Technische Universität München
Institut für Informatik
Lehrstuhl für Effiziente Algorithmen
Lehrstuhl für numerische Programmierung

WS 2002/2003
Worksheet 1

17.10.2002

Fundamental Algorithms

Exercise 1

Consider two different algorithms, $\mathcal{A}$ and $\mathcal{B}$, that solve the same problem. $\mathcal{A}$ shall require $500n^2 - 16n$ operations for a problem of size n; $\mathcal{A}$ shall require $\frac{1}{3}n^3 + \frac{11}{2}n + 7$ operations for the same problem.

(a) If you had a problem of size $n = 256$, which algorithm would you choose?

(b) Which algorithm would you choose, if you had a problem of size $n \geq 2000$?

Exercise 2

HOMECOMPUTER shall be a machine that can perform 10^9 operations per second. Consider that we have five different algorithms for a specific problem. For each algorithm i, we know the number of operations $T_i(n)$ it will perform on a problem of size n:

$$\begin{aligned} T_1(n) &= 6\,000\,000 \cdot n \\ T_2(n) &= 60\,000 \cdot n \log n \\ T_3(n) &= 0.003 \cdot n^2 \\ T_4(n) &= 10^{-6} \cdot n^3 \\ T_5(n) &= 10^{-18} \cdot 2^n \end{aligned}$$

For each algorithm compute the size $n_{\max}$ of the largest problem the respective algorithm can solve within 1 second (1 minute, 1 hour, ...). Enter the maximal problem sizes into the following table:

	1 second	1 minute	1 hour	1 day	1 month (30d)	1year (365d)
$n_{\max}(T_1)$	166	10 000	600 000	$1.44 \cdot 10^7$	$4.32 \cdot 10^8$	$5.26 \cdot 10^9$
$n_{\max}(T_2)$	1 569	62 748	$2.8 \cdot 10^6$	$5.59 \cdot 10^7$	$1.42 \cdot 10^9$	$1.55 \cdot 10^{10}$
$n_{\max}(T_3)$	577 350	$4.47 \cdot 10^6$	$3.46 \cdot 10^7$	$1.69 \cdot 10^8$	$9.29 \cdot 10^8$	$3.24 \cdot 10^9$
$n_{\max}(T_4)$	46 415	181 712	711 379	$2.05 \cdot 10^6$	$6.37 \cdot 10^6$	$1.46 \cdot 10^7$
$n_{\max}(T_5)$	89	95	101	106	110	114

Exercise 3

Write a program for a RAM that will compute the n-th Fibonacci number F_n. Specify the starting configuration (esp. in which register the input n should be placed), and determine the number of work cycles the RAM will perform on the input n (i.e. the uniform time complexity of the RAM).

It is recommended to use the **iterative** algorithm for the Fibonacci numbers as a basis for the RAM program.

Solution:

Starting configuration of the RAM:

Reg.	**value at start**	**will contain ... during program**
R0	n	loop counter (down to 0)
R1	1	F_{j-1}
R2	1	F_j
R3	1	F_{j+1}
R4	1	decrement 1

RAM program:

0	R0 ← R0 - R4	decrement R0 (loop counter)
1	IF R0 = 0 GOTO 6	end loop if R0 becomes 0
2	R3 ← R1 + R2	$F_{j+1} := F_j + F_{j-1}$
3	R1 ← R2	copy former F_j (in R2) to F_{j-1} (in R1)
4	R2 ← R3	copy former F_{j+1} (in R3) to F_j (in R2)
5	GOTO 0	continue loop
6	R0 ← R3	after loop: copy result to R0
7	STOP	