Session 41: Complexity Analyses

- Linear Search
- Binary Search
- Bubble Sort
- Insertion Sort

How much detail is needed?

procedure *max*(a_1, a_2, …., a_n: integers)

max := a_1

for *i* := 2 to *n*

if *max* < a_i then *max* := a_i

return *max*

Loop is executed c1(n) = n - 1 times
- 2 comparisons per loop
- 1 extra comparison at the end

Total cost 2 * c1(n) + 1

c1(n) is $\Theta(n)$

2 * c1(n) is also $\Theta(n)$

2 * c1(n) + 1 is also $\Theta(n)$

Therefore it does not matter whether
- we consider multiple operations within a loop
- constant number of extra operations outside the loop

We just count the loops !

Worst Case Complexity of Linear Search

procedure *linearsearch*(x: integer,
$a_1, a_2, \ldots, a_n$: distinct integers)

$i := 1$

while ($i \leq n$ and $x \neq a_i$)
$i := i + 1$

loop is executed $n+1$ times

if $i \leq n$ **then** *location* := i
else *location* := 0

return *location*

$f(n) = c_1(n+1) + c_2$ is $\Theta(n)$

Worst Case Complexity of Binary Search

procedure binary search(x: integer, $a_1, a_2, \ldots, a_n$: increasing integers)

$i := 1$

$j := n$

while $i < j$

$m := \lfloor (i + j)/2 \rfloor$

if $x > a_m$ **then** $i :=$ m + 1

else $j :=$ m

if $x = a_i$ **then** *location* := i

else *location* := 0

return *location*

Assume $n = 2^k$

How often is the loop executed?

In each loop the number of elements considered is $\frac{1}{2}$ of the previous loop:

2^k
2^{k-1}
2^{k-2}
$\vdots$
1

$\Big\}$ $k + 1 = \log_2(n) + 1$ loops

Therefore the cost is $c_1 \log_2(n) + c_2$

and the complexity is $\Theta(\log n)$

Worst Case Complexity of Bubble Sort

procedure *bubblesort*(a_1,…,a_n: real numbers with $n \geq 2$)

for i := 1 to $n - 1$

for j := 1 to $n - i$

if $a_j > a_{j+1}$

then interchange a_j and a_{j+1}

Outer loop is executed n-1 times

Inner loop:

$i = 1$ $\quad$ $n-1$ executions

$i = 2$ $\quad$ $n-2$

$\vdots$

$i = n-1$ $\quad$ 1

Inner loop is executed

$(n-1) + (n-2) + \dots + 1$ times

This sum is $\frac{n(n-1)}{2}$, a polynomial of degree 2

Therefore complexity is $\Theta(n^2)$

Worst Case Complexity of Insertion Sort

procedure *insertion sort*(a_1,…,a_n:
real numbers with $n \geq 2$)

for j := 2 to n

i := 1

while $a_j > a_i$ **and** $i < j$

$i := i + 1$

$m := a_j$

for k := 0 to $j - i - 1$

$a_{j-k} := a_{j-k-1}$

$a_i := m$

Outer loop is executed $n-1$ times

$j = 2 : 1$
$3 : 2$
$\vdots$
$n : n-1$

} This loop is executed at most $(n-1) + \dots + 1 = \frac{n(n-1)}{2}$ times

$j = 2 : 1$ (if $i = 1$)
$3 : 2$
$\vdots$
$n : n-1$

} This loop is executed at most $(n-1) + \dots + 1 = \frac{n(n-1)}{2}$ times

cost is a polynomial of degree 2
Therefore complexity is $\Theta(n^2)$

Summary

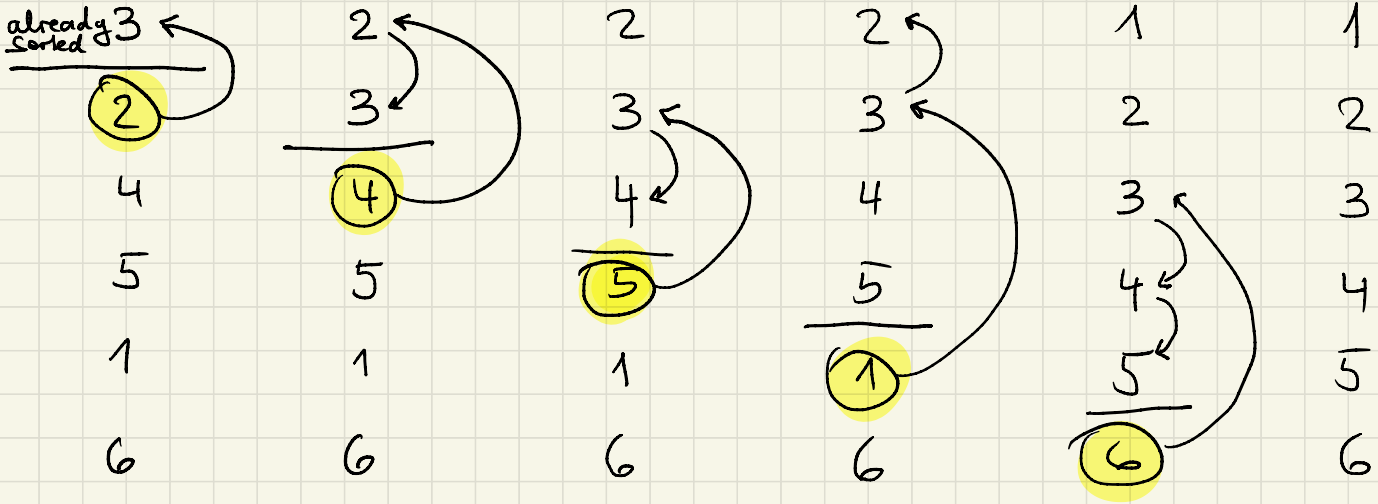
Worst case complexities

- Linear Search: $\Theta(n)$
- Binary Search: $\Theta(\log(n))$
- Bubble Sort: $\Theta(n^2)$
- Insertion Sort: $\Theta(n^2)$

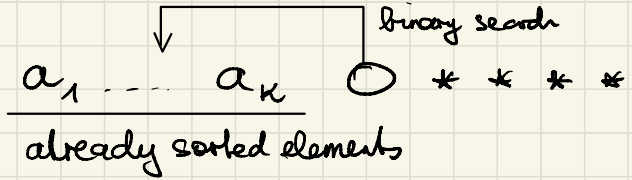
Binary Insertion Sort: uses binary search instead of linear search when sorting (improvement of linear insertion sort)

Note: Since the elements that are already sorted are sorted, we can perform a binary search.

Example: Sorting 3, 2, 4, 5, 1, 6



Complexity of binary insertion sort



Number of comparisons: $\lceil \log_2 1 \rceil + \lceil \log_2 2 \rceil + \dots + \lceil \log_2 (n-1) \rceil \leq n \log_2 n$

Note: linear insertion sort needed $O(n^2)$ comparisons

Problem: we have to count also assignments

In the worst case we have to insert the next element to the first position

Then the following assignments need to be performed

$$a_{k+1} := a_k;\ a_k := a_{k-1};\ \dots,\ a_1 := b$$

So the total number of assignments is in the worst case $O(n^2)$

Computing the value of a polynomial

Given $c, a_0, \dots, a_n$. Compute the value $a_n c^n + a_{n-1} c^{n-1} + \dots + a_0$

Naive approach:

compute $a_k \cdot c^k$, computing c^k requires $k-1$ multiplications

then sum up

total: $n + \dots + 1 + 0 = \frac{n(n+1)}{2}$ multiplications and n additions

Horner's method: with n multiplications and additions

```
y := a_n
for i = n-1 do 0
    y = y * c + a_i
```

a_n

$a_n \cdot c + a_{n-1}$

$(a_n \cdot c + a_{n-1}) \cdot c + a_{n-2} = a_n c^2 + a_{n-1} c + a_{n-2}$