DESIGN AND ANALYSIS OF ALGORITHMS
UE19CS251

Shylaja S S
Department of Computer Science & Engineering

DESIGN AND ANALYSIS OF ALGORITHMS

Bubble Sort

Major Slides Content: Anany Levitin

Shylaja S S

Department of Computer Science & Engineering

Bubble Sort

- Compare adjacent elements of the list and exchange them if they are out of order
- By doing it repeatedly, we end up bubbling the largest element to the last position on the list
- The next pass bubbles up the second largest element and so on and after n – 1 passes, the list is sorted
- Pass i ($0 \leq i \leq n-2$) can be represented as follows:

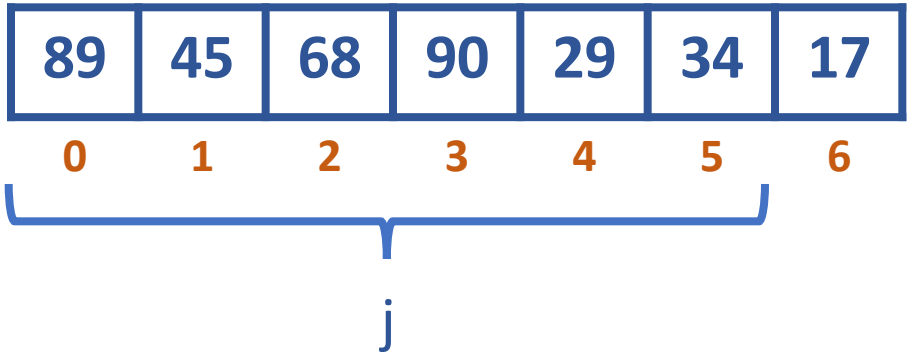
$A[0], A[1], A[2], \ldots, A[j] \overset{?}{\longleftrightarrow} A[j+1], \ldots, A[n-i-1] \mid A[n-i] \leq \ldots \leq A[n-1]$

in their final positions

Bubble Sort

elements in final position

i = 0



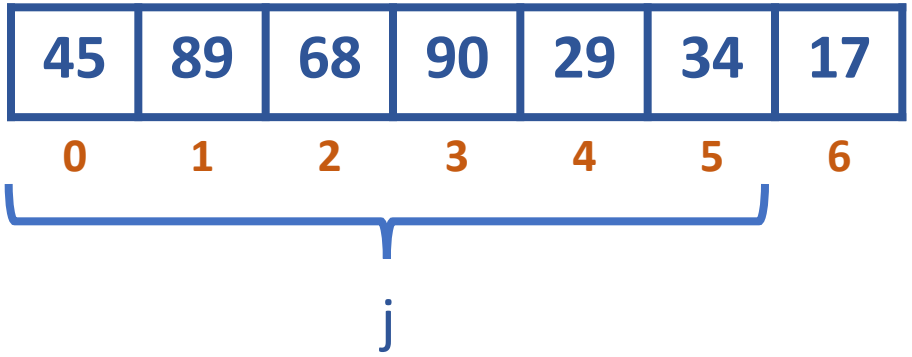
j = 0

45 < 89 ☑ swap

Bubble Sort

elements in final position

i = 0



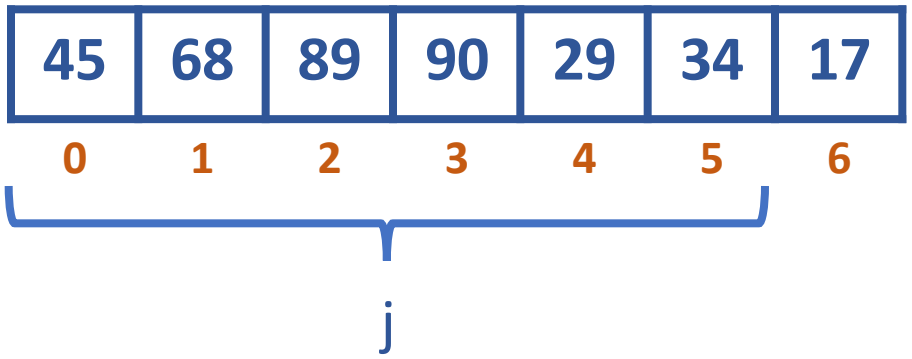
j = 1

68 < 89 ☑ swap

Bubble Sort

elements in final position

i = 0



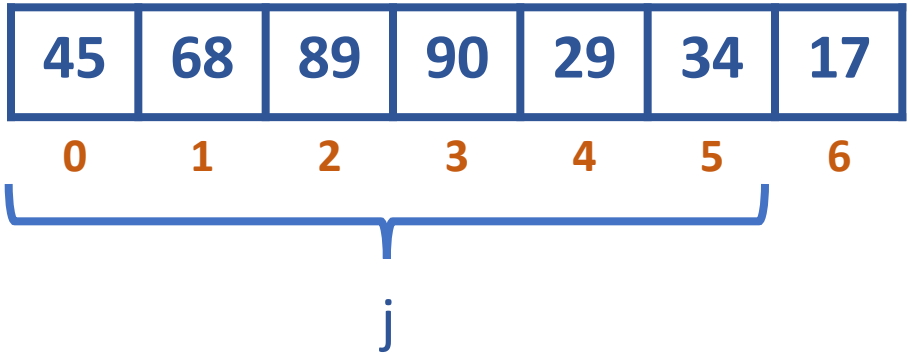
j = 2

90 < 89 ☒ no swap

Bubble Sort

elements in final position

i = 0



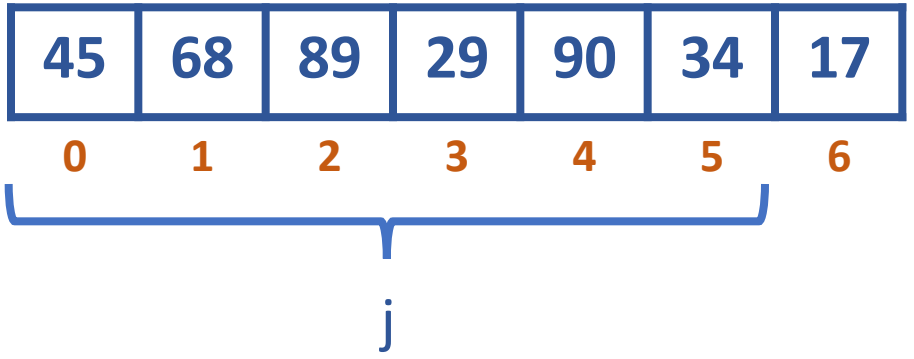
j = 3

29 < 90 ☑ swap

Bubble Sort

elements in final position

i = 0



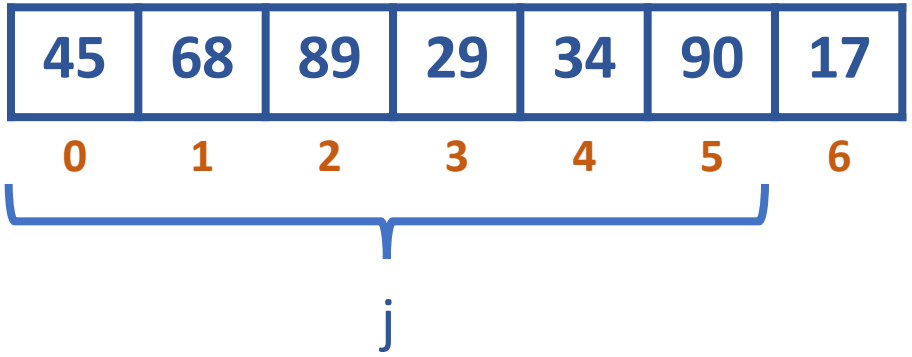
j = 4

34 < 90 ☑ swap

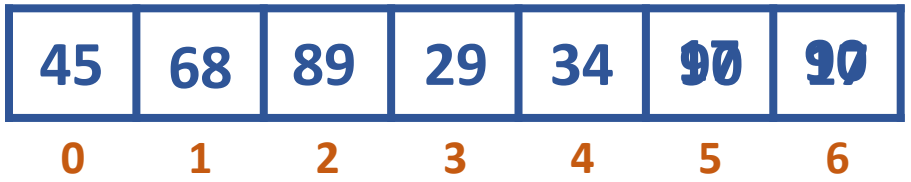
Bubble Sort

elements in final position

i = 0



j = 5

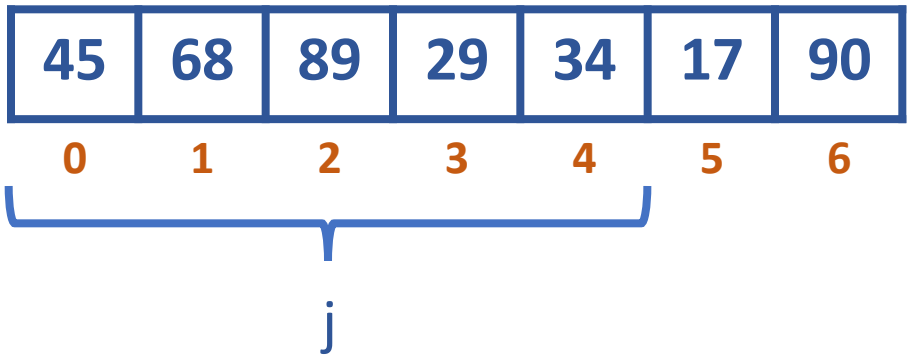


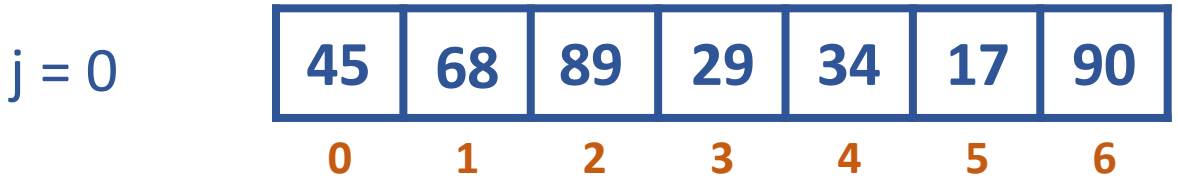
17 < 90 ☑ swap

Bubble Sort

After first iteration

elements in final position



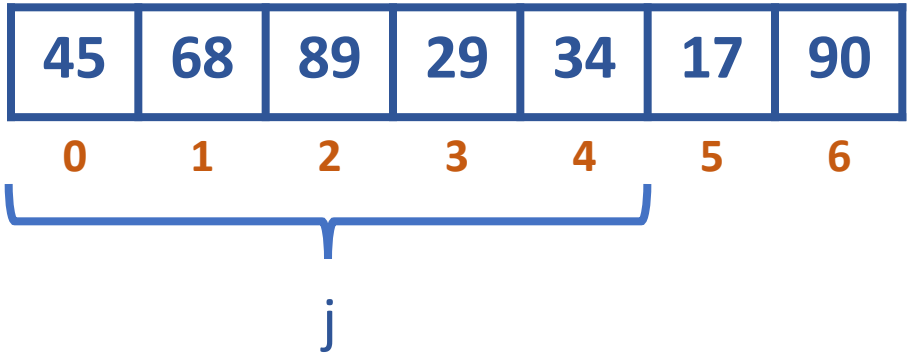


68 < 45 ☒ no swap

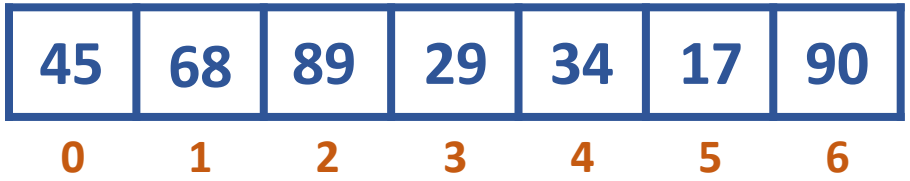
Bubble Sort

elements in final position

i = 1



j = 1

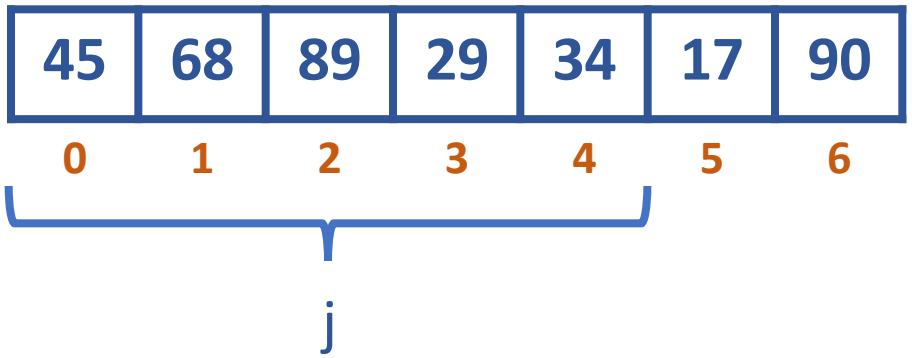


89 < 68 ☒ no swap

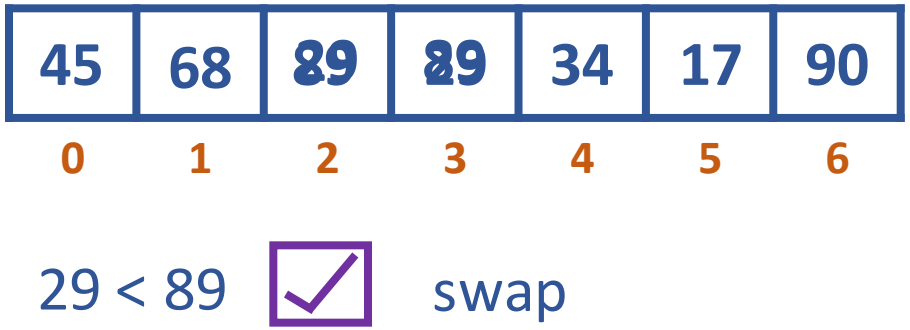
Bubble Sort

elements in final position

i = 1



j = 2

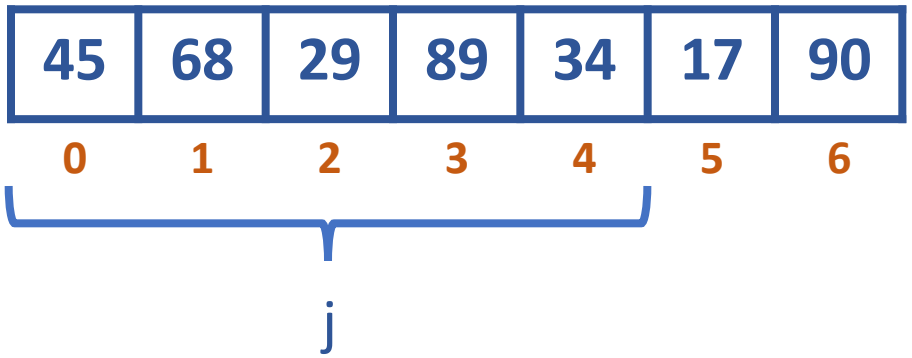


29 < 89 ☑ swap

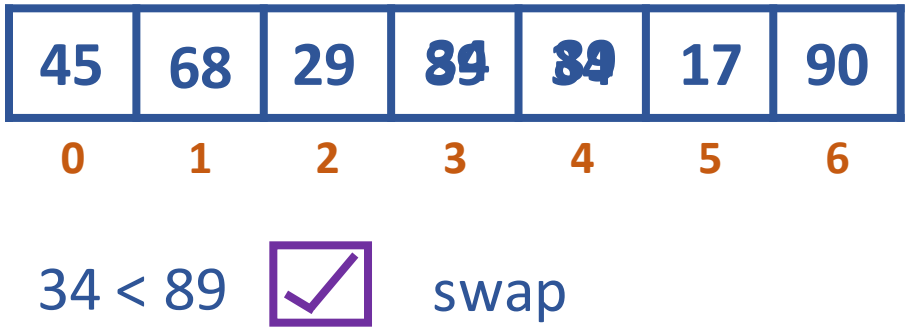
Bubble Sort

elements in final position

i = 1



j = 3

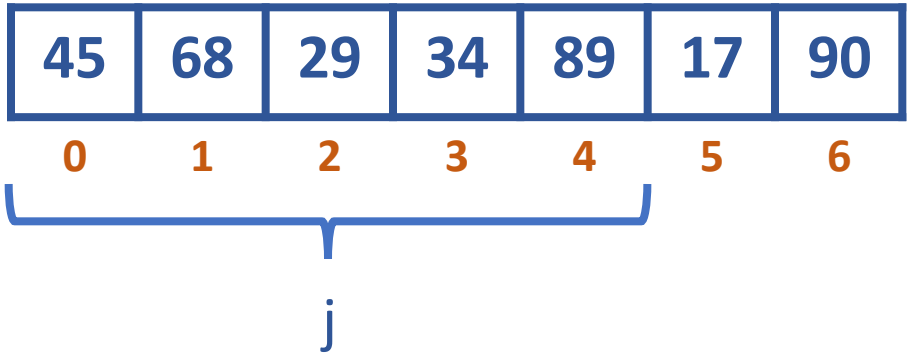


34 < 89 ☑ swap

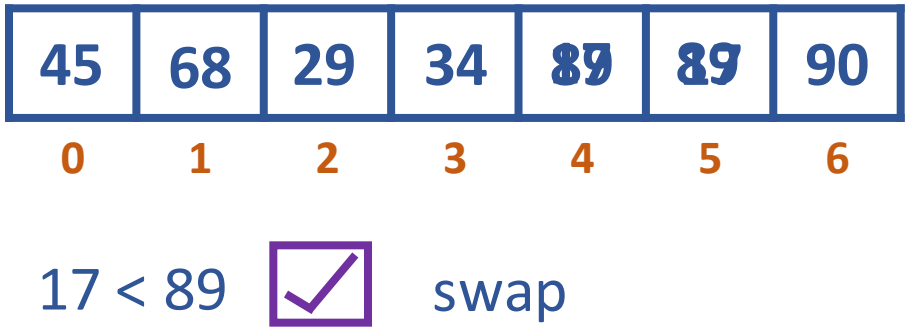
Bubble Sort

elements in final position

i = 1



j = 4

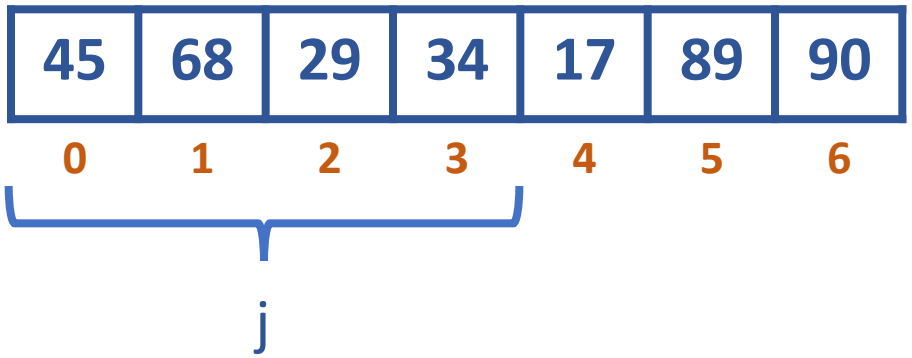


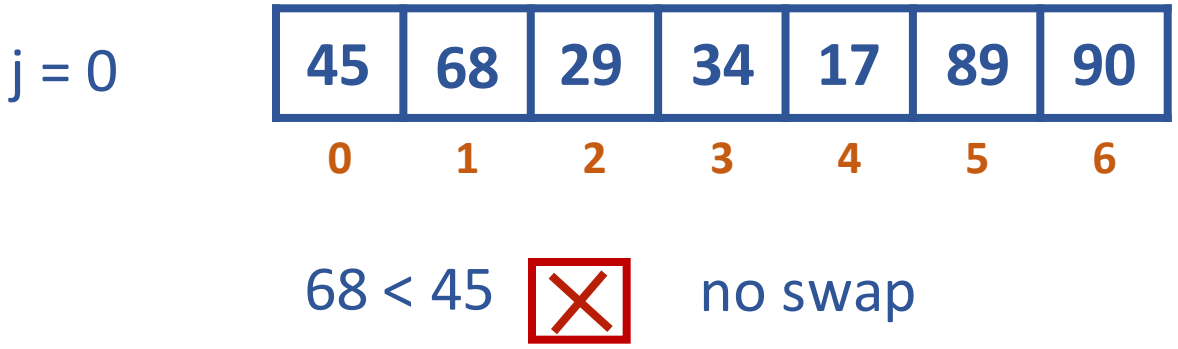
17 < 89 ☑ swap

Bubble Sort

After second iteration

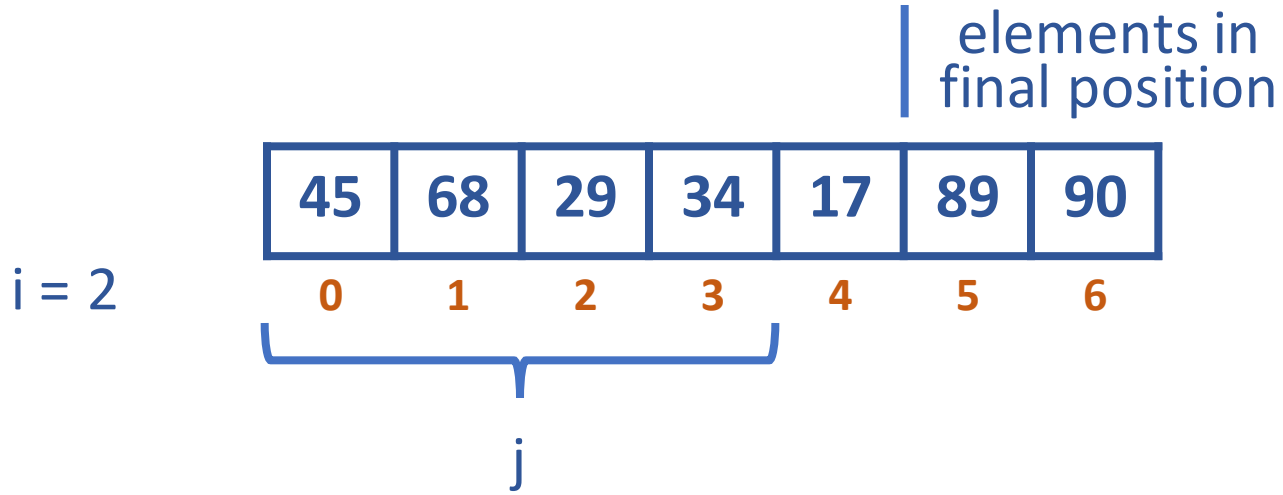
elements in final position

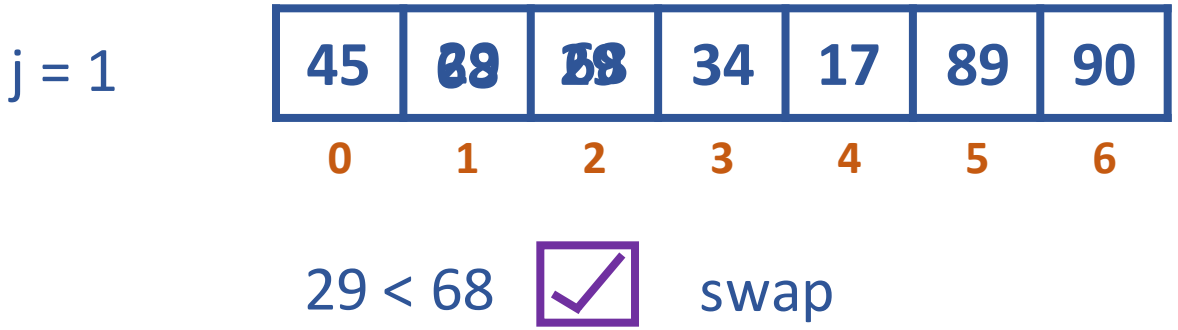




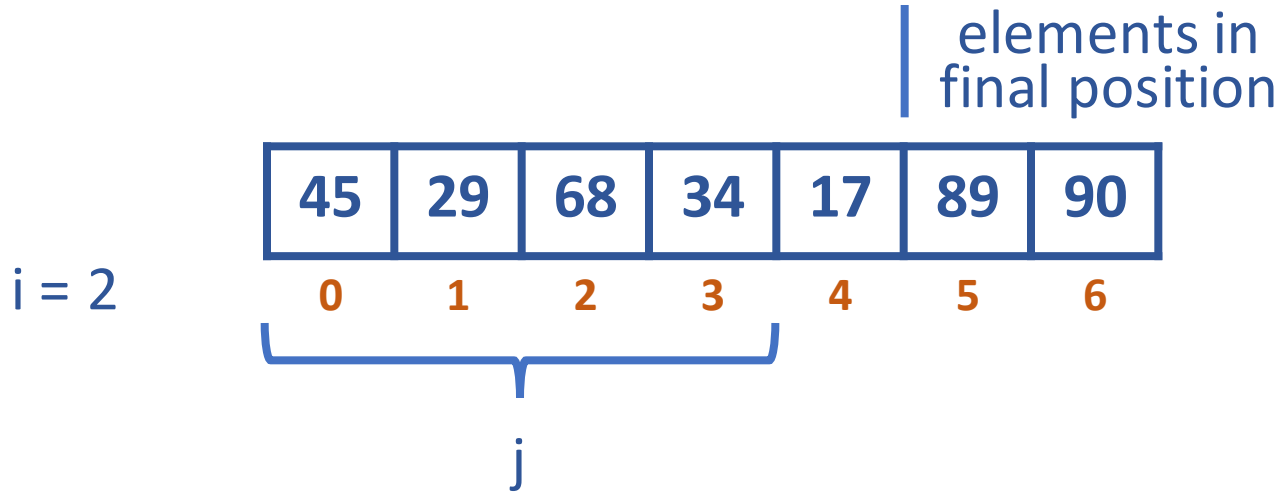
68 < 45 ☒ no swap

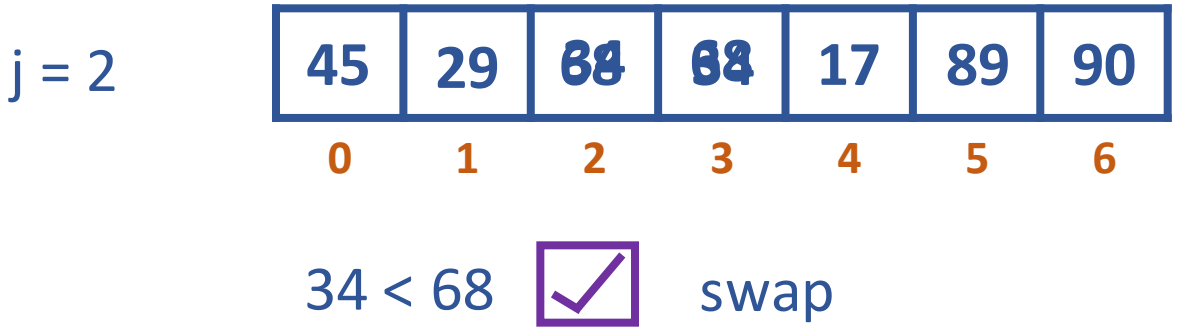
Bubble Sort


elements in
final position
45
68
29
34
17
89
90
0
1
2
3
4
5
6
i = 2
j


j = 1
45
34
17
89
90
0
1
2
3
4
5
6
29 < 68
swap

Bubble Sort

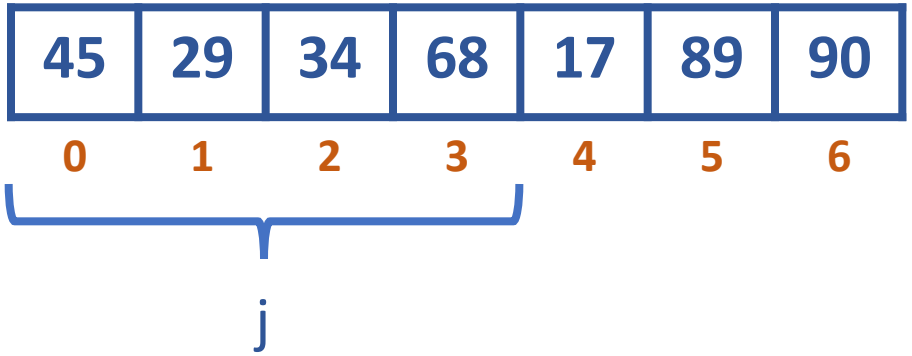

elements in
final position
45
29
68
34
17
89
90
0
1
2
3
4
5
6
i = 2
j


j = 2
45
29
34
68
17
89
90
0
1
2
3
4
5
6
34 < 68
swap

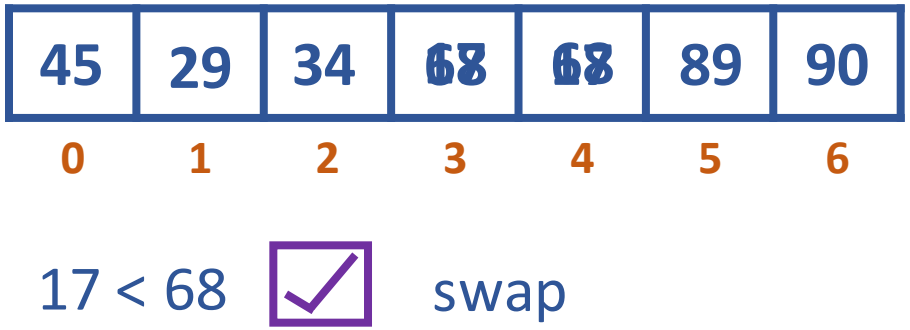
Bubble Sort

elements in final position

i = 2



j = 3

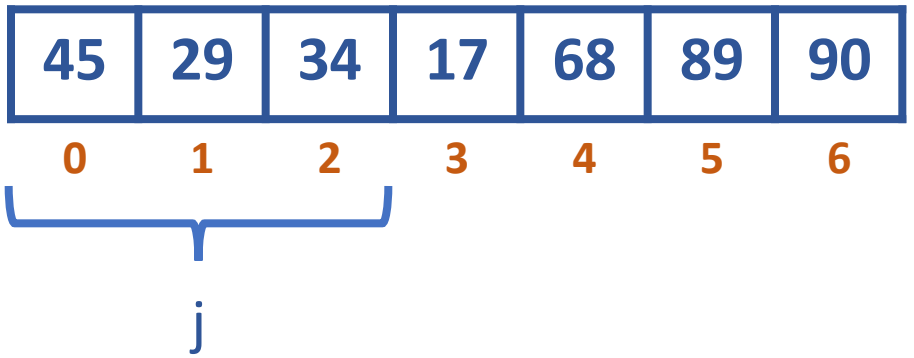


17 < 68 ☑ swap

Bubble Sort

After third iteration

elements in final position



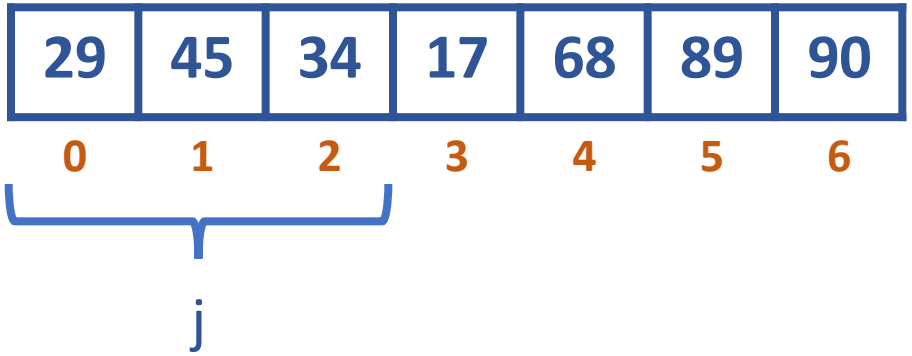
j = 0

29 < 45 ☑ swap

Bubble Sort

elements in final position

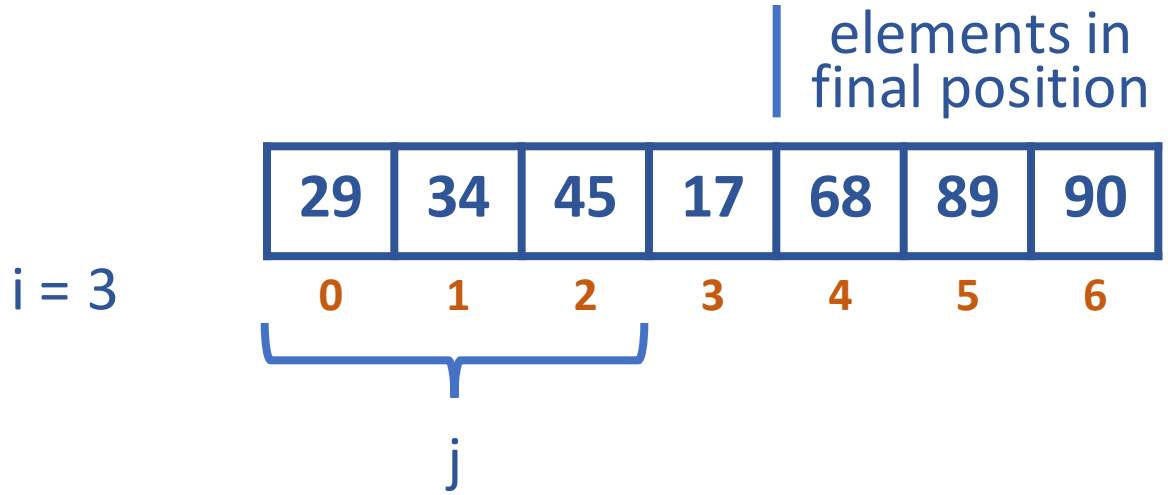
i = 3

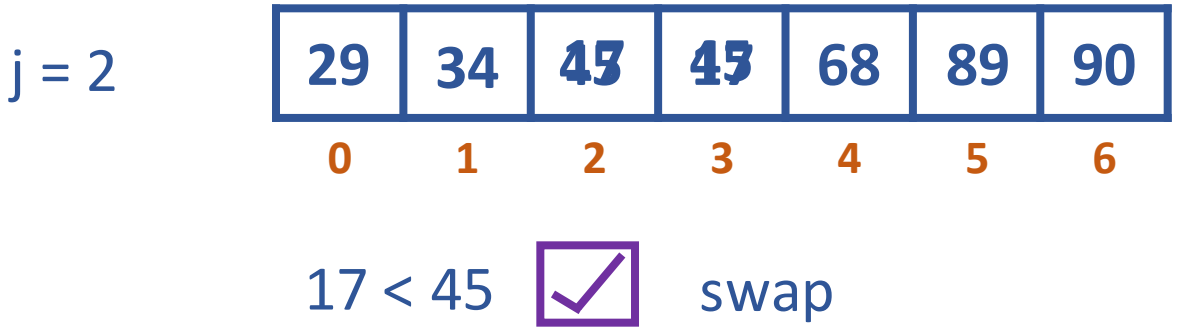


j = 1

34 < 45 ☑ swap

Bubble Sort

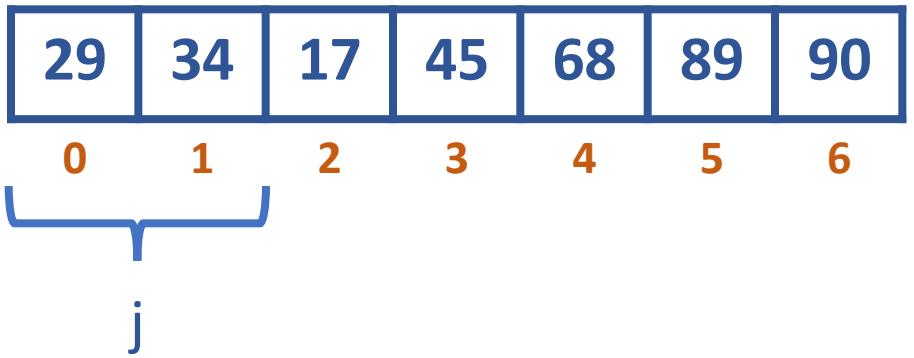

elements in final position
29 34 45 17 68 89 90
0 1 2 3 4 5 6
i = 3
j

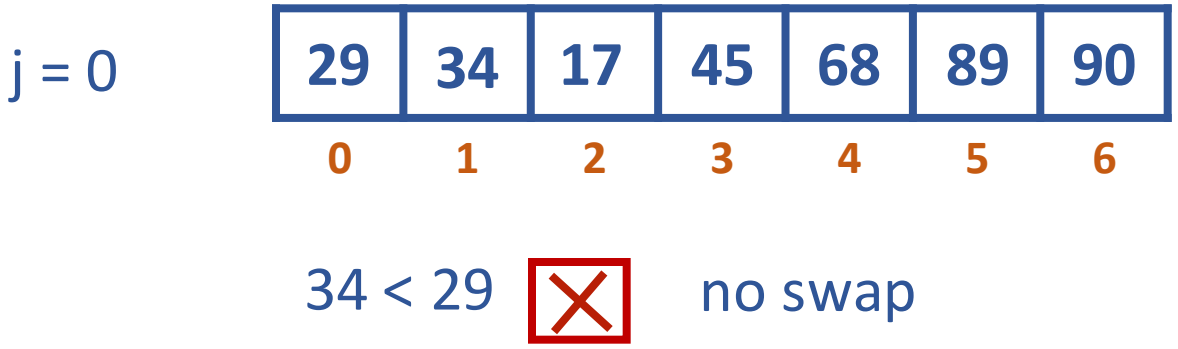

j = 2
29 34 45 17 68 89 90
0 1 2 3 4 5 6
17 < 45
swap

Bubble Sort

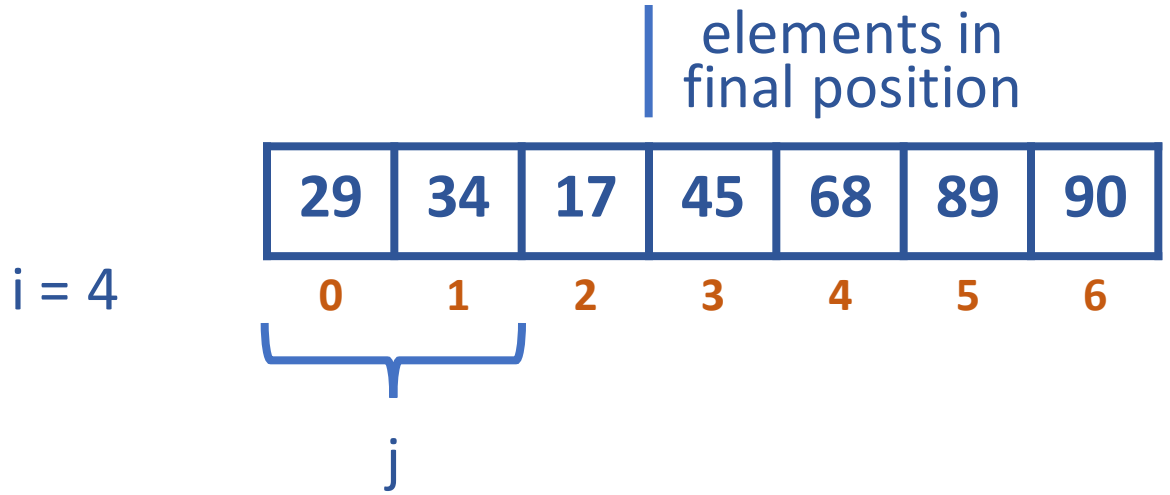
After fourth iteration

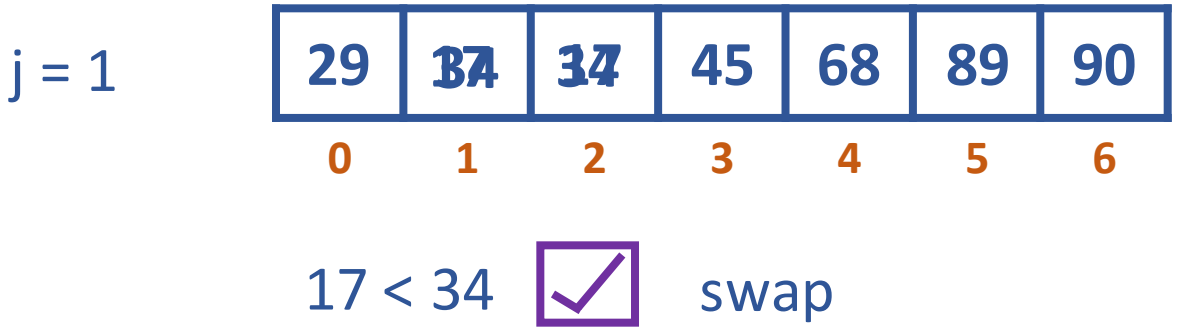
elements in final position





Bubble Sort

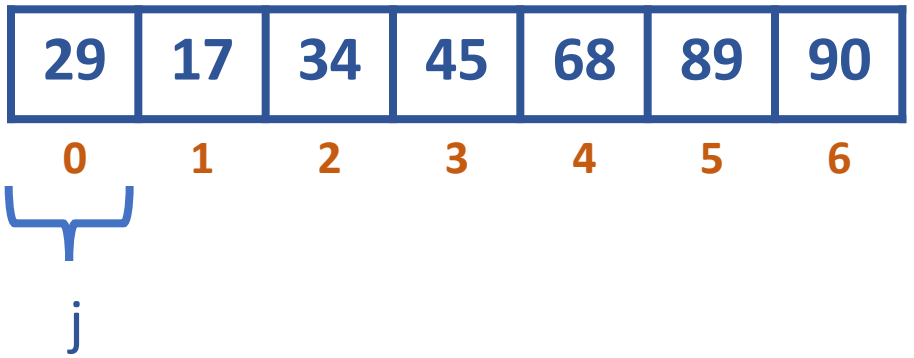

elements in
final position
29
34
17
45
68
89
90
0
1
2
3
4
5
6
i = 4
j

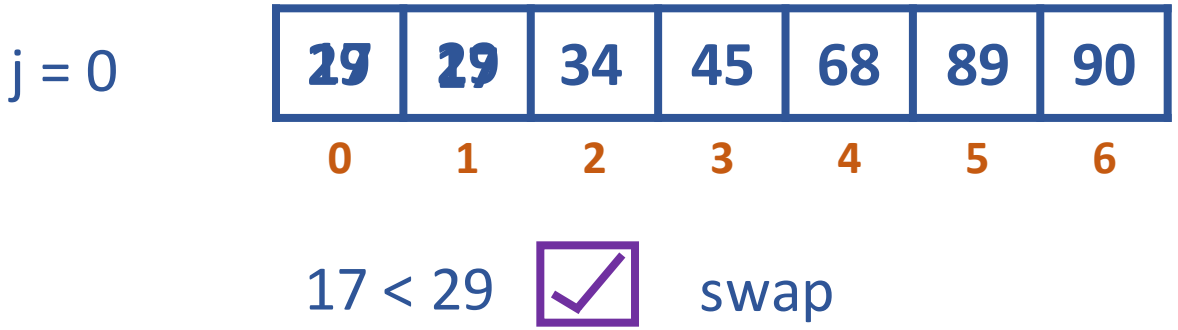

j = 1
29
34
17
45
68
89
90
0
1
2
3
4
5
6
17 < 34
swap

Bubble Sort

After fifth iteration

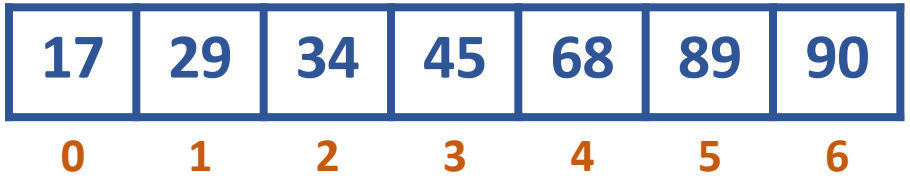
elements in final position





Bubble Sort

After sixth iteration | elements in final position



Bubble Sort

```
ALGORITHM  BubbleSort(A[0 .. n - 1])
//Sorts a given array by bubble sort in  their  final  positions
//Input: An array A[0 .. n  - 1]  of orderable elements
//Output: Array A[0 .. n- 1]  sorted in ascending order
for  i  <-- 0 to n - 2 do
   for  j  <-- 0 to n - 2 - i  do
         if A[ j + 1 ]  <  A[ j ]  swap A[ j ] and A[ j + 1 ]
```

Bubble Sort

Bubble Sort Analysis

$$C(n) = \sum_{i=0}^{n-2} \sum_{j=0}^{n-2-i} 1 = \sum_{i=0}^{n-2} [(n-2-i) - 0 + 1]$$

$$= \sum_{i=0}^{n-2} (n-1-i) = \frac{(n-1)n}{2} \in \Theta(n^2)$$

Bubble Sort is a $\Theta(n^2)$ algorithm

THANK YOU

Shylaja S S

Department of Computer Science & Engineering

shylaja.sharath@pes.edu