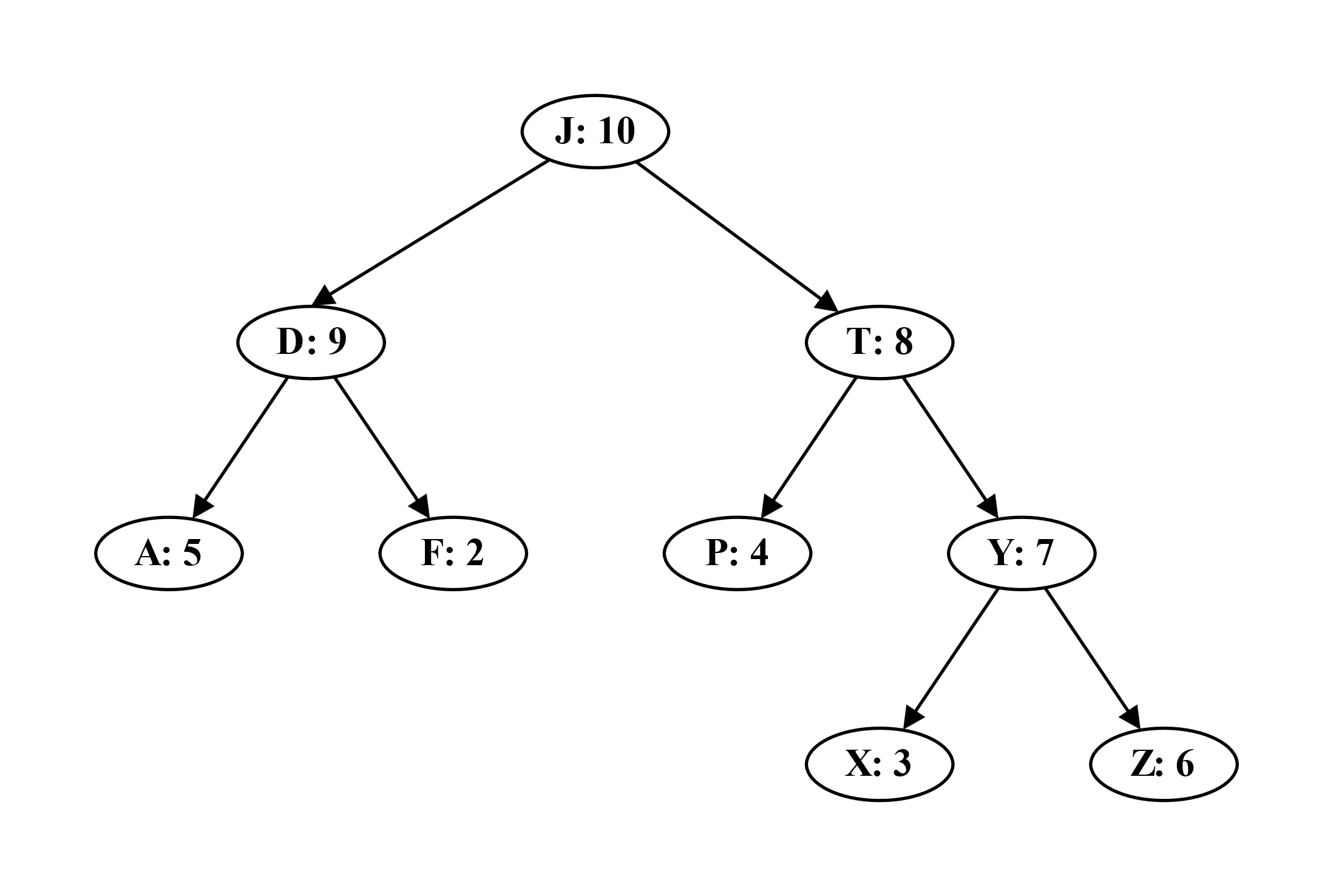
**Advanced Algorithms**

**Exercise for Lecture 8**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student Name** |  | **Student ID** |  |
| **Problem 1** |  | | |
| **Problem 2** |  | | |
| **Problem 3** |  | | |
| **Problem 4** |  | | |
| **Total Score** |  | | |
| **Notes** | Deadline: **2023-10-07 24:00**  Submission Format: ‘**Lecture8\_Name\_Student ID.docx**’, and please send to: **[algorithms\_23fall@163.com](mailto:algorithms_23fall@163.com)**.  This assignment is meant to be an evaluation of your **individual** understanding coming into the course and should be completed **without collaboration** or outside help. | | |

**Problem 1. [20 points]** Consider the following (**max-heap**) treap, where the keys are the letters and the priorities are the integers:



Show the result of inserting the key M, including any necessary rotations. Assume the priority generated for the key M is 15. **Then**, show the result of removing the key T, including any necessary rotations.

**Solution:**

**Problem 2. [25 points]** Lecture 7 introduced the 2-3-4 tree, in which every internal node (other than possibly the root) has two, three, or four children and all leaves have the same depth. Show how to maintain, for every node of a 2-3-4 tree, the height of the subtree rooted at as an attribute . Make sure that your implementation does not affect the asymptotic running times of searching, insertion, and deletion.

**Solution:**

**Problem 3. [25 points]** Given an interval tree and an interval , write an efficient algorithm MIN-INTERVAL-SEARCH() that returns an interval overlapping that has the minimum low endpoint, or if no such interval exists.

**Solution:**

**Problem 4. [30 points]** Suppose that as a new accountant, you must design a data structure that stores the history of all the transactions on the bank account. In keeping with corporate policy, this data structure should support insertion of past and future transactions, as well as deletion of existing transactions. For this problem, you may assume that no two transactions occur on the same date. The data structure should be able to support the following operations:

* INITIALIZE: Initialize the account. The initial balance in the account is $0. This operation should take time.
* INSTRANS(): Insert a given transaction at a given date. The sum can be either positive or negative, and should be added to the balance in the account starting from the following day. Notice that the date can be arbitrary (not necessarily today’s date). This operation should take time, where is the number of transactions in the database.
* DELTRANS(): Delete the transaction that occurs at the given date, if there is any. When a transaction is deleted, the corresponding sum should be subtracted from the balance in the account starting from the following day. This operation should take steps.
* BALANCE(): Returns the balance in the account at the beginning of the given date. This operation should take time.

Explain how to use a red-black tree to implement this data structure, give clear and concise descriptions of the operations, and argue why they meet the desired running times. No need to write code.

**Solution:**