Universitetet i Oslo
Institutt for Informatikk

A. Maus, R.K. Runde, I. Yu

INF2220: algorithms and data structures
Series 2

Topic Balanced trees

Issued: 31. 08. 2017

Classroom

Exercise 1 (Analysis of running time)

1. The following program snippet sorts an integer array `int[] A = new int[n]`. What's the order of running time for that in big-O notation.

```
for (int i = 0; i < n; i++) {
        minj = i;
        for (j = i + 1; j < n; j++) {
                if (A[j] < A[minj]) {
                        minj = j;
                }
        }
        bytt(i, minj);
}
```

2. What's the order of running time for that in big-O notation for the following fragment. Pay attention especially for the conditional inside the second nested loop.

```
for (int i = 1; i <= n; i++) {
 for (int j=1; j <= i*i; j++) {
   if (j % i == 0) {
     for (int k = 0; k < j; k++ )
       sum++;
   }
  }
 }
```

3. For a given $n > 0$, which value will be found in the variable `L2` after executing the following program snippet. The answer should be given as a function of n. Of which order is the running time of this program in big-O notation?

```
i = 1;
L2 = -1;
while (i <= n) {
        i = i * 2;
        L2++;
}
```

Exercise 2 Big-O Fibonacci

The following program will calculates the n'th Fibonacci number. Determine the running time in big-O notation:

```
int fib(int n) {
    if(n<=1) {
        return 1;
    }
    return fib(n-1) + fib(n-2);
}
```

Exercise 3 (Height of a binary tree)

1. Exercise 4.5 in MAW.
2. Show that for a balanced tree with N nodes, the height is $\lfloor log_2(N) \rfloor$.

Exercise 4 (Trees) Given a binary tree (not necessesarily a binary search tree), with an arbitary number of nodes. Write pseudo-code for a function which

(a) returns the smallest value in the tree

(b) returns the largest value in the tree

(c) returns the length of the longest path from the root to a null pointer

(d) returns the length of the shortest path from the root to a null pointer

Exercise 5 (Nodes in a binary tree)

1. Show that in a (non-empty) binary tree with N nodes, there are $N + 1$ `null` links representing children.
2. A *full* node is a node with 2 children. Prove that the number of full nodes + 1 is equal to the number of leaves in a non-empty binary tree.

Exercise 6 (Red-black tree) Build, step by step, red-black trees that result from inserting the following sequences of elements:

1. `41 38 31 12 19 8`
2. `A L G O R I T H M`

Exercise 7 (B-trees)

1. Assume an empty *B-Tree* with $M = 4$ and $L = 4$. Insert the following values in the given order:

   `A B C D G H K M R W Z`

   Show how the tree changes step by step.

2. Assume an empty *B-Tree* with $M = 5$ and $L = 5$. Insert the following values in the given order:

   `2 6 17 20 24 25 27 29 30 31 32 5 21 1 40 45 50 70`

Show how the tree changes step by step.

3. Assume an empty *B-Tree* with $M = 3$ and $L = 4$.

   - Insert the following values:

     `61 27 19 5 7 25 36 4 42 2 13 44 62 98 43 16 24 29 15`

     Show how the tree changes step by step.

4. Assume an empty *B-Tree* with $M = 3$ and $L = 2$.

   - Insert the following values:

     `3 14 1 59 26 5 89 79`

     Show how the tree changes step by step.
   - Delete `59, 5, 3, 1` and `26`. Draw the tree after each deletion.

Lab

Exercise 8 (Binary search) Implement binary search in an array of integers.

Exercise 9 (Rotations) Implement zig and zig-zag rotation for arbitrary binary search trees (i.e. without the color coding of red-black trees).

Exercise 10 (B-Trees) Write a *general* implementation of insertion for a B-Tree. Note that you have to restructure the tree in case the leaf is full after the insertion.

Exercise 11 (Red-black trees) In the previous week, you were asked to analyze the play *Vildanden* from Henrik Ibsen by building a binary search tree. Instead of building a BST, use a *red-black* tree for the play Vildanden this week. Similarly, all the words which are different from upper case and lower case are considered to be the same. Each node in the tree is corresponding to a unique word in the file. Each node should remember the frequency of the corresponding word appear in the play. Calculate the depth of the left and right subtrees for the red-black tree, and compare with the depth of the BST from last week.