September 17, 2020

Sorting algorithms

We consider algorithms consisting of a sequence of statements like this:

IF $a_i < a_j$ THEN "Rearrange $(a_1, \ldots, a_n)$"

comparison of two elements

swap two elements usually

Comparison-based sorting algorithms

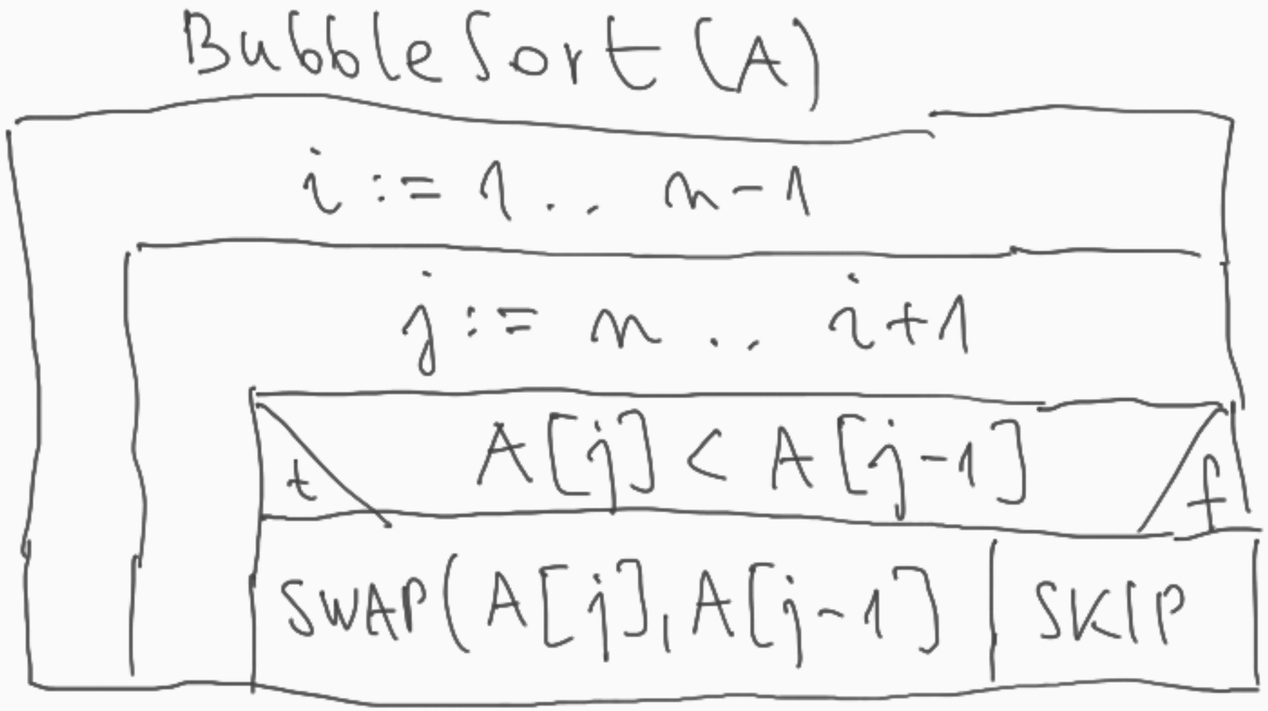
Maybe the most natural approach is this: swap the pairs of consecutive elements with wrong order

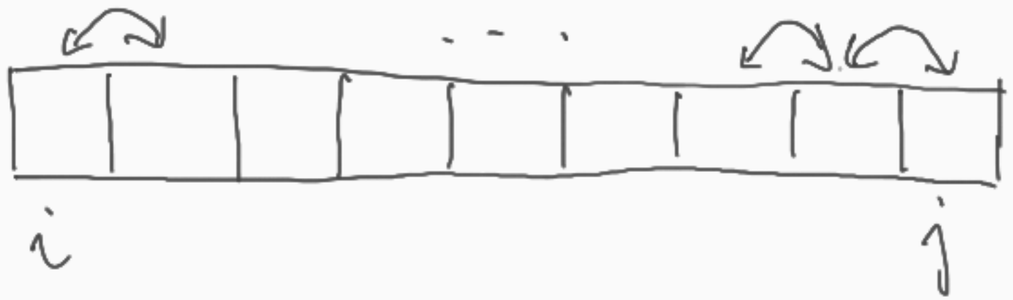
A systematic way to do this

→ bubble sort

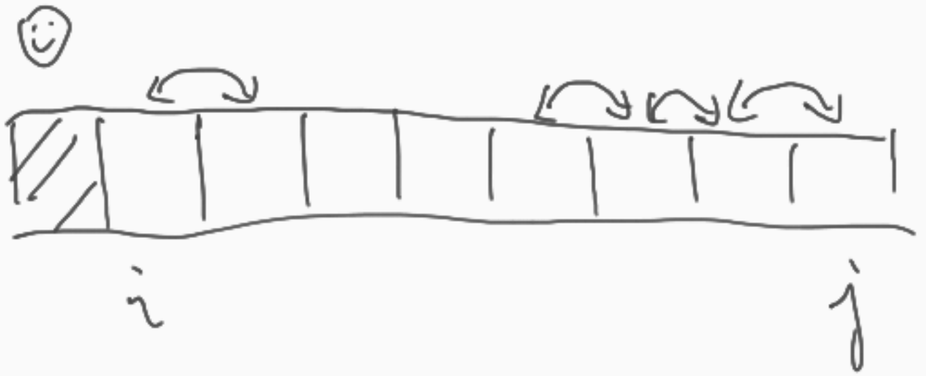
```
BubbleSort (A)                    A[1:n]
for i=1 to n-1 do
    for j=n downto i+1 do
        if A[j] < A[j-1] then SWAP(A[j], A[j-1])
```

```
x := A[j]
A[j] := A[j-1]
A[j-1] := x
```





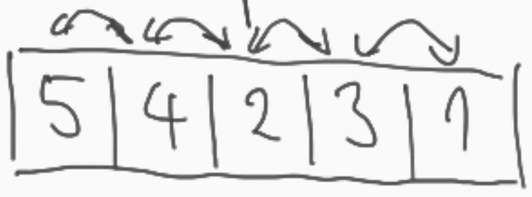
1st phase the smallest element moves to the first position



2nd phase the smallest element in A[2:n] moves to A[2]

the second smallest element in A[1:n]

Example



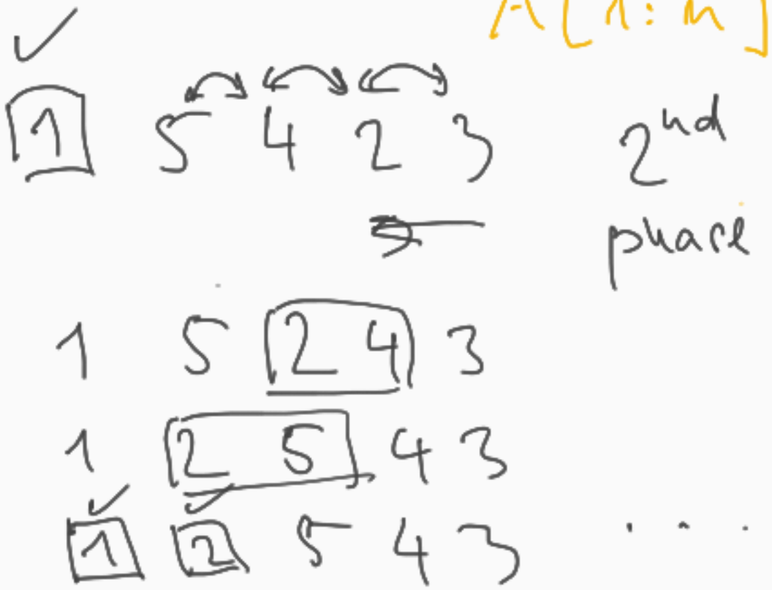
1st phase

5 4 2 1 3

5 4 1 2 3

5 1 4 2 3

1 5 4 2 3



How many comparisions are made during the algorithm?

In general

How many? — When?

- in the worst case ☹
- in the best case ☺
- in average usually hard

How many? - As a function of what?

- the function of the size of the input

How many comparisions are made as a function of the elements during the algorithm in the worst case?

Look at the bubble sort

1st phase → $n-1$ comp's

2nd phase → $n-2$ comp's

⋮

$(n-1)^{th}$ phase → 1 comp's [A[n] and A[n-1])

$$1+2+\ldots+(n-2)+(n-1) = \frac{n(n-1)}{2}$$

this is the exact number for any input with n elements

What can we say about this $\frac{n(n-1)}{2}$?

if $n \sim 1000$, then the alg. is quick

if $n \sim 10^6, 10^9, 10^{12}, \ldots$, then the alg. becomes slower and slower

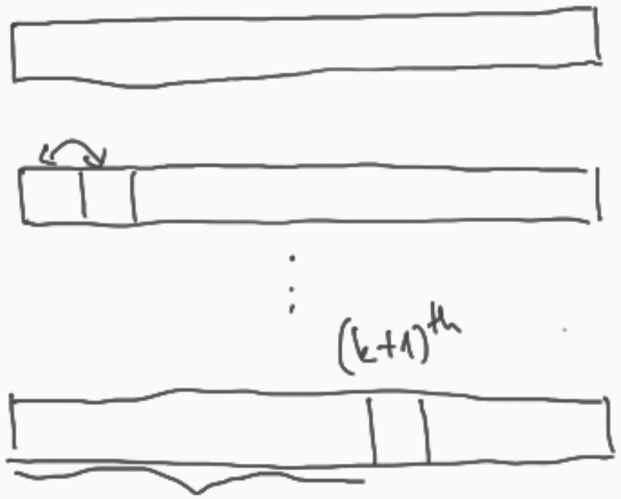
Nevertheless the algorithm is considered as an efficient alg.

theoretically

This is not the only sorting algorithm

Insertion sort

Basic idea:

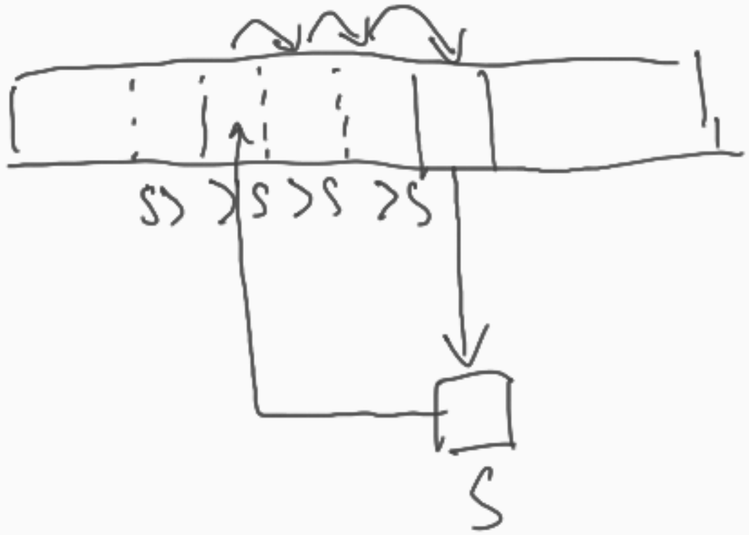


assume that this part is sorted

A[1:n]

1st phase

kth phase



```
InsertionSort (A)
for k = 2 to n do
    S := A[k]
    i := k-1
    while i>0 AND A[i]>S do
        A[i+1] := A[i]
        i := i-1
    A[i+1] := S
```

How many comparisons are made?

Here the best case and the worst case are different

best case: increasingly sorted array

→ $n-1$

worst case: decreasingly sorted array

→ $1 + 2 + 3 + \ldots + n-1 = \frac{n(n-1)}{2}$

5 [2] 3 1 — 1st phase

2 5 [3] 1 — 2nd phase

2 3 5 [1] — 3rd phase

1 2 3 5