Technische Universität München
Department of Computer Science
Chair for Efficient Algorithms
Prof. Dr. Sami Khuri

WS 1999/2000
Problem Set 2
November 9, 1999

Efficient Algorithms and Data Structures

Problem 1 (4+4 credits)

The *level* of a node in a tree is defined to be its distance from the root node. The root has level 0. The *external path length* $E(T)$ of a tree T is the sum of the levels of all its leaves. The *internal path length* $I(T)$ is the sum of the levels of all its internal nodes.

a) Prove that in a t-ary tree (i.e., a tree where every internal node has exactly t children) with n internal nodes, there are exactly $(t-1)n+1$ leaves.

b) Determine the maximum values of $E(T)$ and $I(T)$ over all possible binary trees T with n internal nodes.

Problem 2 (15 credits)

The following algorithm select(k, S) finds the element of rank k in a file S of n numbers. (The smallest number has rank 1, the second smallest has rank 2, etc. The term “file” in this problem refers to a multi-set, or a set that can contain duplicates.)

select(k, S)

1. If $n < 50$ sort the file using insertion sort. Return the kth smallest. (Note that insertion sort does at most $n(n+1)/2$ comparisons.)

2. Divide S arbitrarily into $t = \lceil n/7 \rceil$ subfiles of size 7 (padding with infinities to make all the files be of size 7).

3. Sort each group of 7 using insertion sort.

4. Form a new file T consisting of the middle element of each of the files of 7. Compute $m = \text{select}(\lceil t/2 \rceil, T)$

5. Compare each element of S to m (these are n comparisons). Using these comparisons, partition S into three new files S_0, S_1 and S_2, which are the elements of S that are less than m, equal to m, and greater than m respectively.

6. If $k \leq |S_0|$ then compute select(k, S_0) and return the answer. Else, if $k \leq |S_0| + |S_1|$ return m. Otherwise compute select$(k - |S_0| - |S_1|, S_2)$ and return the answer.

Let $T(n)$ be the worse-case number of comparisons done by this algorithm on a file of size n. (That is, for fixed n, run the algorithm on all files of size n, and run it for all values of k from 1 to n. Take the maximum number of comparisons done in all these runs. This is $T(n)$.)

Prove that $T(n)$ is $O(n)$. Note: Your proof should be rigorous. One way to do this is to find a linear polynomial $an + b$, and prove by induction that $T(n) \leq an + b$ for all $n \geq 7$.

Problem 3 (5+5+7 credits)

Given an integer N, let N_b be its binary representation and define:

- the population count function V_N: number of ones in N_b
- the cumulated population count function P_N: number of ones in the binary representations of all numbers less than N
- the ruler function Y_N: number of trailing ones in N_b
- the cumulated ruler function R_N: number of trailing ones in the binary representations of all numbers less than N

<u>Example:</u> If $N = 14$ ($N_b = 1110$) then $V_N = 3$, $P_N = 25$, $Y_N = 0$ and $R_N = 11$.

In the following, "$\log N$" means "$\log_2 N$".

a) Show that $P_{2^n} = n2^{n-1}$ and $R_{2^n} = 2^n - 1$.

b) Show that $P_N = P_{2^{\lfloor \log N \rfloor}} + \left(N - 2^{\lfloor \log N \rfloor}\right) + P_{N - 2^{\lfloor \log N \rfloor}}$.

   <u>Hint:</u> Note that $2^{\lfloor \log N \rfloor}$ of the numbers less than N start with 0 (if written with $\lfloor \log N \rfloor + 1$ bits).

c) Show that $P_N = \frac{1}{2} N \log N + O(N)$ and that $R_N = N + O(\log N)$.

   <u>Note:</u> Here, the notation $f(n) = g(n) + O(h(n))$ means that $|f(n) - g(n)|$ (the absolute value of the difference between $f(n)$ and $g(n)$) is $O(h(n))$.