Algortihmic complexity

How do we define what degree of complexity an algortihm has?

- Time complexity: running time of the algorithm
- Space complexity: how much memory is it using
- Size of input data: number of items, number of bits

Random access machine

- Arithmetic: addition, subtraction, multiplication
- Control: branch, subroutine call, return
- Data movement: load, store, copy
- data types: char, integer and floating point (decimal)

Analysis of insertion sort

```
for i=2 to n do
    j = i-1;
    t = A[i];
    while j>0 · t<A[j] do
        A[j+1] = A[j];
        j = j-1;
    A[j+1] = t;
```

Execution of a line costs time. Each line has a cost of **c**. This cost is multiplied by as many elements have to be processed by the algorithm. Inside loops, we use the notation $\mathbf{t}_i$. The sum of all multiplications between c, n and t_i gives as the exact running time of insertion sort.

Best, worst and average case

Considering insertion sort, our best case option is that the array is already sorted. $t_i = 1$

$$T(n) = c_1 n + (c_2 + c_3 + c_4 + c_7)(n - 1) = an + b$$

Correctness

Asymptotic complexity

Special case analysis