Homework 3 Test Cases

Introduction

Our test cases are divided into 4 groups. We will use BST and $2D$ to denote binary search trees and the $2d$ trees respectively. All of the computational experiments start with an empty tree. We will denote it by T_e. For any $n \geq 0$, let

$$k_n = \begin{cases} 0 & n = 0 \\ 1000 & n = 1 \\ \lfloor \frac{k_{n-1}+k_{n-2}}{2} \rfloor & n \geq 2 \end{cases}; \quad d_n = n; \quad x_n = \begin{cases} 0 & n = 0 \\ 500 & n = 1 \\ \lfloor \frac{x_{n-1}+x_{n-2}}{2} \rfloor & n \geq 2 \end{cases}; \quad y_n = 500 - x_n.$$

I. Test cases for the show function

Case 1 (BST) Starting from T_e, insert the following sequence of items:

List 1: $[k_0,\ d_0], \ldots, [k_{15}, d_{15}]$

to T_e. Use the show function to create a dot file `t1.dot`.

Case 2 ($2D$) Starting from T_e, insert the following sequence of items:

List 2: $[x_0,\ y_0,\ d_0], \ldots, [x_{15},\ y_{15},\ d_{15}]$

to T_e. Use the show function to create a dot file `t2.dot`.

In your submission, please prepare the png files generated from each of the dot files. They should be named as `t1a.png` and `t1b.png` respectively.

II. Test cases for height computations

Case 3 (BST) Use a standard PRNG (e.g. `rand() function`) to generate a list of 200 data items with distinct keys (You may call it List 3). Insert each data item to T_e. Report the height of the tree (called T_3), in the form of a table, at the following intermediate steps:

n = No. of Nodes in the tree	Height of BST T_3	$lg\, n$	$\sqrt{n}$
0	...	N.A.	...
20	...	...	...
40	...	...	...
...	...	...	...
200	...	...	...

Case 4 ($2D$) Use a standard PRNG (e.g. `rand() function`) to generate a list of 200 data items with distinct keys:

$$\text{List4}: \quad [u_1, v_1, 1], \ldots, [u_{200}, v_{200}, 200] \text{ where } 0 \leq u_i, v_i \leq 50 \text{ for } i = 1, \ldots, 200).$$

Insert each data item to T_e. Report the height of the tree (called T_4), in the form of a table, at the following intermediate steps:

n = No. of Nodes in the tree	Height of BST T_4	$lg\, n$	$\sqrt{n}$
0	...	N.A.	...
20	...	...	...
40	...	...	...
...	...	...	...
200	...	...	...

III. Test cases for sequence of dictionary operations insert and delete where the data items have distinct keys

`Case 5:` (BST) First create a list of data items as follows:

Starting from the empty tree T_e, Insert 10 elements chosen from `List 1` randomly to the search tree. Let the 10 data items, in the order of insertion, be $a_1, a_2, \ldots, a_{10}$. Print the dot file of the tree obtained (named as `5a.dot`). Delete the element a_1 (should be at the root) and print the dot file of the resulting tree (named as `5b.dot`).

`Case 6:` ($2D$) Starting from the empty tree T_e, Insert 10 elements chosen from `List 2` randomly to the search tree. Let the 10 data items, in the order of insertion, be $b_1, b_2, \ldots, b_{10}$. Print the dot file of the tree obtained (named as `6a.dot`). Delete the element b_1 (should be at the root) and print the dot file of the resulting tree (named as `6b.dot`).

IV. Test cases for sequence of dictionary operations insert, delete and search where the data items may have duplicate keys

`Case 7:` (BST) Create the following list of items:

List 7: $[k_1,\ d_1], \ldots, [k_{10}, d_{10}], [k_1,\ d_{11}], \ldots, [k_{10}, d_{20}]$

Starting from the empty tree T_e, Insert all the elements from `List 7` to the search tree in the given order. Delete the element $[k_1,\ d_1]$ (should be at the root) and search for data items with the key k_1 and so on. Report the result in the form of a table as shown:

stage	data item at root	the root (after deletion)	search for data with key
0	$[k_1,\ d_1]$	$[k_x,\ d_x]$	$[k_1, d_{11}]$
1	$[k_x,\ d_x]$	$[k_{x'},\ d_{x'}]$	$[k_x, d_u]$ (it may not exists)
...	...	...	...
...	...	...	...
20	...	nil	nil

Print the dot file of the resulting tree right after stage 10 (named as `t7.dot`).

`Case 8:` $(2D)$ Create the following list of items:

List 8: $l_1, \ldots, l_{18}$

where

$[x_1,\ y_1,\ d_1] = l_1, \ldots, [x_6,\ y_6,\ d_6] = l_6$;

$[x_1,\ y_7,\ d_7] = l_7, \ldots, [x_6,\ y_{12},\ d_{12}] = l_{12}$;

$[x_7,\ y_1,\ d_1 3] = l_{13}, \ldots, [x_{12},\ y_6,\ d_{18}] = l_{18}$.

We will then perform the following test. Let T be the $2D$ tree formed by inserting $l_1, \ldots, l_{18}$ to an empty $2D$ tree in the given order. After that, We will do the following:

```
i=1;
While  (T is not empty) {
  delete the root node of T
  update T
  search if l_i is in T, if so, print l_i to the screen
  increment i;
  }
```

Print the dot file of the resulting tree when the tree has 12 (named as `t8a.dot`) elements and when the tree has 6 (named as `t8b.dot`) elements.