Insertion sort

1. Introduction
2. Algorithm presentation
3. Operation counts: best, worst, average

1. Introduction

Insertion sort is a sorting algorithm.

Sorting: placing elements in order (smallest to largest or largest to smallest)

Assumption: for two elements x, y, only three possible cases: (i) x < y, (ii) x = y or (iii) x > y

Insertion-sort:

• can be used to sort items/elements in an array or list

• an example of decrease and conquer

• easy to understand/code

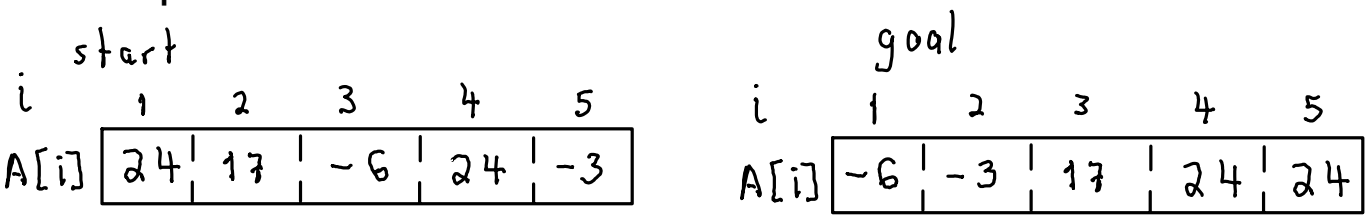
• one of many different sorting algorithms

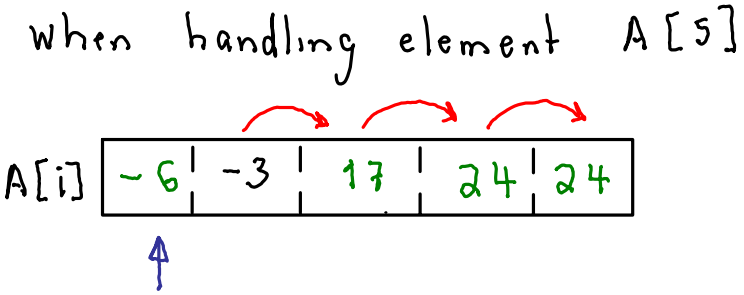
2. Algorithm presentation

Description level

1. We start with array $A[1..n]$ of numbers in no particular order. We want rearrange the array so that the numbers are in order from smallest to largest.
2. We handle one number at a time. Let $A[j]$ be number we are handling. All numbers before $A[j]$ are sorted. When handling $A[j]$, we look for correct position k, $1 \leq k \leq j$ where $A[j]$ belongs. When we find k, we shift all elements in $A[k..j-1]$ right one position and put $A[j]$ into position k.
3. After handling $A[j]$, we handle $A[j+1]$. We repeat until all numbers have been handled.

Example



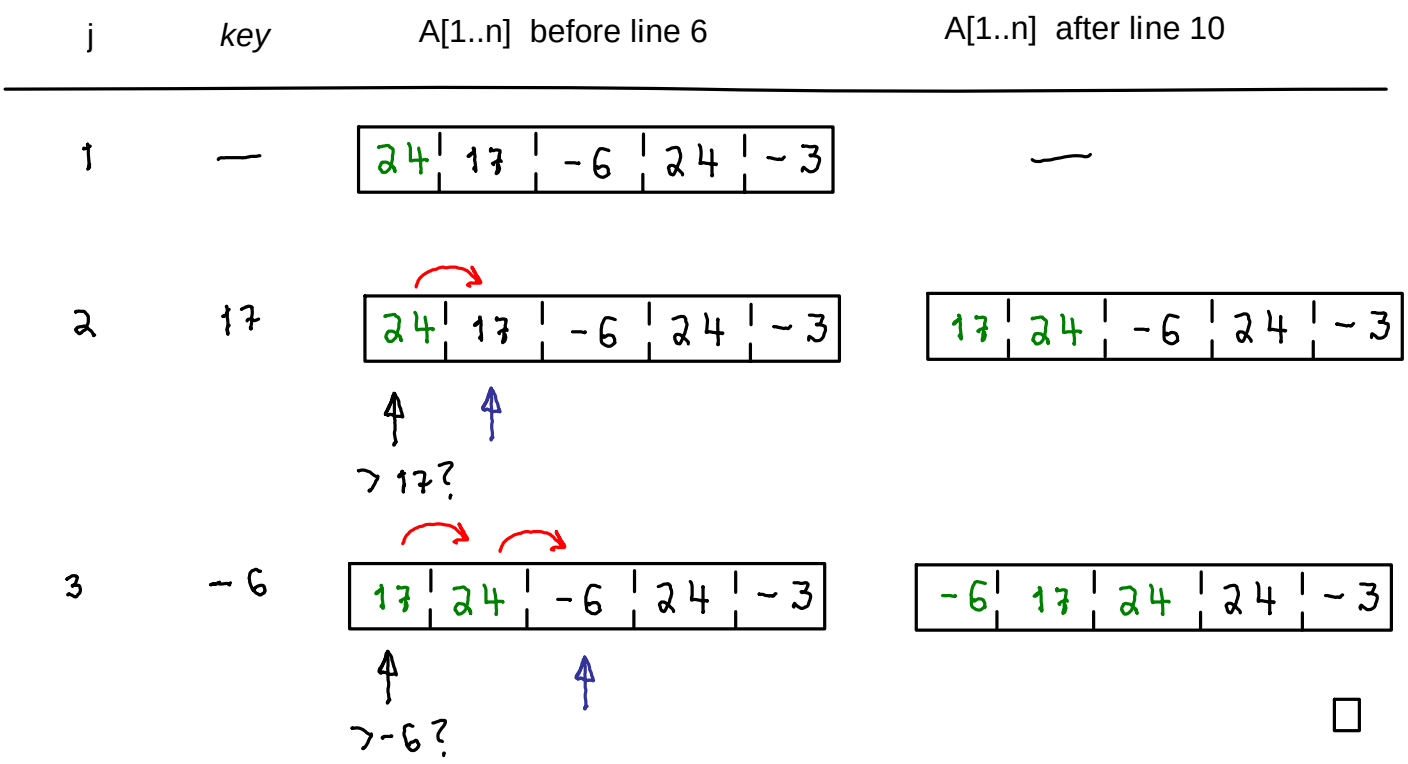


□

Pseudocode

```
1   INSERTSORT(A)
2   input: number array A output: sorted array A
3   /* The numbers in input A[1..n] may be in any order. On output the
4   numbers in A are sorted from smallest to largest. */
5   for j from 2 to A.length
6     key = A[j], k = j
7     while k ≥ 2 and A[k − 1] > key
8       A[k] = A[k − 1], k = k − 1
9     end
10    A[k] = key
11  end
```

Example



3. Operation counts: best, worst, average

Pseudocode

```
1   INSERTSORT(A)
2   input: number array A output: sorted array A
3   /* The numbers in input A[1..n] may be in any order. On output the
4   numbers in A are sorted from smallest to largest. */
5   for j from 2 to A.length
6     key = A[j], k = j
7     while k ≥ 2 and A[k − 1] > key
8       A[k] = A[k − 1], k = k − 1
9     end
10    A[k] = key
11  end
```

How many times will each line be executed?

• for-loop goes through A.length −1 elements; line 6 (or 5 or 10) executed A.length −1 times (n - 1)

• how many times line 8 executed depends on initial order of A[1..n]

Best

Is it possible we never need to execute line 8? What conditions?

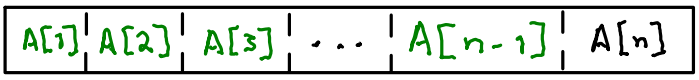
Yes.

If array was already sorted at start.

Worst

Consider j = n (element A[n]). Most number of times line 8 must be executed? (n - 1)

Consider j = n-1 (element A[n-1]). Most number of times line 8 must be executed? (n - 2)



Average

Consider j = n (element A[n]). Best assumption on how many numbers in A[1..(n-1)] are greater than A[n]? (n - 1) / 2

Consider j = n-1 (element A[n-1]). Best assumption on how many numbers in A[1..(n-2)] are greater than A[n-1]? (n - 2) / 2

j	most number of times line 8 executed	average number of times line 8 executed
n	$(n-1)$	$(n-1)/2$
$n-1$	$(n-2)$	$(n-2)/2$
$\vdots$		
i	$(i-1)$	$(i-1)/2$
$\vdots$		
2	1	$1/2$
	$\sum_{r=1}^{n-1} r = \frac{n(n-1)}{2}$	$\frac{1}{2}\sum_{r=1}^{n-1} r = \frac{n(n-1)}{4}$