

# Data Structures and Algorithms

## Exercise 2

Markus Innerebner, Romans Kasperovics  
dsa(a]inf.unibz.it

Free University of Bozen - Bolzano, Italy

Thursday, March 12rd, 2009

## Solutions 02

### Task 1

	1 sec	1 min	1 hour	1 day	1 month	1 year
$\sqrt{n}$	1e+6	1.296e+6	1.296e+13	7.465e+16	6.7185e+18	8.7071e+23
$n$	1,000	60,000	3.6e+6	8.64e+7	2.592e+9	9.3312e+11
$n \ln n$	190	6,799	286,499	5,562,845	138,278,494	1.4929e+9
$n^2$	31.6227	244.9489	1,897.3665	9.2951e+7	5.091e+8	9.660e+9
$n^3$	10	39.1486	153.2616	442.0837	1,373.65709	9,771.9034
$2^n$	9.9658	15.8727	21.7796	26.3645	31.2714	39.7633
$n!$	6	8	9	11	12	14

## Solutions 02

### Task 1

For each function  $f(n)$  and time  $t$  in the following table, determine the largest size  $n$  of a problem that can be solved in time  $t$ , assuming that the algorithm to solve the problem takes  $f(n)$  milliseconds.

	$\lg n$
1 sec	1.0715e+301
1 min	6.3058e+18061
1 hour	9.3064e+2167415
1 day	4.2205e+26008991
1 month	5.7682e+780269748
1 year	

## Solutions 02

### Task 2

Order the following functions according to their asymptotic complexity:

#### Functions

- $30 \cdot \log_2 n = \Theta(\log_2 n)$
- $5 \cdot n + n^2 + 1 = \Theta(n^2)$
- $\log_{10}^2 n = \Theta(\log_{10} n)^2$
- $2^n + 5 = \Theta(2^n)$
- $4^{\sqrt{n}} = \Theta(4^{\sqrt{n}})$
- $5^3 \cdot n = \Theta(n)$
- $3 \cdot \log_{10} n = \Theta(\lg n)$
- $(n+1)! = \omega(2^n), o(n^n)$
- $4^{\log_2 n} = (2^2)^{\log_2 n} = (2^{\log_2 n})^2 = \Theta(n^2)$
- $\sqrt{n} = \Theta(\sqrt{n})$

#### Ordering of Functions

- 1  $3 \cdot \log_{10} n$
- 2  $30 \cdot \log_2 n$
- 3  $\log_{10}^2 n$
- 4  $\sqrt{n}$
- 5  $5^3 \cdot n$
- 6  $4^{\log_2 n}, 5 \cdot n + n^2 + 1$
- 7  $3^{\sqrt{n}}$
- 8  $2^n + 5$
- 9  $(n+1)!$

## Solutions 02

### Task 3

Let  $A_1$  be a sorted array of length  $n_1$ . Let  $A_2$  be an unsorted array of length  $n_2$ .  $A_1$  and  $A_2$  store integers and do not contain duplicates.

- 3a) Write an algorithm that prints the integers that are present in both arrays. The asymptotic complexity of your algorithm must be less than  $O(n_1 \cdot n_2)$ . The arrays may not be changed and your solution may not use more than a constant amount of additional memory.
- 3b) Determine the exact running time of your algorithm.
- 3c) Determine a tight asymptotic bound for the runtime of your algorithm.
- 3d) Give at least two examples of special cases of the input data specifying the type of the special case

## Solutions 02

### Task 3b/c

Determine the exact running time of your algorithm.

Instruction	Cost	times
<b>for</b> $i := 1$ <b>to</b> $n_2$ <b>do</b>	$c_1$	$n_2 + 1$
$l := 1$ ; $r := n_1$ ;	$c_2$	$n_2$
<b>do</b>	0	$n_2 \log n_1$
$m := \lfloor (l+r)/2 \rfloor$	$c_3$	$n_2 \log n_1$
<b>if</b> $A_1[m] < A_2[i]$ <b>then</b> $print\ A_1[m]$ ; $r := 0$ ; <b>else if</b> $A_1[m] > A_2[i]$ <b>then</b> $r := m - 1$ <b>else</b> $l := m + 1$	$c_4$	$n_2 \log n_1$
<b>while</b> $(l \leq r)$	$c_5$	$n_2 \log n_1 + 1$

$$T(n_1, n_2) = c_1 * (n_2 + 1) + c_2 * n_2 + c_3 * n_2 \log n_1 + c_4 * n_2 \log n_1 + c_5 * (n_2 \log n_1 + 1) = \Theta(n_2 \log n_1)$$

## Solutions 02

### Task 3a

```
# define n1 8
# define n2 8
int A1[n1] = {1,3, 4,9 ,12,25,27,39}, A2[n2] = {7,1,15,48,22,27, 5,70};
int main() {
    int i,q,l,r,m;
    for(i=0;i<n2;i++){
        l=0; r=n1-1;
        do {
            m = floor((l+r)/2);
            if(A1[m]==A2[i]) {
                printf("%2d\n", A1[m]);
                break;
            }
            else if(A1[m]>A2[i]) r = m-1;
            else l=m+1;
        } while (l<=r);
    }
    return 0;
}
```

## Solutions 02

### Task 3c

Determine a tight asymptotic bound for the runtime of your algorithm.

#### Asymptotic Complexity

$$T_{\text{best}}(n) = \Theta(2 * n - 1) \text{ (assume } n_1 = n_2)$$

$$T_{\text{worst}}(n) = \Theta(n_2 \log n_1)$$

### Special cases

	Special case	$A_1$	$A_2$
1	Worst case	(1, 2, 3, 4, 5, 6, 7)	(0, 9, 15, 12, 11, 10, 13)
2	Best case	(1, 2, 3, 4, 5, 6, 7)	(4, 2, 6, 1, 3, 5, 7)