3 Trees: traversal and analysis of standard search trees

Summer Term 2011
Jan-Georg Smaus

Binary Search Trees
- Binary trees for storing sets of keys (in the internal nodes of trees), such that the operations
  - find
  - insert
  - delete (remove)
are supported.
- Search tree property: All keys in the left subtree of a node \( p \) are smaller than the key of \( p \), and the key of \( p \) is smaller than all keys in the right subtree of \( p \).
- Implementation:

Search tree property:
- All keys in the left subtree of a node \( p \) are smaller than the key of \( p \), and the key of \( p \) is smaller than all keys in the right subtree of \( p \).

Implementation:

Traversals of trees
- Traversal of the nodes of a tree
  - for output
  - for calculating the sum, average, number of keys ...
  - for changing the structure

Most important traversal orders:
1. Preorder = NLR (Node-Left-Right)
   first visit the root, then recursively the left and right subtree (if existent)
2. Postorder = LRN
3. Inorder = LNR
4. The mirror image versions of 1-3

Preorder
Preorder traversal is recursively defined as follows: Let \( p \) be the root.
- Visit \( p \),
- traverse the left subtree of \( p \) in preorder,
- traverse the right subtree of \( p \) in preorder.

Preorder implementation
```java
// Preorder Node-Left-Right
void preOrder();
    preOrder(root);
    System.out.println (i);
}
void preOrder(SearchNode n)
if (n == null) return;
System.out.print (n.content+" ");
preOrder(n.left);
preOrder(n.right);
```

Postorder
```
// Postorder Left-Right-Node
void postOrder();
    postOrder(root);
    System.out.println (i);
}
```
...
The traversal order is: first the left subtree, then the root, then the right subtree:

// Inorder Left-Node-Right
void inOrder()
    {inOrder(root);
     System.out.println();
    }
// ...
Internal path length

Internal path length is a measure for judging the quality of a search tree $t$. Recursive definition:

1. If $t$ is empty, then $I(t) = 0$.
2. For a tree $t$ with left subtree $t_l$ and right subtree $t_r$:
   
   $I(t) = I(t_l) + I(t_r) + n_l$ inner nodes of $t$

   Apparently:
   
   $I(t) = \sum_{p} (\text{depth}(p) + 1)$
   
   $p$ internal node in $t$

Average search path length

For a tree $t$ the average search path length is defined by:

$D(t) := I(t)/n = \#\text{ nodes in } t$

Question: What is the size of $D(t)$ in the

- best
- worst
- average

case for a tree $t$ with $n$ internal nodes?

Internal path: best case

We obtain a complete binary tree

Internal path: worst case

We obtain a random balanced tree

Random trees

- Without loss of generality, let $\{1, \ldots, n\}$ be the keys to be inserted.
- Let $s_1, \ldots, s_n$ be a random permutation of these keys.
- Hence, the probability that $s_i$ has the value $k$, $P(s_i=k) = \frac{1}{n}$.
- If $k$ is the first key, $k$ will be stored in the root.
- Then the left subtree contains $k-1$ elements (the keys $1, \ldots, k-1$)
  and the right subtree contains $n-k$ elements (the keys $k+1, \ldots, n$).

Expected internal path length

- $EI(n)$: Expectation for the internal path length of a randomly generated binary search tree with $n$ nodes
- Apparently we have:
  
  $EI(0) = 0$
  
  $EI(1) = 1$
  
  $EI(n) = \frac{1}{n} \sum_{k=1}^{n} (EI(k-1) + EI(n-k)) + n$
  
  $= n + \frac{1}{n} \left( \sum_{k=1}^{n} EI(k-1) + \sum_{k=1}^{n} EI(n-k) \right)$

- Assume: $EI(n) = 1.386 n \log_2 n - 1.846 n + O(\log n)$. 

Proof (1)

\[ E(I(n+1)) = (n+1) + \frac{\gamma}{n+1} \sum_{k=1}^{n} E(I(k)) \]

and hence

\[ E(I(n+1)) = (n+1)^2 + \frac{\gamma}{n+1} \sum_{k=1}^{n} E(I(k)) \]

From the last two equations it follows that

\[ (n+1) E(I(n+1)) - n E(I(n)) = (n+1)^2 + \frac{\gamma}{n+1} \sum_{k=1}^{n} E(I(k)) \]

Thus,

\[ E(I(n+1)) = \frac{2n + 1 + \gamma}{n+1} E(I(n)) + \frac{2}{n+1} \]

Proof (2)

By induction over \( n \) it is possible to show that for all \( n \geq 1 \):

\[ E(I(n)) = 2(n+1) H_n - 3n \]

\[ H_n = 1 + \frac{1}{2} + \ldots + \frac{1}{n} \]

is the \( n \)-th harmonic number, which can be estimated as follows:

\[ H_n = \ln n + \gamma + \frac{1}{2n} + O\left(\frac{1}{n^2}\right) \]

where \( \gamma \approx 0.5772 \ldots \) the so-called Euler constant.

Proof (3)

Thus,

\[ E(I(n)) = 2n \ln n - (3 - 2\gamma) + n \ln n + 1 + 2\gamma + O\left(\frac{1}{n}\right) \]

Observation

- Search, insertion and deletion of a key in a randomly generated binary search tree with \( n \) keys can be done, on average, in \( O(\log n) \) steps.
- In the worst case, the complexity can be \( \Omega(n) \).
- One can show that the average distance of a node from the root in a randomly generated tree is only about 40% above the optimal value.
- However, by the restriction to the symmetrical successor, the behaviour becomes worse.
- If \( n^2 \) update operations are carried out in a randomly generated search tree with \( n \) keys, the expected average search path is only \( \Theta(\sqrt{n}) \).

Resulting binary tree after \( n^2 \) updates
Structural analysis of binary trees

Question: What is the average search path length of a binary tree with $N$ internal nodes if the average is made over all structurally different binary trees with $N$ internal nodes?

Answer: Let $I_N = \text{total internal path length of all structurally different binary trees with } N \text{ internal nodes}$

$B_N = \text{number of all structurally different trees with } N \text{ internal nodes}$

Then $I_N / B_N =$

Summary

The average search path length in a tree with $N$ internal nodes (averaged over all structurally different trees with $N$ internal nodes) is:

$\frac{1}{N} \cdot \frac{I_N}{B_N}$