rule works by completing the proof of the following lemma:

Lemma 1. *Let $I = (S, [a, b])$ be any instance with $a \le b$. Let S_i be an interval in S chosen as described in part (b). Then S_i is in some optimal cover.*

Proof (long form).

1. Consider any instance $I = (S, [a, b])$ and interval $S_i \in S$ as described in the lemma.
2. Let $C = \{C_1, C_2, \ldots, C_k\}$ be an optimal cover for I.
3. Let $C_j \in C$ be an interval in C that $\star \cdots \star$
   (C_j must exist because $\star \cdots \star$)
4. Let $C' = \{S_i\} \cup C \setminus \{C_j\}$ be obtained from C by replacing C_j by S_i.
   (If $S_i = C_j$, then $C' = C$.)
5. $C' = \{S_i\} \cup C \setminus \{C_j\}$ is also a cover of $[a, b]$ because $\star \cdots \star$
6. So S_i is in some optimal cover.

□

(d) Complete the proof of the optimal-substructure lemma for this rule:

Lemma 2 (optimal substructure)**.** *Let $I = (S, [a, b])$ be any instance with $a \le b$. Let S_i be an interval in S chosen as described in part (b). Define instance $I' = (S, [a', b'])$ such that*

$\star \cdots \star$

Let R be any optimal cover for instance I'. Then $\{S_i\} \cup R$ is an optimal cover for I.

Proof (long form).

1. Suppose for contradiction that $\{S_i\} \cup R$ is not optimal for I.
2. $\{S_i\} \cup R$ is a cover for I because
   $\star \cdots \star$

3. Since $\{S_i\} \cup R$ is not optimal for I, it must be that there exists a smaller cover for I.
4. Let C^* be a smaller cover for I, with S_i in C^*. (C^* exists by Lemma 1.)
5. Let $R' = C^* \setminus \{S_i\}$.
6. Then R' is a cover for I' because
   ⋆ · · · ⋆
7. But $C^* = \{S_i\} \cup R'$ is smaller than $\{S_i\} \cup R$ (Step 4), so R' is smaller than R.
8. This contradicts that R is optimal for I'.

□

(e) Extend your rule to obtain a greedy algorithm for INTERVAL COVERING. Give a recursive algorithm. Describe the algorithm precisely in words, then give pseudo-code for it. As closely as possible, follow the style of the corresponding (iterative or recursive) algorithm for ACTIVITY SELECTION in the lecture notes. Here's a candidate:

---

rcover($I = (S, [a, b])$): — *recursive* INTERVAL COVERING *algorithm* —
1. if $a > b$: return $\emptyset$.
2. Let $S_i \in S$ and I' be chosen for I as described in Lemma 1.
3. Recursively compute $R = \text{rcover}(I')$, then return $\{S_i\} \cup R$.

(f) Complete the following proof of correctness of the recursive algorithm.

Warning: be very careful to make sure your inductive argument is sound! Is the instance I' actually smaller than the instance I'?

Theorem 1. *The recursive algorithm is correct.*

Proof (long form).

1. Say that the algorithm is *correct for* $I = (S, [a, b])$ if it returns an optimal cover for I.
2. ⋆ · · · ⋆
3. Inductively, change is correct for all instances.

□

(g) Now give an iterative version of your algorithm. Here's a template:

---

icover($S, [a, b]$): — *iterative* INTERVAL COVERING *algorithm* —
1. $X \leftarrow ()$ *(The empty sequence.)*
2. while $a \leq b$ do:

3.1. Choose an interval $S_i \in S$ such that
⋆ · · · ⋆

3.2. append S_i to X *(Add next piece to partial solution.)*

3.3. $[a, b] \leftarrow$ ⋆ · · · ⋆

4. return X

(h) Complete the proof of the following lemma:

Lemma 3 (generalizes Lemma 1)**.** *The iterative algorithm maintains the following invariant:*

greedy invariant*: The partial solution X is a subset of some optimal cover.*

Proof (long form).

1. Consider the execution of the algorithm on any instance $I = (S, [a, b])$.
2. ⋆ · · · ⋆
3. Next we show that the algorithm maintains the invariant.

4. The invariant holds at the start of the first iteration, when $X = \emptyset$.
5.1. Consider any iteration t such that the invariant holds at the start of the iteration.
5.2. Let $X = \{X_1, \ldots, X_{t-1}\}$ be the partial solution at the start of the iteration.
5.3. $\star \cdots \star$
5.4. By the invariant, for some optimal solution, say C, we have $X \subseteq C$.
5.5. Let S_j be an interval in C that $\star \cdots \star$
(It exists, because $\star \cdots \star$.)
5.6. Let $C' = \{X_t\} \cup C \setminus \{S_j\}$ be the cover obtained from C by replacing X_t by S_j.
(If $X_t = S_j$, then $C' = C$.)
5.7. $\star \cdots \star$
5.8. So $C' = \{X_t\} \cup C \setminus \{S_j\}$ covers all the points that C does,
5.9. And $|C'| = |C|$, so C' is also an optimal cover for I.
5.10. Since $\{X_1, \ldots, X_{t-1}, X_t\} \subseteq C'$, the invariant holds at the end of iteration t
6. By Block 5, each iteration that starts with the invariant true ends with it true.
7. And the invariant holds initially. So it holds throughout.

$\square$

(i) Complete the following proof:

Theorem 2. *The iterative algorithm is correct.*

Proof (long form).

1. Consider the execution of the algorithm on any instance $I = (S, [a, b])$.
2. Consider the state after the final iteration. Let X be the set at that time.
3. By Lemma 8, the invariant holds then, so $X \subseteq C$ for some optimal cover C.
4. X is a cover for I because $\star \cdots \star$
5. Since C is optimal, $|C| \leq |X|$.
6. So $X = C$, and the set X returned by the algorithm is an optimal cover for I.

$\square$

solutions on next page

Solutions for practice problem. (INTERVAL COVERING)

(a) The instance is $I = (S, [0, 1])$ with $S = \{[0, 1/2], [1/2, 1], [1/10, 9/10]\}$.
The rule chooses $[1/10, 9/10]$ but the only optimal cover is $\{[0, 1/2], [1/2, 1]\}$.

(b) Here's a better rule. Given $I = (S, [a, b])$ with $a \geq b$:
choose an $S_i \in I$ that contains a, and, among intervals containing a, ends latest.

(c) Here's a proof that the rule chooses an interval in some optimal cover:

Lemma 4. *Let $I = (S, [a, b])$ be any instance with $a \leq b$. Let S_i be an interval in S chosen as described in part (b). Then S_i is in some optimal cover.*

Proof (long form).

1. Consider any instance $I = (S, [a, b])$ and interval $S_i \in S$ as described in the lemma.
2. Let $C = \{C_1, C_2, \ldots, C_k\}$ be an optimal cover for I.
3. Let $C_j \in C$ be an interval in C that contains a. (C_j must exist because C covers $[a, b]$.)
4. Let $C' = \{S_i\} \cup C \setminus \{C_j\}$ be obtained from C by replacing C_j by S_i.
(If $S_i = C_j$, then $C' = C$.)
5. $C' = \{S_i\} \cup C \setminus \{C_j\}$ is also a cover of $[a, b]$ because
S_i also contains a and ends no earlier than C_j,
so each point $x \in [a, b]$ covered by C_j is also covered by S_i.
6. So S_i is in some optimal cover.

□

(d) Here's the optimal-substructure lemma for this rule.

Lemma 5 (optimal substructure)**.** *Let $I = (S, [a, b])$ be any instance with $a \leq b$. Let S_i be an interval in S chosen as described in part (b). Define instance $I' = (S, [a', b'])$ such that*

$a' = f_i$ is the ending time of S_i, and $b' = b$.

Let R be any optimal cover for instance I'. Then $\{S_i\} \cup R$ is an optimal cover for I.

Proof (long form).

1. Suppose for contradiction that $\{S_i\} \cup R$ is not optimal for I.
2. $\{S_i\} \cup R$ is a cover for I because
S_i covers all points in $[a, f_i)$, and the intervals in R cover all points in $[a', b'] = [f_i, b]$.
3. Since $\{S_i\} \cup R$ is not optimal for I, it must be that there exists a smaller cover for I.
4. Let C^* be a smaller cover for I, with S_i in C^*. (C^* exists by Lemma 4.)
5. Let $R' = C^* \setminus \{S_i\}$.
6. Then R' is a cover for I' because
S_i only covers points in $[a, f_i)$, so R' must cover $[f_i, b]$.
7. But $C^* = \{S_i\} \cup R'$ is smaller than $\{S_i\} \cup R$ (Step 4), so R' is smaller than R.
8. This contradicts that R is optimal for I'.

□

(e) Here is the recursive algorithm.

---

rcover($I = (S, [a, b])$): — *recursive* INTERVAL COVERING *algorithm* —

1. if $a > b$: return $\emptyset$.
2. Let $S_i \in S$ and I' be chosen for I as described in Lemma 4.
3. Recursively compute $R = \text{rcover}(I')$, then return $\{S_i\} \cup R$.

(f) Here's a proof of correctness of the recursive algorithm.

Theorem 3. *The recursive algorithm is correct.*

Proof (long form).

1. Say that the algorithm is *correct for* $I = (S, [a, b])$ if it returns an optimal cover for I.
2. Recall that any valid instance $I = (S, [a, b])$ must have $[a, b] \subseteq \cup_{S_i \in S} S_i$.
3. Define $\phi(I = (S, [a, b]))$ to be the number of intervals in S that contain any point in $[a, b]$.
4. We prove by induction on $\phi(I)$ that change(I) is correct for all valid instances I.
5. For any $I = (S, [a, b])$ with $\phi(I) = 0$, it must be that $a > b$, so the solution $\emptyset$ is correct.

6.1. Consider any valid instance $I = (S, [a, b])$ with $\phi(I) > 0$. So $a \leq b$.
6.2.1. Assume that the algorithm is correct for all valid instances I' with $\phi(I') < \phi(I)$.
6.2.2. Consider the execution of the algorithm on I.
6.2.3. Let S_i and $I' = (S, [a', b'])$ be as computed in Line 2 of the algorithm for this iteration.
6.2.4. Since I is a valid instance, so is I' (using here that $[a', b'] \subseteq [a, b]$).
6.2.5. And $\phi(I') < \phi(I)$ because $S_i \in I \setminus I'$ does not cover any point in $[a', b'] = [f_i, b]$.
6.2.6. So, by the inductive assumption, R is optimal for the instance I'.
6.2.7. So, by Lemma 5, the solution $\{S_i\} \cup R$ returned by change(I) is optimal for I.
6.3. By Block 6.2, if change is correct for all I' with $\phi(I') < \phi(I)$, then it is correct for I.

7. By Block 6, for all I, if change is correct for all I' with $\phi(I') < \phi(I)$, it is correct for I.
8. By Step 5, change is correct for all I with $\phi(I) = 0$.
9. Inductively, change is correct for all instances.

□

(g) Here's the iterative algorithm:

---

icover$(S, [a, b])$: — *iterative* Interval Covering *algorithm* —
1. $X \leftarrow ()$ *(The empty sequence.)*
2. while $a \leq b$ do:
3.1. Choose a in interval $S_i \in S$ such that
S_i contains a, and among such intervals ends latest.
3.2. append S_i to X *(Add next piece to partial solution.)*
3.3. $a \leftarrow$ the end time f_i of S_i
4. return X

(h) Here's a proof that the iterative algorithm for Interval Covering maintains the invariant.

Lemma 6 (generalizes Lemma 4)**.** *The iterative algorithm maintains the following invariant:*

greedy invariant*: The partial solution X is a subset of some optimal cover.*

Proof (long form).

1. Consider the execution of the algorithm on any instance $I = (S, [a, b])$.
2. Let $(X_1, X_2, \ldots, X_T)$ be the intervals added to X, in the order added (one per iteration).
3. Let $a_0 = a$. For $t \in \{1, 2, \ldots, T\}$, let a_t be the endpoint of X_t.
4. For each t, the intervals $(X_1, \ldots, X_t)$ cover $[a_0, a_t)$, because (by inspection of the algorithm) each interval X_s contains the endpoint a_{s-1} of the previous interval X_{s-1} (or a_0 if $s = 1$).
5. This and the termination condition ($a_T > b$) imply that at termination X covers $[a, b]$.

6. Next we show that the algorithm maintains the invariant.
7. The invariant holds at the start of the first iteration, when $X = \emptyset$.
8.1. Consider any iteration t such that the invariant holds at the start of the iteration.

8.2. Then $X = \{X_1, \ldots, X_{t-1}\}$ is the partial solution at the start of the iteration.
8.3. The value of the variable a at the start of the iteration is a_{t-1}.
8.4. So X_t contains a_{t-1} and, among intervals in I that do, ends latest.
8.5. By the invariant, for some optimal solution, say C, we have $X \subseteq C$.
8.6. Let S_j be an interval in C that contains a_{t-1}.
(It exists, as C covers $[a, b]$ and $a \leq a_{t-1} \leq b$.)
8.7. Let $C' = \{X_t\} \cup C \setminus \{S_j\}$ be the cover obtained from C by replacing X_t by S_j.
(If $X_t = S_j$, then $C' = C$.)
8.8. By Step 4, X covers $[a_0, a_{t-1})$. And X_t contains a_{t-1} and ends no later than S_j.
8.9. So each point x covered by S_j is either covered by X (if $x < a_{t-1}$) or by X_t (otherwise).
8.10. So $C' = \{X_t\} \cup C \setminus \{S_j\}$ covers all the points that C does,
8.11. And $|C'| = |C|$, so C' is also an optimal cover for I.
8.12. Since $\{X_1, \ldots, X_{t-1}, X_t\} \subseteq C'$, the invariant holds at the end of iteration t
9. By Block 8, each iteration that starts with the invariant true ends with it true.
10. And the invariant holds initially. So it holds throughout. □

(i) Here's a proof that the iterative algorithm is correct.

Theorem 4. *The iterative algorithm is correct.*

Proof (long form).

1. Consider the execution of the algorithm on any instance $I = (S, [a, b])$.
2. Consider the state after the final iteration. Let X be the set at that time.
3. By Lemma 8, the invariant holds then, so $X \subseteq C$ for some optimal cover C.
4. X is a cover for I by Step 4 of the previous proof.
5. Since C is optimal, $|C| \leq |X|$.
6. So $X = C$, and the set X returned by the algorithm is an optimal cover for I. □