Fundamental Algorithms | WS 2002/2003
Dr. Michael Bader | 4.11.2002

Bubble-Sort on a RAM

Definition of the Problem

Input: A sequence of integers $a_{i_{\min}}, \ldots, a_{i_{\max}}$

Output: A permutation (reordering) $a'_{i_{\min}}, \ldots, a'_{i_{\max}}$ such that $a'_{i_{\min}} \leq \ldots \leq a'_{i_{\max}}$

Algorithm in Pseudo-Code

BUBBLESORT(A)
  for i **from** $i_{\min}$ **to** $i_{\max}$ **do**
    for j **from** $i_{\max}$ **downto** $i+1$ **do**
      if $A[j] < A[j-1]$ **then** exchange $A[j] \leftrightarrow A[j-1]$

A shall be an array that contains the input sequence: $A[i] = a_i$ for $i = i_{\min}, \ldots, i_{\max}$.

After BUBBLESORT has terminated, A will contain the sequence $a'_{i_{\min}} \leq \ldots \leq a'_{i_{\max}}$, i.e. $A[i] = a'_i, i = i_{\min}, \ldots, i_{\max}$.

Starting configuration

We presume that $i_{\min} > 6$. Thus, we obtain 6 unused registers that we will require for our program:

Reg.	value at start	will contain ... during program
R0	$i_{\min}$ (lowest array index)	i
R1	$i_{\max}$ (largest array index)	$i_{\max}$
R2	0	j
R3	1	decrement 1
R4	0	intermediate results
R5	0	intermediate results
R6	0	intermediate results
R$i_{\min}$ ⋮ R$i_{\max}$	$A[i_{\min}] = a_{i_{\min}}$ ⋮ $A[i_{\max}] = a_{i_{\max}}$	array elements $A[i]$ (partially sorted)

Program on the RAM

0	R4 $\leftarrow$ R0 - R1	start of i-loop; compare i and $i_{\max}$
1	IF R4 $>$ 0 GOTO 17	if $i > i_{\max}$ goto 17 (i-loop finished)
2	R2 $\leftarrow$ R1	$j := n$ (start of j-loop)
3	R4 $\leftarrow$ R2 - R0	compare j and i
4	IF R4 = 0 GOTO 15	if $j \leq i$ goto 15 (j-loop finished)
5	R4 $\leftarrow$ RR2	R4 := $A[j]$
6	R5 $\leftarrow$ R2 - R3	R5 := $j - 1$
7	R6 $\leftarrow$ RR5	R6 := $A[j-1]$
8	R4 $\leftarrow$ R6 - R4	compare $A[j-1]$ (in R6) and $A[j]$ (in R4)
9	IF R4 = 0 GOTO 13	if $A[j-1] \leq A[j]$ goto 13 (continue j-loop)
10	R4 $\leftarrow$ RR2	
11	RR2 $\leftarrow$ R6	swap elements $A[j]$ (was in RR2) and $A[j-1]$ (was in RR5)
12	RR5 $\leftarrow$ R4	
13	R2 $\leftarrow$ R2 - R3	decrease loop counter j (by R3=1)
14	GOTO 3	continue j-loop
15	R0 $\leftarrow$ R0 + R3	increase loop counter i (by R3=1)
16	GOTO 0	continue i-loop
17	STOP	

Uniform Time Complexity

For an input x of uniform size n, $|x|_{uni} = n$, the uniform time complexity of our RAM will be bounded by

$$\frac{9}{2}n(n-1) + 7n + 2 \leq T_M^{uni}(x) \leq \frac{12}{2}n(n-1) + 7n + 2,$$

because

- the i-loop is repeated exactly n times; in each repetition, the j-loop is repeated $n - i$ times; hence, the total number of repetitions of the j-loop is

$$\sum_{i=1}^{n}(n-i) = \sum_{k=0}^{n-1} k = \frac{1}{2}n(n-1)$$

- one iteration of the j-loop takes at least 9 work cycles (statements in light grey), plus 3 work cycles (statements in dark grey) if array elements have to be exchanged;
- each i-loop takes 5 work cycles resulting from the 5 statements outside of the j-loop
- the first two loop statements (0 and 1 in the i-loop, 3 and 4 in the j-loop) are executed one more time after the respective loop is finished; this leads to $2n+2$ additional work cycles.

Therefore,

$$T_M^{uni}(n) \in \Theta(n^2)$$