4 Amortized complexity, time complexity of algorithms

4.1 Solving homework from last tutorial

a) A function $T(n)$ is given on natural number by the following recurrence. Find it asymptotic behaviour. Justify you answer (i.e. either state the theorem you have used, or solve directly).

1. $T(n) = 2\,T(\frac{n}{3}) + 1$, $T(1) = 1$.
2. $T(n) = 4\,T(\frac{n}{3}) + n^{\frac{3}{2}} \log_2 n$, $T(1) = 1$.

b) A function $T(n)$ is given on natural number by the following recurrence. Find it asymptotic behaviour. Justify you answer (i.e. either state the theorem you have used, or solve directly).

1. $T(n) = T(n-1) + c^n$, $T(1) = 1$, a kde $c > 1$ je konstanta z $\mathbb{R}^+$.

4.2 Find the amortized complexity of function $insert(x)$. The function $insert(x)$ adds an element to an array by the following way: We begin with one element array, when the array is filled it is doubled, the origin array is copied to new one, and x is inserted.

4.3 Multiplication of long numbers: Given two binary words a, b; we want to calculate their product. Find the time complexity of the following two algorithms:

Algorithm 1 (naive): Assume that addition of two numbers takes a constant time.

```
mezivysledek := 0
 for  (i=1; i< a+1;i++) do
    mezivysledek := mezivysledek + b
return mezivysledek
```

Algorithm 2 (reccursive): An arbitrary number with $2N$ digits can be written as $2^N\,A + B$ where A and B are numbers with N digits. A product of such numbers is then

$$(2^N \cdot A + B) \cdot (2^N \cdot C + D) = (2^{2N} \cdot AC + 2^N\,(AD + BC) + BD).$$

Assume that additions can be performed in constant time as well as multiplication by a power of 2. Numbers with N digits are multiplied recursively by the same algorithm.

4.4 Decide what is the size for time and space complexity if an instance is:

1. a sequence of numbers $a_1, a_2, \ldots a_n$,
2. a graph with n vertices and m edges,
3. a matrix of order $n \times m$,
4. a number x which is the instance of a problem (e.g. test of primarility).

4.5 Given two numbers $a, b > 0$ and a pseudo code

```
while a > 0 do
   if a < b then
       (a, b) := (2a, b − a)
   else
       (a, b) = (a − b, 2b)
end while
return b
```

Is this a pseudo code of an algorithm (which always terminates)? If yes, calculate its time complexity.

4.6 Given the following algorithm

$i = N$;
`while` $(i > 0)$ `do` $\{$
  $j = 0$;
  `while` $(j^2 < i)$ `do` $\{$`write`$(\star)$; $j++$ $\}$
  $i = i - 2$ $\}$

Calculate the asymptotic time complexity, i.e. find the simplest function $f(N)$ such that $\Theta(f(N))$ is the asymptotic growth of the number of symbols $\star$. (A function is not sufficient, you have to justify why it is correct.)

Problem to be solved before next tutorial

4.7 Given the following algorithm

$i = N$
`while` $(i > 0)$ `do` $\{$
  $j = 0$;
  $i := \lfloor i/3 \rfloor$;
  `while` $(j < i)$ `do` `write`$(\star)$; $j := j + 1$ $\}$

Calculate the asymptotic time complexity, i.e. find the simplest function $f(N)$ such that $\Theta(f(N))$ is the asymptotic growth of the number of symbols $\star$. (A function is not sufficient, you have to justify why it is correct.)