TTIT33 DALGOPT – Algoritmer och Optimering
DALG part of the exam 2005-12-14
Outline of the proposed solutions for selected problems.

1. (a) (2 p)
   - The big-O notation characterizes the asymptotic behaviour of a function. For a small size of input data (smaller than $n_0$ in the definition of the $O$ relation) the asymptotic time complexity provides no information. The measurement might have been done on such data.
   - The input data could have been chosen in such a way that they were close to the best case data for A and close to the worst case data for B.
   - The big-O notation gives only the upper bound for time complexity. If $A$ is in $O(n^2)$ it may also be in e.g. $O(n \log n)$.

The problem discusses comparison of running time of two algorithms on the same data. Thus it implicitly assumes that the algorithms were run on the same machine. So the guess that $A$ was run on faster machine and $B$ on a slower one is not a relevant explanation. Similarly, a guess that one of the algorithms might have required communication with external memory while the other did not would make comparison of running time irrelevant.

(b) The program consists of two nested loops. The number of iterations does not depend on input data. For $i = k$, where $k = 1, \ldots, n-1$, the inner loop executes $n-k$ times, thus the total number of executions the if statement is $(n-1) + (n-2) + \ldots + 1 = (n(n-1))/2$. The if statement, executes in a constant time. The constant depends on data. The worst case is if for every $i$ each of the elements $T[i+1], T[i+2], \ldots, T[n]$ is smaller or equal $T[i]$, thus if $T$ is sorted in decreasing order.

Thus the worst case time complexity of the program is $O(n^2)$.

Beware of incorrect justification saying that we have two nested loop each of them executing $n -1$ times.

(c) Denote by $T(n)$ the execution time of the algorithm on dat $n$. Thus $T(0)$ is a constant time $t_0$, and for $n > 0$ we have $T(n) = T(n-1) + t_1$ where $t_1$ is a constant time needed for computation of $n-1$ from $n$, for multiplication of the computed value of $Fact(n-1)$ by $n$ and for returning the result. Thus, by unfolding of this recurrence equation we get $T(n) = T(n-2) + 2t_1 = T(n) + nt_1 = t_0 + nt_1$. Hence the time complexity of the algorithm is $O(n)$.

2. There are several possibilities to organize the elements as the requested trees. We show one possible solution for each case. The justification should summarize the respective definitions.

(6 p)
(a) AVL-tree is a binary search tree with height-balance property (G&T 4th edition p. 429)

```
    7
   / \  \
  5 11
 / \ / \ /
3 6 10 12
 / / / /
2 8
```

(b) (2,3) tree as discussed in the lecture is a multiway search tree with the properties (analogous to (2,4) trees (G&T 4th ed. p.455)):
   - Size property: every internal node has at most three children.
   - Depth property: all the external nodes have the same depth.

```
    8
   / \  \
  3 6 11
 / | \ / \ /
2 5 7 10 12
```

(c) Heap is a complete binary tree such that in every node other than the root the key stored at this node is greater than or equal to the key stored at the node’s parent. (G&T 4th ed.p. 334)

```
   2
  / \  \
 3 5
 / \ / \ /
6 7 8 10
 / /
11 12
```

3. (3 p)

(a) The solution is not given. It should follow similar examples discussed in G&T and/or in lecture slides. For QuickSort there may be several variants of pivot selection. There are also in-place versions of QuickSort where partition is done by swapping elements. Any of the QuickSort variants can be used in the answer.

(b) For InsertionSort the best case data is a sorted sequence. The complexity of the algorithm in this case is $O(n)$. The execution of the algorithm reduces essentially to a check performed by a single traversal of the data. The worst case is when the data is sorted in reverse order. In that case each element is shifted to the beginning of the sorted part of the sequence, so that the worst case time complexity is $O(n^2)$.

For QuickSort the best case data is such that the method for pivot selection selects always the median element of the subsequence. In this case the complexity is $O(n \log n)$, since at each stage the problem is divided into two subproblems of equal size.

The worst case data is such that the method for pivot selection always selects the minimal element of the sequence (or the maximal element). Thus the original problem of size $n$ is divided into the subproblem of size $n - 1$ and the empty subproblem. This results in the worst case complexity $O(n^2)$.
4. Explanation not given. (2 p)
   (a) (1 p)
   
   \[
   \begin{array}{cccc}
   0 & 1 & 2 & 3 \\
   14 & 11 & 6 & 29 \\
   \end{array}
   \]

   (b) (1 p)
   
   \[
   \begin{array}{cccc}
   0 & 1 & 2 & 3 \\
   - & 11 & 6 & - 14 \\
   \end{array}
   \]

5. (3 p)
   (a) Depth-first search for given order of choosing neighbours: c,a,b,f,e,d (1 p)
   (b) Breadth-first search for given order of choosing neighbours: c,a,f,e,b,d (1 p)
   (c) This directed acyclic graph has only one topological ordering: c,a,b,d,f,e (1 p)