

General instructions. Answer the problems given in the rest of this document. Edit this LaTeX template to add your answers, as described in the file named `00_README.txt`, then compile it to produce `homework_5.pdf`.

Don't know? Say so. If you don't know the answer to a problem or any part of the problem, we encourage you to simply write “*I don't know*” instead of giving an answer. (If you write “I don't know” you can also add any questions you might have.)

Gentle reminder: homework is for learning. As noted in [Lecture Note 1](#), engaging fully with the homeworks is the best way to learn the material. *The primary role of homeworks is to help you learn the material, and the primary role of feedback on homeworks is to help you improve your ideas.*

Please try to explain your ideas as clearly as you can. The clearer your explanations are, the more useful the feedback on them can be.

Collaboration and citation policy. If you are doing this homework as part of a course, follow the course's collaboration and citation policy. In any case, at the end of each answer, list the people you collaborated with, and cite every source that your answer uses ideas from (other than the lecture notes and the sources listed in the lecture notes). If there are none, write “none”.

Problem 1. (*Greedy algorithm for Puncturing Intervals; see Exercise 5.14*)

Recall the Puncturing Intervals problem from Homework 4:

input: Set $I = \{J_1, J_2, \dots, J_n\}$ of intervals, where $J_i = [s_i, e_i]$. We assume $e_1 \leq e_2 \leq \dots \leq e_n$.

output: A minimum-size finite set $P = \{p_1, p_2, \dots, p_k\}$ of points such that $J_i \cap P \neq \emptyset$ for each $J_i \in I$.

Consider the following rule for picking a single point p_1 that is guaranteed to be in some correct solution:

1. Let p_1 be the end-time e_1 of the earliest-ending interval J_1 .

Consider the following two lemmas:

Lemma 1. *Let I be any non-empty instance of Puncturing Intervals. Let e_1 be the end-time of the earliest-ending interval J_1 . Then I has a correct solution that contains e_1 .*

Lemma 2. *Let I be any non-empty instance of Puncturing Intervals. Let e_1 be the end-time of the earliest-ending interval J_1 . Let D contain the intervals in I that don't contain e_1 . Let R be any optimal solution for D (as a Puncturing Intervals instance). Then $\{e_1\} \cup R$ must be a correct solution for I .*

- (a) Give a long-form proof of Lemma 1. As much as possible, your proof should mimic the structure of the proof of Lemma 5.4 (for Activity Selection) in Lecture Note 5.
- (b) Give a long-form proof of Lemma 2. As much as possible, your proof should mimic the structure of the proof of Lemma 5.5 (for Activity Selection) in Lecture Note 5.
- (c) Following the lemmas, define a correct recursive algorithm `rpuncture` for Puncturing Intervals. As much as possible, your algorithm should mimic the structure of the `rselect` algorithm for Activity Selection in Section 5.3. Define the algorithm precisely in words, then give pseudo-code for it.
- (d) Prove Theorem 1, below, for your algorithm. As much as possible, your proof should mimic the structure of the proof of Theorem 5.2 (for `rselect`) in Lecture Note 5. But you must give your proof in long form, following the template given here.

Theorem 1. *For any instance $I = \{J_1, J_2, \dots, J_n\}$ of Puncturing Intervals, `rpuncture`(I) returns a correct solution for I .*

Note: for each Part (a)-(d), the template provides the corresponding proof or algorithm for Activity Selection as a starting point. You can edit that proof or algorithm to create your own answer.

(continued)

Problem 1(a) answer

Lemma 1. *Let I be any non-empty instance of Puncturing Intervals. Let e_1 be the end-time of the earliest-ending job J_1 . Then I has a correct solution that contains e_1 .*

Proof (long form).

1. Consider any I and J_1 as described in the lemma, and let S^* be any correct solution for I .
- 2.1. Case 1. Suppose that J_1 is in S^* .
- 2.2. Then I has a correct solution that contains J_1 (namely, S^*).
- 3.1. Case 2. In the remaining case, J_1 is not in S^* .
- 3.2. Let J_1^* be the first (earliest ending) job in S^* . Note $J_1^* \neq J_1$.
- 3.3. We'll show that exchanging J_1 for J_1^* in S^* gives a correct solution (containing J_1).
- 3.4. Let $S' = \{J_1\} \cup S^* \setminus \{J_1^*\}$ be obtained from S^* by replacing J_1^* by J_1 .
- 3.5. Job J_1 ends no later than J_1^* ends, and J_1^* ends before any other job in S^* starts.
- 3.6. So J_1 ends before any job (other than J_1^*) in S^* starts.
- 3.7. So S' is also pairwise disjoint. And S' has the same size as S^* .
- 3.8. So S' is also a correct solution. So I has a correct solution (namely S') that contains J_1 .
4. So I has a correct solution that contains J_1 . □

(continued)

Problem 1(b) answer

Lemma 2. *Let I be any non-empty instance of Puncturing Intervals. Let e_1 be the end-time of the earliest-ending job J_1 . Let D contain the jobs in I that don't contain e_1 . Let R be any optimal solution for D (as a Puncturing Intervals instance). Then $\{e_1\} \cup R$ must be a correct solution for I .*

Proof (long form).

1. Consider any I , J_1 , D , and R as described in the lemma.
2. Let S^* be a correct solution for I that contains J_1 (it exists by Lemma 1).
3. Let $R' = S^* \setminus \{J_1\}$ be obtained by removing J_1 from S^* .
4. Then R' is a pairwise-disjoint subset of D (using here that S^* is pairwise disjoint).
5. Since R is a maximum-size pairwise-disjoint subset of D , we have $|R| \geq |R'|$.
6. So $|S^*| = 1 + |R'| \leq 1 + |R| = |\{J_1\} \cup R|$.
7. That is, $\{J_1\} \cup R$ is as large as S^* .
8. And $\{J_1\} \cup R$ is pairwise disjoint (because R is, and $R \subseteq D$ so all jobs in R are disjoint from J_1).
9. By the previous two steps, $\{J_1\} \cup R$ must also be a correct solution for I . □

(continued)

Problem 1(c) answer The algorithm chooses the earliest-ending job J_1 , then recurses on the set containing those jobs that are disjoint from J_1 . Here is pseudo-code:

$\text{rselect}(I = \{J_1, J_2, \dots, J_n\})$: — jobs are ordered by end time
 1. if $n = 0$: return $\emptyset$. — if I is empty, return the empty set
 2. Recall that J_1 is the earliest-ending job in I .
 3. Let D contain the jobs in I that are disjoint from J_1 .
 4. Recursively compute $R = \text{rselect}(D)$, then return $\{J_1\} \cup R$.

Problem 1(d) answer

Theorem 1. For any instance $I = \{J_1, J_2, \dots, J_n\}$ of Puncturing Intervals, $\text{rpuncture}(I)$ returns a correct solution for I .

Proof (long form).

1. The proof is by induction on n .
2. For the base case $n = 0$, rselect returns the empty set, so is correct in this case.
- 3.1. Consider executing $\text{rselect}(I)$ for any I with $n \geq 1$. Assume rselect is correct on smaller instances.
- 3.2. Let J_1 , D , and $R = \text{rselect}(D)$ be as computed by Lines 2–4 of the algorithm.
- 3.3. Note that $|D| < |I|$, so by the assumption in Step 3.1 R is a correct solution for D .
- 3.4. So, by Lemma 2, the solution $\{J_1\} \cup R$ returned by $\text{rselect}(I)$ is correct for I .
4. By Block 3, for any instance I with $n \geq 1$, if rselect is correct on smaller instances, then $\text{rselect}(I)$ is correct.
5. By this, Step 2, and induction, rselect is correct for all instances. □

Problem 1 collaboration and citations

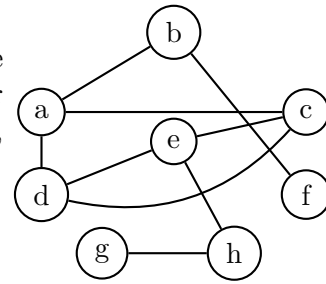
I collaborated on this problem with the following people:

● ★ ⋯ ★

My answers use ideas from the following sources (other than the lecture notes and textbooks):

● ★ ⋯ ★

Problem 2. (*Exercise 6.3; BFS*) Simulate BFS on the graph to the right, starting from the node a . When there are multiple choices for the next node to visit, break ties alphabetically (choose b before c , etc.).



- List the nodes in each level ℓ (at distance ℓ from a).
- For each node other than a , give its parent in the BFS tree.
- Optionally, draw the BFS tree. If you do this, and your answers for Parts (a) and (b) are substantially incorrect, you may get partial credit here.

Problem 2 answer

- Nodes in level 0: a
 Nodes in level 1: $\star \cdots \star$
 Nodes in level 2: $\star \cdots \star$
 Nodes in level 3: $\star \cdots \star$
 Nodes in level 4: $\star \cdots \star$

- Parent of b : $\star \cdots \star$
 Parent of c : $\star \cdots \star$
 Parent of d : $\star \cdots \star$
 Parent of e : $\star \cdots \star$
 Parent of f : $\star \cdots \star$
 Parent of g : $\star \cdots \star$
 Parent of h : $\star \cdots \star$

Problem 2(c) answer (OPTIONAL!) ¹**Problem 2 collaboration and citations**

I collaborated on this problem with the following people:

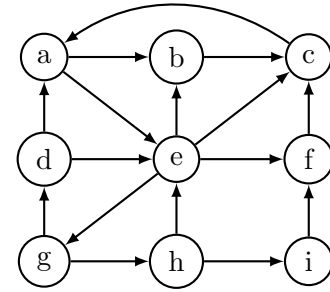
● ★ ⋯ ★

My answers use ideas from the following sources (other than the lecture notes and textbooks):

● ★ ⋯ ★

¹SEE FURTHER INSTRUCTIONS IN THE COMMENTS IN PROBLEM_2_ANSWER.TEX!

Problem 3. (*Exercise 6.4; DFS*) Simulate DFS (as described in the lecture notes) on the graph to the right. When there are multiple choices for the next node to visit, break ties alphabetically (choose *a* before *b*, choose *b* before *c*, etc.).



- What is the parent of each node in the resulting DFS forest? (Write “none” for each root.)
- What’s the post-order number (finishing time, Section 6.3.4), of each node?
- Which edges are forward edges? Back edges? Cross edges?
- Optionally, draw the DFS tree, and label each node with its post-order number. If you do this, and your answers for Parts (a)–(c) are substantially incorrect, you may get partial credit here.

Problem 3 answer

- Parent of *a*: ★...★
 Parent of *b*: ★...★
 Parent of *c*: ★...★
 Parent of *d*: ★...★
 Parent of *e*: ★...★
 Parent of *f*: ★...★
 Parent of *g*: ★...★
 Parent of *h*: ★...★
 Parent of *i*: ★...★
- Post-order number of *a*: ★...★
 Post-order number of *b*: ★...★
 Post-order number of *c*: ★...★
 Post-order number of *d*: ★...★
 Post-order number of *e*: ★...★
 Post-order number of *f*: ★...★
 Post-order number of *g*: ★...★
 Post-order number of *h*: ★...★
 Post-order number of *i*: ★...★
- Forward edges: ★...★
 Back edges: ★...★
 Cross edges: ★...★

Problem 3(d) answer (OPTIONAL!) ²**Problem 3 collaboration and citations**

I collaborated on this problem with the following people:

• ★ ⋯ ★

My answers use ideas from the following sources (other than the lecture notes and textbooks):

• ★ ⋯ ★

²SEE FURTHER INSTRUCTIONS IN THE COMMENTS IN PROBLEM_3_ANSWER.TEX!