

Algorithms – Insertion Sort (1C)

Copyright (c) 2017 – 2018 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using LibreOffice and Octave.

Insertion Sort Algorithm

```
procedure insertion sort(a1, ..., an : real numbers with n ≥ 2)
```

```
for j := 2 to n
```

```
    i := 1
```

```
        while aj > ai
```

```
            i := i + 1
```

```
        m := aj
```

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

```
            aj-k = aj-k-1
```

```
        ai := m
```

{a₁, ..., a_n is in increasing order}

Nested loop k – constraints

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

```
    aj-k = aj-k-1
```

$$j - i - 1 \geq 0 \quad j \geq i + 1 \quad i \leq j - 1 \quad i < j$$

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

($k=0$)

$$a_{j-0} = a_{j-0-1}$$

($k=1$)

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

($k=2$)

$$a_{j-2} = a_{j-2-1}$$

$$\vdots = \vdots$$

($k=j-i-1$)

$$a_{j-(j-i-1)} = a_{j-(j-i-1)-1}$$



$$\begin{array}{l} a_j = a_{j-1} \\ a_{j-1} = a_{j-2} \\ a_{j-2} = a_{j-3} \\ \vdots = \vdots \\ a_{i+1} = a_i \end{array}$$

increasing index

Nested loop k – rearranging for understanding

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

```
    aj-k = aj-k-1
```

$$j - i - 1 \geq 0 \quad j \geq i + 1 \quad i \leq j - 1 \quad i < j$$

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

a_j	$=$	a_{j-1}
a_{j-1}	$=$	a_{j-2}
a_{j-2}	$=$	a_{j-3}
\vdots	$=$	\vdots
a_{i+1}	$=$	a_i

increasing index

a_{i+1}	$=$	a_i
\vdots	$=$	\vdots
a_{j-2}	$=$	a_{j-3}
a_{j-1}	$=$	a_{j-2}
a_j	$=$	a_{j-1}

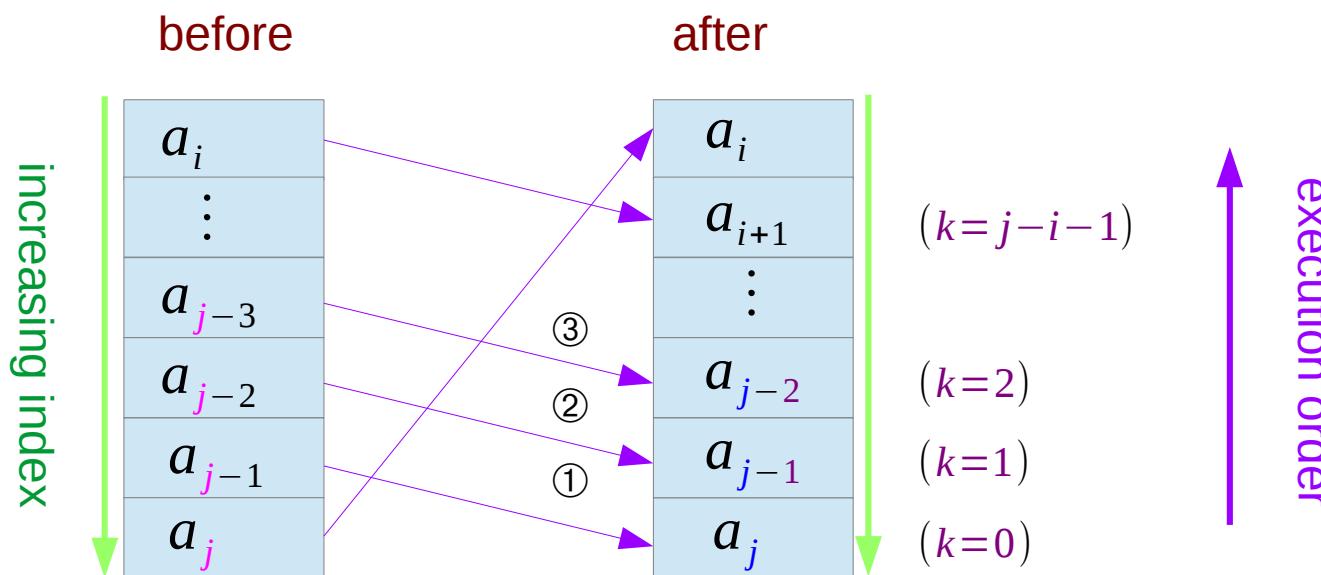
$(k=j-i-1)$
 $(k=2)$
 $(k=1)$
 $(k=0)$

execution order

Nested loop k – data movement

```
m := aj
for k := 0 to j - i - 1
    aj-k = aj-k-1
ai := m
```

$i < j$



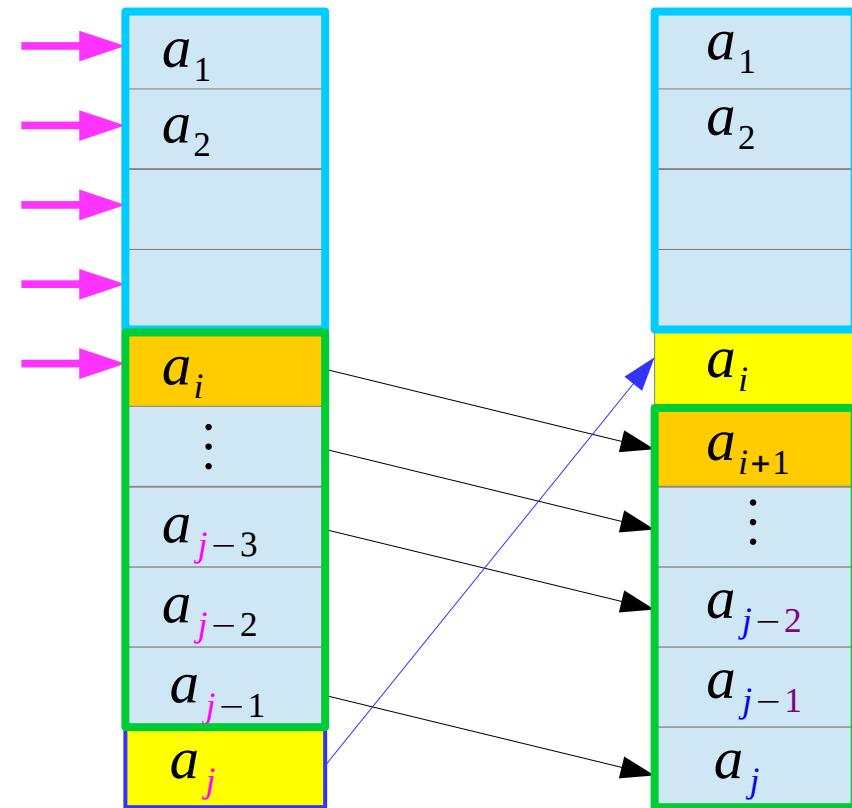
Nested loop i – finding out of order a_i

```
i := 1  
while  $a_j > a_i$   
    i := i + 1
```

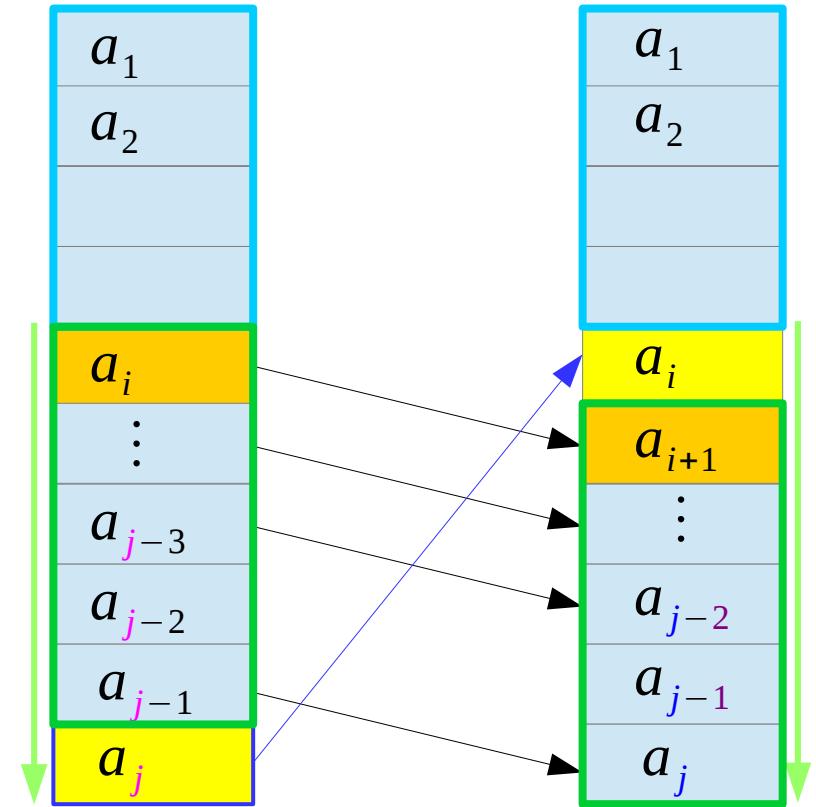
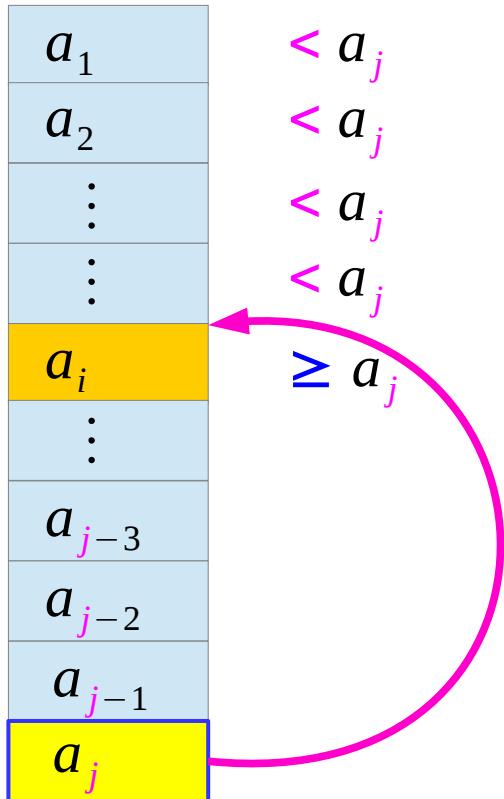
If $a_i < a_j$ increment I

If $a_i \geq a_j$ break the loop

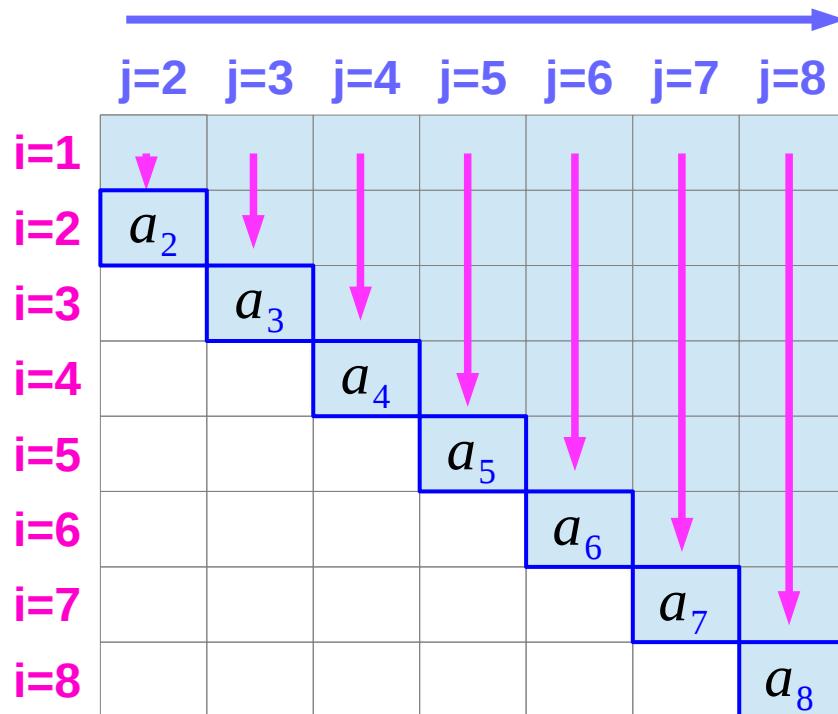
a_i is the 1st one that is greater than a_j



Nested loop i – inserting a_i at the correct position



Nested loop iterations



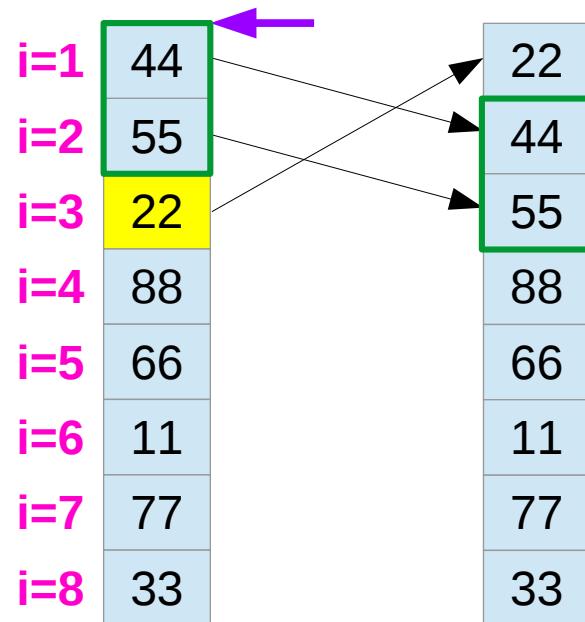
```
for j := 2 to n
    i := 1
    while  $a_j > a_i$ 
        i := i + 1
        m :=  $a_j$ 
    for k := 0 to j - i - 1
         $a_{j-k} = a_{j-k-1}$ 
     $a_i := m$ 
```

Step j=2

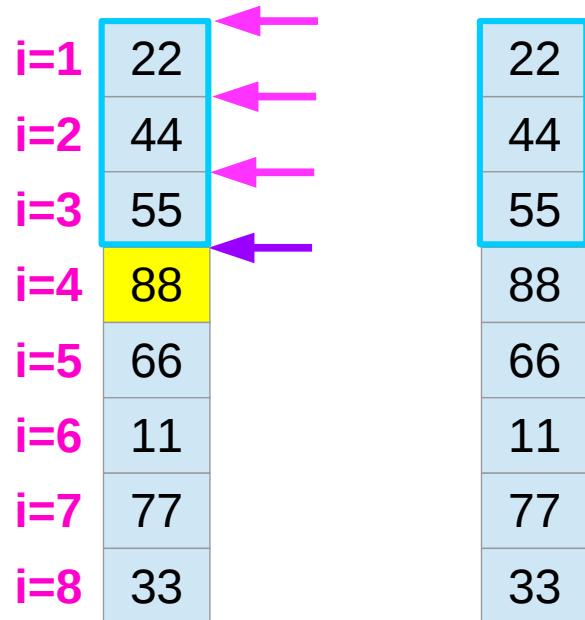
i=1	44
i=2	55
i=3	22
i=4	88
i=5	66
i=6	11
i=7	77
i=8	33

The diagram illustrates the state of an array during the second step of an insertion sort algorithm. The array has 8 elements, indexed from i=1 to i=8. The elements are: 44, 55, 22, 88, 66, 11, 77, 33. The element at index i=2, which is 55, is highlighted in yellow. Two purple arrows point from the right towards the 55 cell, indicating that it is being shifted to its correct position in the sorted portion of the array.

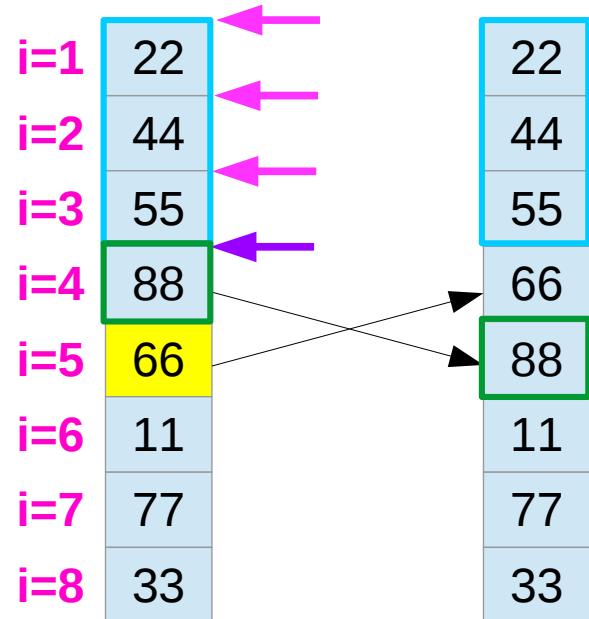
Step j=3



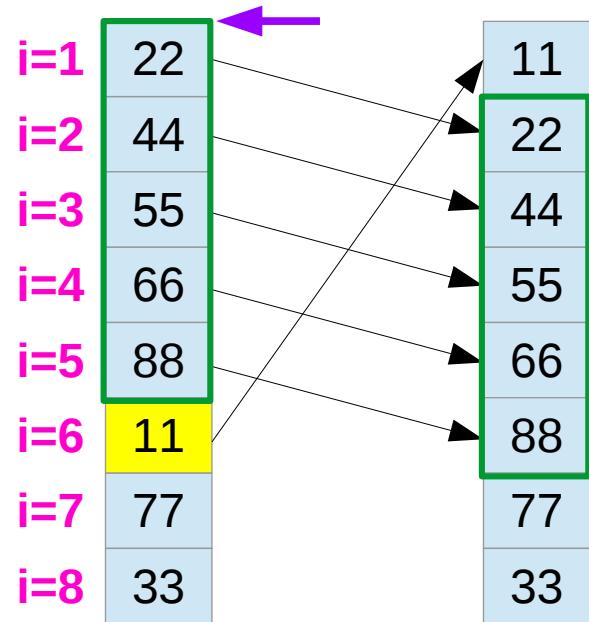
Step j=4



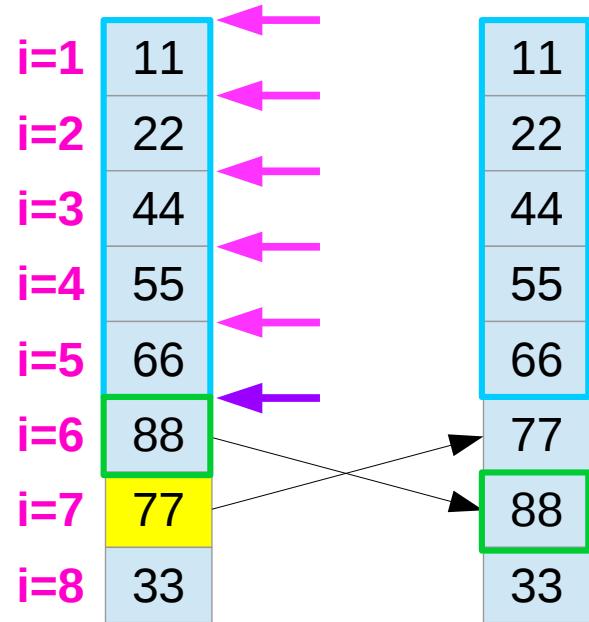
Step j=5



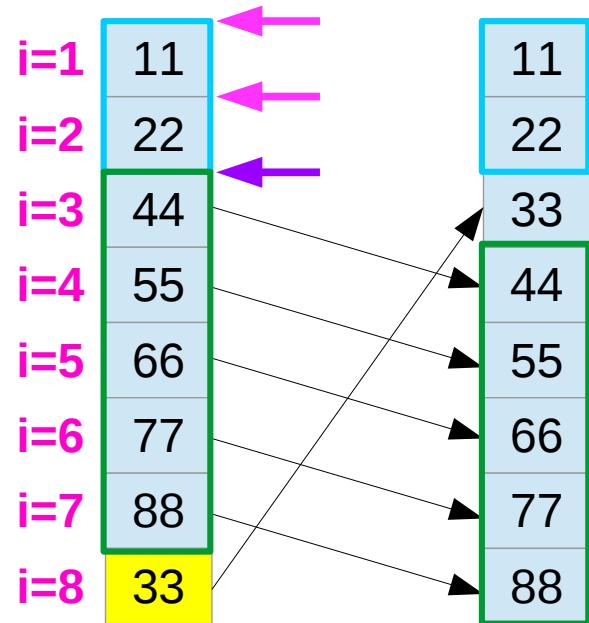
Step j=6



Step j=7



Step j=8



Nested loop iterations

References

- [1] <http://en.wikipedia.org/>
- [2]