PLA12202 Diskreetti matematiikka — Pori

Assignment: Using rotations to decrease the height of a binary search tree — kevät 2020

A binary search tree (BST) is a commonly used data structure. In a BST each vertex has a unique value, called the vertex *key*. When x is a vertex, we let $k(x)$ be its key. We will assume $k(x)$ is an integer. A BST is a binary tree having the following property:

> If vertex y is in the left subtree of vertex x, then $k(y) \leq k(x)$. If vertex y is in the right subtree of vertex x, then $k(y) \geq k(x)$.

In general, it is desirable that a BST's height be small. In this assignment you will write Matlab routines that can be used to decrease the height of an existing BST.

For simplicity, in this assignment we assume that a vertex's label is the same as its key. In addition, we assume that the keys of a BST with n vertices are $1, 2, \dots n$. Figure 1 contains an example of such a BST. In Figure 1, each vertex label (except for the root) contains either 'L' or 'R' to identify whether the vertex is a left child or a right child. For example vertex 3 is the left child of vertex 5. The 'L' or 'R' letter is included solely for clarity. The BSTs shown in this assignment are drawn automatically and the position of the vertices in the BST are controlled by the Matlab software.

We will use the BST of Figure 1 often as an example.

In this assignment we will store a BST using a matrix, `A`, and a scalar, `root`. The root of the BST is given by `root`. For example, for the BST of Figure 1, `root`$= 2$. If a BST has n vertices, then `A` will have n rows. Row i of `A` contains information related to vertex i. The contents of the first two columns of row i are as follows:

$$\texttt{A}(i,1) = \text{the row in } \texttt{A} \text{ containing the left child of vertex } i \tag{1a}$$

$$\texttt{A}(i,2) = \text{the row in } \texttt{A} \text{ containing the right child of vertex } i \tag{1b}$$

If vertex i has no left child, then $\texttt{A}(i,1) = -1$. If vertex i has no right child, then $\texttt{A}(i,2) = -1$. For example, the `A` matrix of the BST of Figure 1 is as follows:

$$\texttt{A1} = \begin{bmatrix} -1 & -1 \\ 1 & 5 \\ -1 & 4 \\ -1 & -1 \\ 3 & 9 \\ -1 & 7 \\ -1 & -1 \\ 6 & -1 \\ 8 & 10 \\ -1 & -1 \end{bmatrix} \tag{2}$$

(From here on, to save space we will display the transpose of this `A`-matrix.)

It will be necessary to add more information to the `A` matrix. In particular two more columns will be needed for each vertex:

$$\texttt{A}(i,3) = \text{the row in } \texttt{A} \text{ containing the parent of vertex } i \tag{3a}$$

$$\texttt{A}(i,4) = \text{the height of the subtree whose root is vertex } i \tag{3b}$$

In fact both columns 3 and 4 can be computed from columns 1 and 2 of the `A`-matrix (1).

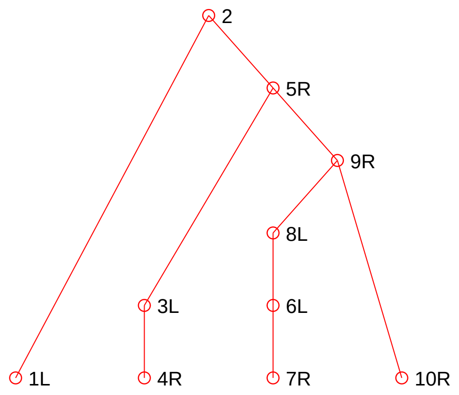

Figure 1: A binary search tree with 10 vertices.

Consider again the BST of Figure 1. When columns 3 and 4 from (3) are added to the A1-matrix of (2) we get the following:

$$\mathtt{A1} = \begin{bmatrix} -1 & 1 & -1 & -1 & 3 & -1 & -1 & 6 & 8 & -1 \\ -1 & 5 & 4 & -1 & 9 & 7 & -1 & -1 & 10 & -1 \\ 2 & 0 & 5 & 3 & 2 & 8 & 6 & 9 & 5 & 9 \\ 0 & 5 & 1 & 0 & 4 & 1 & 0 & 2 & 3 & 0 \end{bmatrix}^T \tag{4}$$

Consider for example vertex 9: its parent is vertex 5 (A1(9,3)) and the height of BST whose root is vertex 9 is 3 (A1(9,4)).

Your first task will be to write a routine **addParent** which is called as follows:

```
A = addParent(A, root);
```

Using only the first two columns of the A-matrix (1), routine **addParent** produces the third column (3a).

Your second task will be to write a routine **addHeight** which is called as follows:

```
A = addHeight(A, root);
```

Using only the first three columns of the A-matrix, routine **addHeight** produces the fourth column (3b).

To try to reduce the height of a BST, the *rotation* technique can be used. There are two basic rotations: right and left. These are depicted in Figure 2. When a rotation is performed a number of changes must be made to the BST. We consider here performing a left rotation at some vertex X. A right rotation is similar.

When we consider the left rotation of Figure 2, our starting situation is the BST shown on the left side of the figure:

- the parent of X is A
- the left-child of X is Y and the right-child of X is Z
- the left and right subtrees of Y are T_1 and T_2
- the left and right subtrees of Z are T_3 and T_4

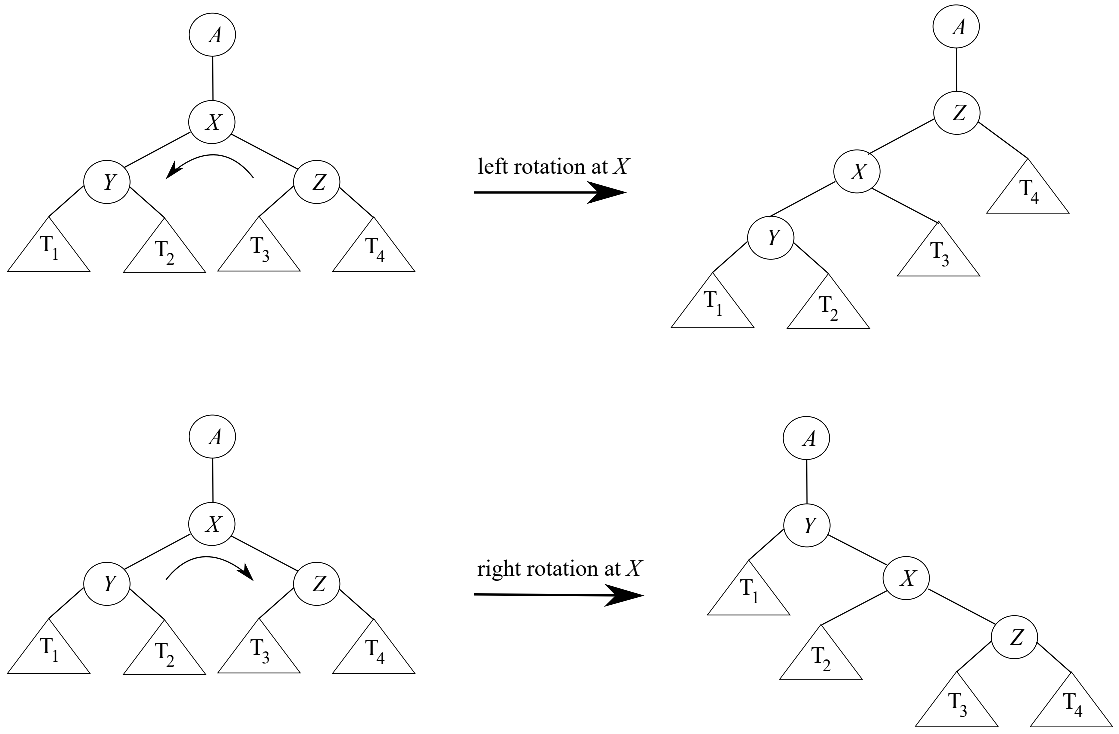

Figure 2: Left and right rotations at the subtree whose root is X.

Figure 2 shows the general situation when Y, Z, T_1, T_2, T_3 and T_4 all exist. To perform a left rotation at X, we only require that X and Z exist; one or more of Y, T_1, T_2, T_3 and T_4 can be empty. To perform a left rotation the changes in the following pseudocode must be made:

```
comment computing a left-rotation at vertex $X$ as in Figure 2
1. set the left child of $Z$ to $X$
2. set the right child of $X$ to the root of $T_3$
3. set the parent of $X$ to $Z$
4. set the parent of the root of $T_3$ to $X$
5. set the parent of $Z$ to $A$
6. set either the left child or the right child of $A$ to $Z$
7. compute the height of the subtree whose root is $X$
8. compute the height of the subtree whose root is $Z$
9. compute the height of the subtree whose root is $A$
```

Note that in many of these computations one must take into account cases where one or more vertices do not exist. For example, if T_3 does not exist, this will affect lines 2, 4 and 6. In addition, if X was the `root` of the starting BST, then after the left rotation, Z will be the new `root`.

If, for example, one performs a left rotation at vertex 5 of the BST of Figure 1, one obtains the BST of Figure 3. After the left rotation has been performed, the `A`-matrix of (4) changes

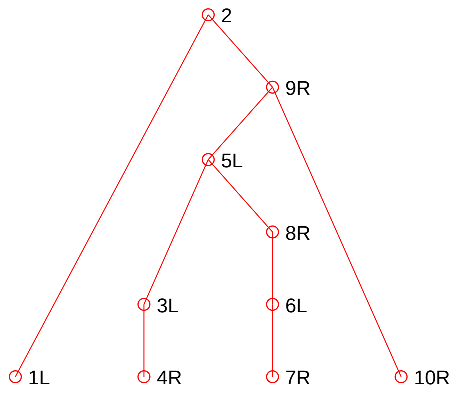

Figure 3: The binary search tree that results when a left rotation is done at vertex 5 of the BST of Figure 1.

to the following:

$$\begin{bmatrix} -1 & 1 & -1 & -1 & 3 & -1 & -1 & 6 & 5 & -1 \\ -1 & 9 & 4 & -1 & 8 & 7 & -1 & -1 & 10 & -1 \\ 2 & 0 & 5 & 3 & 9 & 8 & 6 & 5 & 2 & 9 \\ 0 & 5 & 1 & 0 & 3 & 1 & 0 & 2 & 4 & 0 \end{bmatrix}$$

There is no change in the `root` variable. If, however, a left rotation was done at vertex 2, then the value of `root` would also change.

Your third task will be to write a routine `leftRot`, which is called as follows:

```
[A, root] = leftRot(A, root, X);
```

This routine must make changes in the `A` matrix and in `root` corresponding to the above pseudocode.

In addition to a routine for a left rotation, a routine for a right rotation is also needed:

```
[A, root] = rightRot(A, root, X);
```

Your fourth task will be to write routine `rightRot`.

Three Matlab routines, `randBST`, `plotTreeA` and `rotations01`, are included with this assignment. To create a randomly generated BST having `n` vertices, one can run the following:

```
[A, root] = randBST(n);
```

To plot a BST from its `A` matrix one can run the following:

```
plotTreeA(A);
```

Note that for `plotTreeA` to work, the `A` matrix must contain the third column having the parent of each vertex. The figures of BSTs in this assignment were made using routine `plotTreeA`. To try to reduce the height of a BST using rotations, one can run the following:

```
[Anew,rootnew] = rotations01(A,root);
```

For routine `rotations01` to run correctly, it needs routines `addParent`, `addHeight`, `leftRot` and `rightRot`. When `rotations01` is run using the BST of Figure 1, the BST of Figure 4 results. When we compare Figure 1 and Figure 4, we see that routine `rotations01` reduced

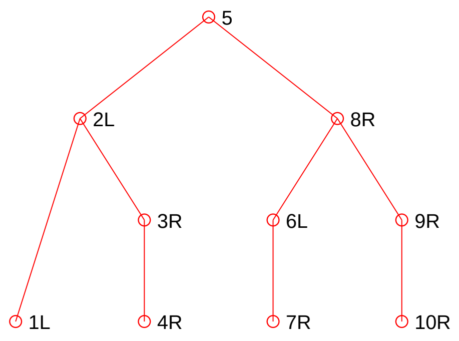

Figure 4: The BST that is obtained by running routine `rotations01` on the BST of Figure 1.

the BST height from 5 to 3.

1. Write the routine `addParent`.
2. Write the routine `addHeight`.
3. Write the routines `leftRot` and `rightRot`.
4. In parts (a), (b) and (c) a BST is given as a `A` matrix and a `root` value. For each BST use the `rotations01` routine to try to reduce the height of the BST. In each case, make a plot of the original BST and the BST that results after `rotations01` has been run.

   (a) $\begin{bmatrix} -1 & 1 & 2 & -1 & 4 & -1 & 6 & 3 & -1 & 8 & 10 & -1 \\ -1 & -1 & 5 & -1 & 7 & -1 & -1 & 9 & -1 & -1 & 12 & -1 \end{bmatrix}^T$, $\texttt{root} = 11$

   (b) $\begin{bmatrix} -1 & 1 & -1 & 3 & 4 & -1 & 2 & -1 & 8 & 7 & -1 & 11 & 12 & -1 \\ -1 & 5 & -1 & -1 & 6 & -1 & 9 & -1 & -1 & 13 & -1 & -1 & 14 & -1 \end{bmatrix}^T$, $\texttt{root} = 10$

   (c) $\begin{bmatrix} -1 & 1 & 2 & -1 & -1 & -1 & -1 & 3 & -1 & -1 & -1 & 11 & 10 & -1 \\ -1 & -1 & 4 & 5 & 6 & 7 & -1 & 9 & 13 & 12 & -1 & -1 & 14 & -1 \end{bmatrix}^T$, $\texttt{root} = 8$

5. The purpose of this question is to see whether `rotations01` will in general decrease the height of a BST having 1000 vertices. Do the following:

   - Using `randBST`, generate randomly 50 different BSTs having 1000 vertices. Compute and store the height of each BST.
   - Run each randomly generated BST through `rotations01`. Compute the height of the new BST and store it.
   - Let h_1 be the height of the randomly generated BST before it has been run through `rotations01`. Let h_2 be the height of the BST after it has been run through `rotations01`. Compute a 95% confidence interval for the difference $h_1 - h_2$. (Matlab's `ttest2` function can be used.)