Problem 1. Recall the Puncturing Intervals problem:

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

output: A minimum-size finite set $P = \{p_1, p_2, \ldots, p_k\}$ of points such that $J_i \cap P \ne \emptyset$ 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.

(b) Give a long-form proof of Lemma 2.

(c) Following the lemmas, define a correct recursive algorithm rpuncture for Puncturing Intervals. Define the algorithm precisely in words, then give pseudo-code for it.

(d) Prove Theorem 1, below, for your algorithm. But you must give your proof in long form, following the template given here.

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

(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, J_1, e_1 as described in the algorithm, and let P^* be any correct solution for I.

2.1. <u>Case 1.</u> Suppose that e_1 is in P^*.

2.2. Then I has a correct solution that contains e_1 (namely, P^*).

3.1. <u>Case 2.</u> In the remaining case, e_1 is not in P^*.

3.2. Let p_1^* be the point such that $J_1 \cap p_1^* = \emptyset$ in P^*. Because P* is a correct solution, it must contain this point that intersects with J_1, the earliest ending interval. Note $p_1^* \neq e_1$.

3.3. We'll show that exchanging p_1^* for e_1 in P^* gives a correct solution (containing e_1).

3.4. Let $P' = \{e_1\} \cup P^* \setminus \{p_1^*\}$ be obtained from P^* by replacing p_1^* by e_1.

3.5. Point e_1 must be later than p_1^*, because both $J_1 \cap p_1^* = J_1 \cap e_1 = \emptyset$, and e_1 (endpoint) $\neq p_1^*$.

3.6. Since J_1 ends before any other job, if p_1^* was also satisfying $J_i \cap p_1^* = \emptyset$ for some other J_i(s) $\neq J_1$ in I (so the J_is overlapped with J_1), then because $s_i \leq e_1 \leq e_i$, then e_1 would also satisfy $J_i \cap e_1 = \emptyset$.

3.7. So P' satisfies the condition for J_1 and any overlapping jobs with J_1, J_is. And P' has the same size as P^*.

3.8. So P' is also a correct solution. So I has a correct solution (namely P') that contains e_1.

4. So I has a correct solution that contains e_1. $\square$

(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, e_1, J_1, D, and R as described in the lemma.
2. Let P^* be a correct solution for I that contains e_1 (it exists by Lemma 1).
3. Let $R' = P^* \setminus \{e_1\}$ be obtained by removing e_1 from P^*.
4. Then R' is a solution for puncturing intervals of D (using here that D contains the jobs in I that don't contain e_1).
5. Since R is a minimum-size optimal solution for puncturing intervals of D, we have $|R| \leq |R'|$.
6. So $|\{e_1\} \cup R| = 1 + |R| \leq 1 + |R'| = |P*|$.
7. That is, the size of $\{e_1\} \cup R$ is at most the size of P^*.
8. $\{e_1\} \cup R$ satisfies the condition $J_i \cup p_i \neq \emptyset$ for intervals that contain e_1 and intervals that do not.
9. By the previous two steps, $\{e_1\} \cup R$ must also be a correct solution for I. □

(continued)

Problem 1(c) answer The algorithm chooses the endpoint of the earliest-ending job J_1, e_1, then recurses on the set containing those jobs that don't contain e_1. Here is pseudo-code:

rpuncture($I = \{J_1, J_2, \ldots, J_n\}$): — *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 at e_1.
3. Let D contain the jobs in I that don't contain e_1 ($D = \{J_i \in I : J_i \cap e_1 = \emptyset\}$).
4. Recursively compute $R = \text{rpuncture}(D)$, then return $\{e_1\} \cup R$.

Problem 1(d) answer

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

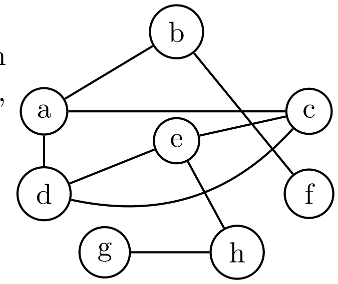
Proof (long form).

1. The proof is by induction on n.
2. For the base case $n = 0$, rpuncture returns the empty set, so is correct in this case.

3.1. Consider executing rpunture(I) for any I with $n \geq 1$. Assume rpuncture is correct on smaller instances.
3.2. Let e_1, D, and $R = \text{rpuncture}(D)$ be as computed by Lines 2 through 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 $\{e_1\} \cup R$ returned by rpuncture(I) is correct for I.

4. By Block 3, for any instance I with $n \geq 1$, if rpuncture is correct on smaller instances, then rselect(I) is correct.
5. By this, Step 2, and induction, rpuncture is correct for all instances. □

Problem 2. 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.).



(a) List the nodes in each level ℓ (at distance ℓ from a).p

(b) For each node other than a, give its parent in the BFS tree.

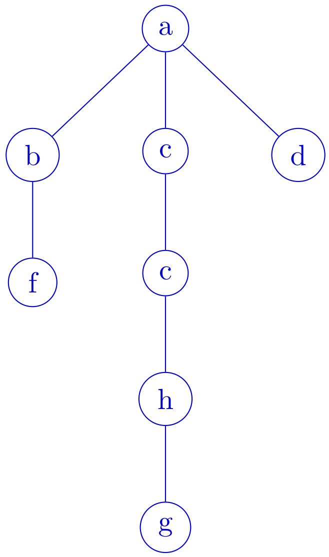
(c) Optionally, draw the BFS tree. If you do this, and your answers for Parts (a) and (b) are substantially incorrect, you may get partial credit here.

Problem 2 answer

(a) Nodes in level 0: a
Nodes in level 1: b, c, d
Nodes in level 2: e, f
Nodes in level 3: h
Nodes in level 4: g

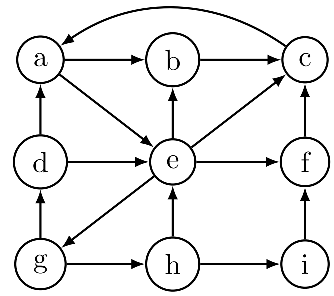
(b) Parent of b: a
Parent of c: a
Parent of d: a
Parent of e: c
Parent of f: b
Parent of g: h
Parent of h: e

Problem 2(c) answer (OPTIONAL!) [1]



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

Problem 3. Simulate DFS 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.).



(a) What is the parent of each node in the resulting DFS forest? (Write "none" for each root.)

(b) What is the post-order number of each vertex?

(c) Which edges are forward edges? Back edges? Cross edges?

(d) 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

(a) Parent of a: *none*
Parent of b: a
Parent of c: b
Parent of d: g
Parent of e: a
Parent of f: e
Parent of g: e
Parent of h: g
Parent of i: h

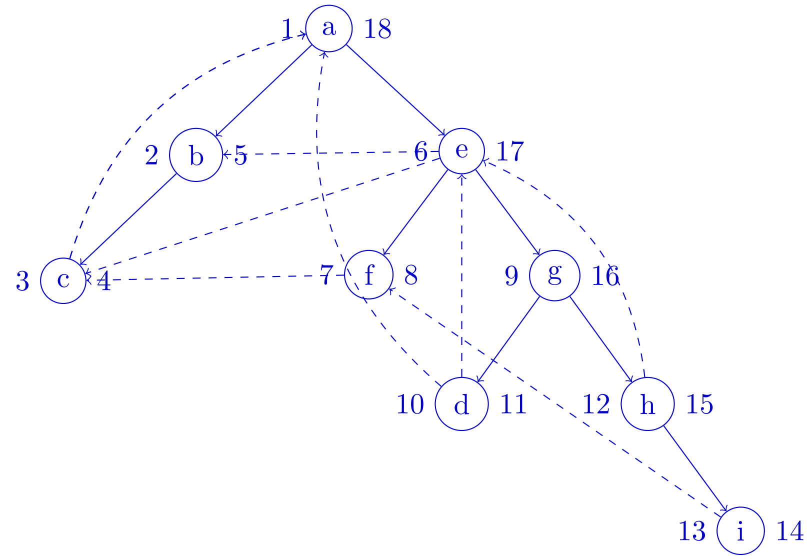
(b) Post-order number of a: 18
Post-order number of b: 5
Post-order number of c: 4
Post-order number of d: 11
Post-order number of e: 17
Post-order number of f: 8
Post-order number of g: 16
Post-order number of h: 15
Post-order number of i: 14

(c) Forward edges: *none*

Back edges: $(c, a), (d, a), (d, e), (h, e)$

Cross edges: $(e, b), (e, c), (f, c), (i, f)$

Problem 3(d) answer (OPTIONAL!) [2]



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