S. Schwoon / D. Suwimonteerabuth Winter Term 2008/09

Fundamental Algorithms

Example Problems

1. **Growth of functions**

   (a) Let A and B be two algorithms that take time in $\mathcal{O}(n)$ and $\mathcal{O}(n^2)$, respectively. Is it true or false to argue that A runs faster than B on *all* inputs? Justify your answer.

   (b) Consider the algorithm below whose input is the number n. Give a function f such that $f(n)$ is the value of s after running the algorithm.

```
s := 0;
for i = 1 to n do
    for j = i to n do
        if j = 1 then
            for k = j to n do
                s := s + 2;
            end
        else
            s := s + 1;
        fi
    end
end
```

   (c) Are the following statements true or false? Justify your answer.

   i. $3n \in \mathcal{O}(n \log n)$

   ii. $\frac{3}{2}n \in \Omega(n^{3/2} - n)$

2. **Sorting**

   (a) Consider the algorithm `heapify` below. Suppose that a is an array with the five elements $3, 10, 5, 4, 7$ (in this order). Is a a heap? What are the contents of a after running the algorithm on it, with $i = 1$ and $k = 5$? Do the new contents have the heap property?

Input: an array a with indices $1 \ldots n$, indices k and i
Output: if the slice $i+1 \ldots k$ is a heap initially, then afterwards $i \ldots k$ is a heap

```
tmp := a[i];
while 2 * i ≤ k do
    largest := 2 * i;
    if a[largest + 1] > a[largest] then
        largest := largest + 1;
    fi
    if a[largest] > tmp then
        a[i] := a[largest];
    fi
    i := largest;
od
a[i] := tmp;
```

(b) There is a mistake in the algorithm above. Can you spot it? Provide an input (i.e., values for a, i, k) where the mistake surfaces.

3. **Searching**

(a) Draw the ordered binary tree by inserting new nodes into the empty tree in the following order:

$$5, 3, 2, 1, 9, 6, 7, 10, 4, 8 \; .$$

(b) Draw some ordered AVL tree containing the same elements as in (a).

(c) Write an algorithm for the following task: You are given an ordered binary tree containing n nodes altogether, and an array a with indices $1 \ldots n$. After running your algorithm, the array a should contain the values stored in the binary tree, in descending order. (Assume that a node in the binary tree is a record `BinNode`, with the items `value`, `left`, and `right`, as in the lecture.) If you cannot write down an algorithm, describe how you would solve the problem in plain language.

4. **Graphs**

(a) Draw a graph with nodes $a \ldots i$ that contains exactly two strongly connected components, and it is possible to generate the following post-order numbering in this graph.

$$a, b, c, d, e, f, g, h, i \; .$$

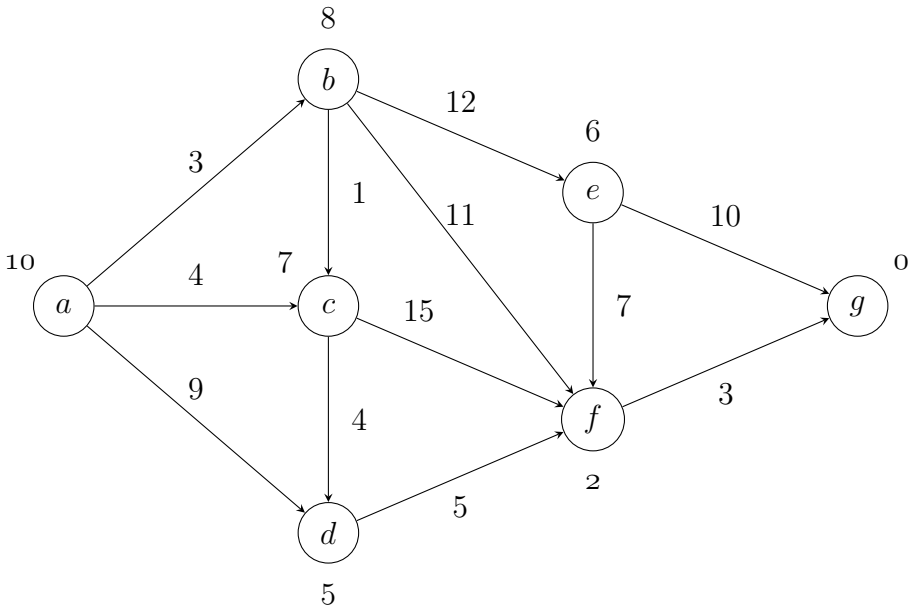
(b) Here is a new proposal for computing the strongly connected components of a given directed graph G:

> Perform a *pre-order* numbering on G and generate the graph G_R by reversing all edges of G. Then, while G_R still contains nodes, pick the node n with the lowest number and remove all nodes reachable from n. Each set of removed nodes is an SCC.

Is this proposal correct? If yes, provide an argument why it is. If not, provide an example where the algorithm fails and explain why.

(c) Below you see two directed graphs. In both of them, we are interested in finding the shortest distance from node a to node g using the A^* algorithm. The values of the heuristic functions are given next to the nodes. Check, for both graphs, whether the given heuristic is monotone. (You are not required to simulate A^* itself.)

i.



ii.