Programming Assignment 3: Optimal Binary Search Tree

Problem Statement

In this assignment, you will be implementing the optimal binary search tree (OBST) algorithm. As input, you will be given a sample set of key-lookups. Using the frequency of each key-lookup, compute the optimal structure of a binary search tree for all future lookups. You can assume that the sample set accurately represents future lookups and that no key is left out. This means that there will be no ‘dummy keys.’ Print the cost of the optimal tree, as well as its structure in the format specified in the book (pg. 404). Do not include the dummy nodes.

Program Specifics

- Read input from standard in and print output to standard out. Input will an integer specifying the number of key lookups followed by a list of keys, each on its own line. All keys will be strings.
  - I will post sample input and output that you can use to test your program. However, I will be running your programs on a different (and probably much larger) input, so make sure to do your own tests.
- You must use the dynamic programming-based OBST algorithm, as discussed in class. The point of this assignment is to get experience with dynamic programming. If you were to write your program recursively, rather than iteratively, it would likely have exponential run-time. You don’t want this!
  - You should follow the pseudocode on pg. 402, but keep in mind that there are no dummy keys. You can work around this by either modifying the algorithm, or making all values in of $q_i$ equal to 0 (because they have 0 probability).
- You only need to print the optimal tree structure and its cost– you don’t have to implement a full binary search tree. Use the cost function defined in class and in the course notes.
- All code must be written in Java, Python or C/C++. If you use Python, please specify which version you used (i.e. 2.X or 3.X) when you turn it in.

Due Wednesday, May 22 @ 11:59 PM