Algorithms and Data Structures II (1DL230)
Uppsala University – Autumn 2009
Tutorial 1 (1.5h)

Nikos Nikoleris and Pierre Flener

September 11, 2009

1 Asymptotic growth rates ($\sim 30\%$)

Rank the following functions by order of growth; that is, find an arrangement g_1, g_2, ..., g_{12} of the functions satisfying $g_1 = \Omega(g_2)$, $g_2 = \Omega(g_3)$, ..., $g_{11} = \Omega(g_{12})$. Partition your list into equivalence classes such that $f(n)$ and $g(n)$ are in the same class if and only if $f(n) = \Theta(g(n))$.

2^{2^n}	$n!$
$n2^n$	$n-7$
$n \log n$	$\log^6 n$
$\log 4n^4$	$\frac{n}{\log n}$
$\frac{n^6}{\log^8 n}$	$\frac{\log n}{n^5}$
$3n^9 - 7$	e^n

2 The substitution method, the recursion-tree method & the master theorem ($\sim 30\%$)

Give asymptotic upper and lower bounds for $T(n)$ in each of the following recurrences. Assume that $T(n)$ is constant for sufficiently small n. Make your bounds as tight as possible, and justify your answers.

1. $T(n) = 3T(n/2) + n \lg n$
2. $T(n) = 5T(n/5) + n/\lg n$
3. $T(n) = 2T(n/4) + \sqrt{n}$
4. $T(n) = 3T(n/3 + 5) + n/2$
5. $T(n) = 2T(n/2) + n/\lg n$
6. $T(n) = T(n/2) + T(n/4) + T(n/8) + n$
7. $T(n) = T(n-1) + 1/n$

8. $T(n) = T(n-1) + \lg n$
9. $T(n) = T(n-2) + 2\lg n$
10. $T(n) = T(\sqrt{n}) + 1$

3 Divide & Conquer ($\sim 40\%$)

3.1 Towers of Hanoi

Consider the following basic problem. There are three pegs and n distinct size discs. The discs are placed in the left peg in the order of their sizes. The smallest one is at the top while the largest one is at the bottom.

1. Find a way to move all the discs from the left to the right peg (using the middle peg as an intermediate buffer) following the rules:
   - Each time only one disc can be moved from on peg to another.
   - No disc can be placed on top of a smaller one.
2. Give the corresponding pseudocode.
3. Find the recurrence $T(n)$ that describes the running time of your algorithm for n discs.
4. Can the master method be applied to this recurrence? Give a tight asymptotic bound for this recurrence.

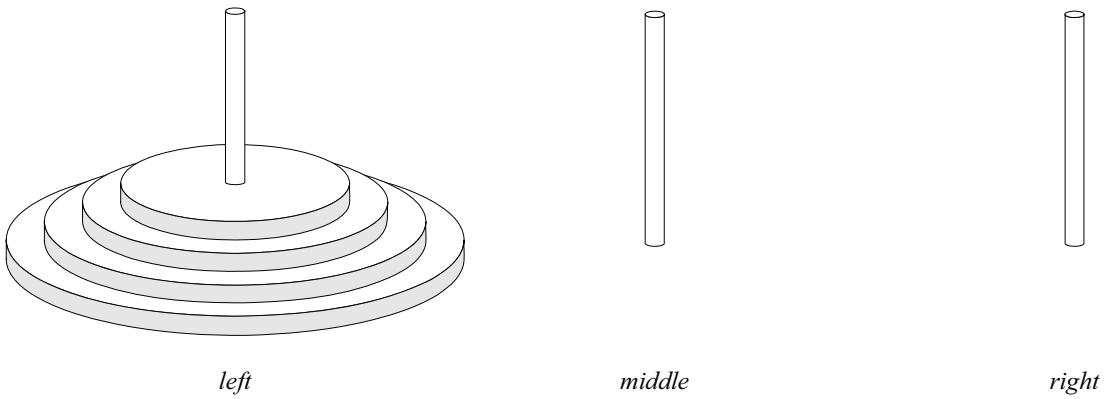

Figure 1: Towers of Hanoi

3.2 Unimodal Search

An array $A[1..n]$ is unimodal if it consists of an increasing sequence followed by a decreasing sequence, or more precisely, if there is an index $m \in 1, 2, \ldots, n$ such that:

- $A[i] < A[i+1]$ for all $1 \leq i < m$, and
- $A[i] > A[i+1]$ for all $m \leq i < n$.

Give an algorithm to compute the maximum element of a unimodal input array in $O(\log n)$ time. Prove the bound on its running time.