Exam INF102@UiB, spring 2013
- An unofficial translation

Markus Dregi

No auxiliary resources allowed. You do not have to give java code in your solutions, pseudo code or a description is sufficient. Justify all your answers!

Task 1

In this task we are to look at different approaches for evaluating a polynomial function $f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$ for a given value of x. You can assume that the values of a_i is given as an array a of length $n + 1$, such that $a[i] = a_i$. You can perform the operations $+$ and $*$ in constant time. Note that you cannot calculate a^b in constant time.

a) The following code snippet gives a straight forward implementation that calculates he function value of f for a given value of x. Derive a formula for the exact number of $+$ and $*$ operations that are executed. The formula is to be a function of n and you do not have to differentiate between the two operations ($+$ and $*$). The formula should hence give the sum of the number of $+$ and $*$ operations that are executed.

```
fx = a[0];
for (i=1;i<=n;i++) {
    prod = a[i];
    for (j=1;j<=i;j++)
        prod = prod * x;
    fx = fx + prod;
}
return fx;
```

b) Explain how you can rewrite the code such that the inner most loop is not necessary anymore (the loop iteration over values of j). Do the same analysis as in a).

c) For $n = 4$ we can rewrite $f(x)$ as $a_0 + x(a_1 + x(a_2 + x(a_3 + x(a_4))))$. This formulation is called Horners formula and can be generalized to arbitrary values of n. Write a code snippet that uses this way of computing $f(x)$ and analyze the code as you did in a) and b). Tip: Start your computations from the inner most pair of parenthesis.

Task 2

a) Describe and analyze an efficient algorithm that given a heap T and a key k outputs every key in T that has value larger than k.

b) Explain why no algorithm based on compares can build a binary search tree (BST) and execute less than $\log(N!) \sim N \log N$ compares.

Task 3

Assume that you use the Union-Find data structure to represent partitions over the universe $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Initially, we have every element in the universe as a singleton (every element is alone in its set). We then execute `union` on the following pairs: $(4, 7)$, $(3, 4)$, $(3, 8)$, $(2, 5)$, $(1, 2)$, $(6, 9)$, $(6, 1)$, $(7, 5)$.

1. Display how the final data structure looks like if we always set the lowest numbered root-node to point to the higher numbered root node.

2. Display how the final data structure looks like if we always set the root node with the least elements underneath to point to the root node with the most elements underneath. We break ties by the rule from a).

3. Now, we are to represent partitions over the universe $\{1, 2, \ldots, n\}$ for some positive integer n. How many compares do we do in the worst case when executing the `find` operation if we did the `union` as described in a) and b)? Give the answers as functions of n.

Task 4

Let $G = (V, E)$ be an undirected graph with positive weights on its edges.

1. Prove that if you replace all of the weights on the edges in G by the weights multiplied with some positive number t then a minimum spanning tree in G is also minimum after the replacement.

2. Prove that if the weights are unique, meaning that no two edges have the same weight, than the minimum spanning tree is unique.

3. Assume all of the edges to have unique values.

   1) Is the lightest edge (the one with the lowest weight) always in the minimum spanning tree of G?

   2) Can the heaviest edge be in the minimum spanning tree of G?

   3) Is the lightest edge in a cycle of G always in the minimum spanning tree of G?