Bubble Sort

Bubble Sort also known as exchange sort, is a simple sorting algorithm. It works by repeatedly stepping through the list to be sorted, comparing two items at a time and swapping them if they are in wrong order.

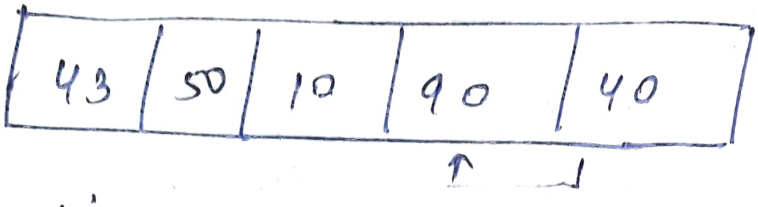
The pass through the list is repeated until no swaps are needed, which means the list is sorted.

The algorithm gets its name from the way smaller elements "bubble" to the top of the list via the swaps.

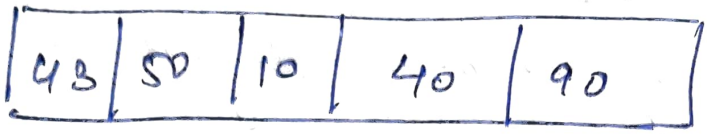
In this method we assume that the lower value items from the set are light and bubble up to the top.

Because it only uses comparison to operate on elements.

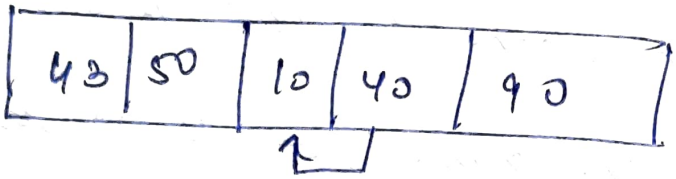
complexity: The Bubble sort is generally considered to be the most inefficient sorting algorithm in common usage. Under best conditions the bubble sort can approach a constant O(n) level of complexity. General case is an extremely bad $O(n^2)$.



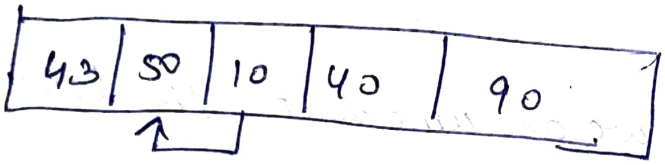
First iteration



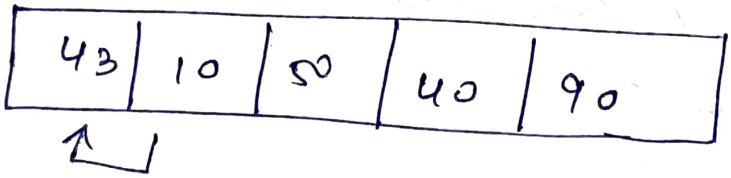
90 is greater than 40 so interchange.



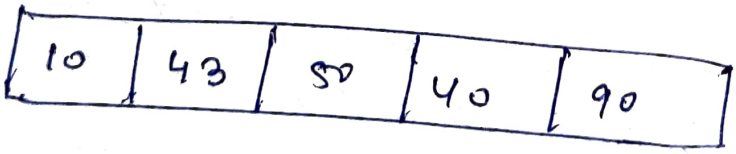
Here 10 is less than 40 so no interchange

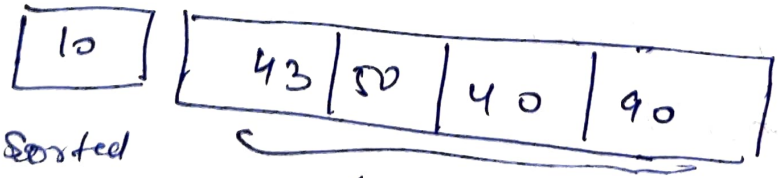


Here 50 greater than 10 so interchange



Here 43 is greater than 10 so interchange

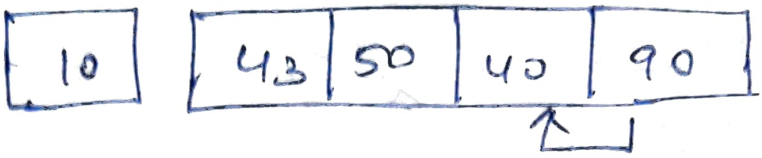




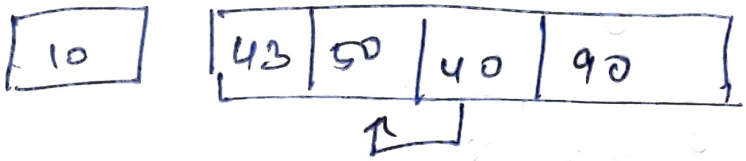
Sorted

Unsorted.

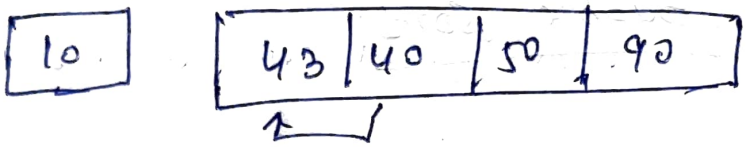
Second Iteration



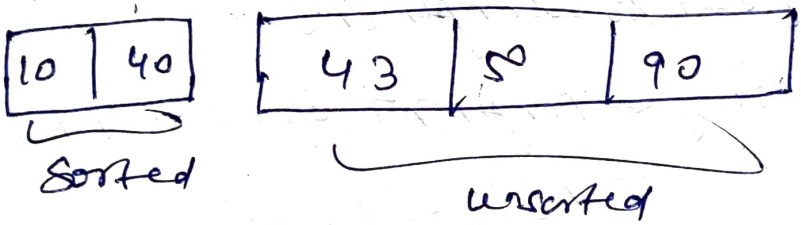
Here 40 is less than 90 so no interchange



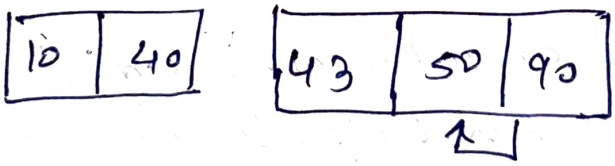
Here 40 is less than 50 so interchange



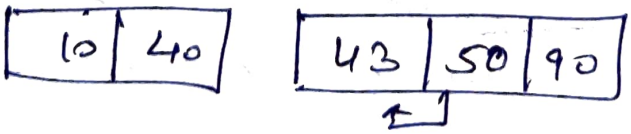
Here 43 is greater than 40 so interchange



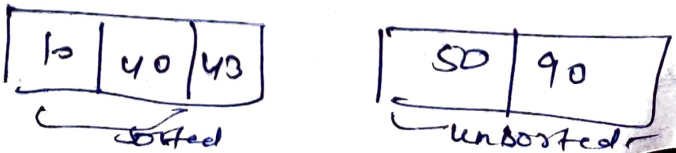
Third Iteration



Here 50 is less than 90 so no interchange



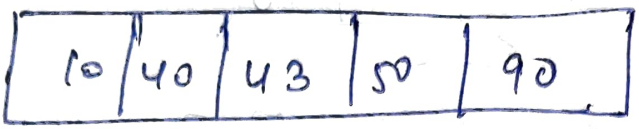
Here 43 is less than 50 so no interchange.



fourth iteration

Here 50 is less than 90 so no interchange

Final sorted array is.



Algorithm of Bubble sort:

```
Bubble_Sort(A)
1. For i ← 1 to length[A]
2.     For j ← length[A] down to i+1
3.         if A[j] < A[j-1]
4.             exchange(A[j], A[j-1])
```

The outer loop is executed in n-1 times. Each time the outer loop is executed, the inner loop is executed. Inner loop executed in n-1 times at first, linearly dropping to just once. On average, inner loop executed about n/2 times for each execution of the outer loop. In the inner loop comparison is always done (in constant time) the swap might be done. Thus the result is n*n/2*k

i.e. $O(n^2/2 + k) = O(n^2)$

Q. Illustrate the operation of Bubble Sort on the array.

$$A = \langle 5, 2, 1, 4, 3, 7, 6 \rangle$$

Sol^n Here $A = \langle 5, 2, 1, 4, 3, 7, 6 \rangle$

So length[A] = 7

i = 1 to 7 and j = 7 to 2

Take i = 1 and j = 7

A[7] = 6 and A[6] = 7

Here A[7] < A[6] so interchange A[7] and A[6]

now A[] =

1	2	3	4	5	6	7
5	2	1	4	3	6	7

Now i = 1, j = 6 then A[6] = 6

A[5] = 3 and A[5] < A[6]

So, No interchange.

Now i = 1, j = 5 then A[5] = 3

A[4] = 4 and A[4] > A[5]

So exchange (A[5], A[4])

and A[] =

5	2	1	3	4	6	7

Now i=1, j=4 then A[4]=3

A[3] = 1 and A[4] > A[3]

Now i=1 j=3 then A[3]=1

A[2] = 2 and A[3] < A[2]

So, exchange (A[3], A[2])

then A[] =

5	1	2	3	4	6	7

Now, i=2, j=2 then A[2]=1

A[1]=5 and A[2] < A[1]

So, exchange (A[2], A[1])

then A[] =

1	5	2	3	4	6	7

Now, i=1, j=7 then A[7]=7

A[6]=6 and A[7] > A[6]

No interchange

Similarly i=2, j=6, 5, 4 No exchange

then i=2 j=3

A[3] = 2

A[2]=5 and A[3] < A[2]

So, exchange (A[3], A[2])

and A[] =

1	2	5	3	4	6	7

Now, i=3, j=7, 6, 5 No change

then i=3, j=4

A[4]=3

A[3]=5 and A[4]<A[3]

So exchange (A[4], A[3])

then A[] = | 1 | 2 | 3 | 5 | 4 | 6 | 7 |

Now i=4, j=7 No change

then i=4, j=6 No change

Now i=4, j=5

then A[5]=4 and A[4]=5

A[5] < A[4]

So exchange A[5], A[4] and we get

A[] = | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

This is final sorted array.

Analysis

In the Bubble Sort, the first pass requires $(n-1)$ comparisons to fix the highest element to its location, the second pass requires $(n-2)$ ... k^{th} pass requires $(n-k)$ and the last pass requires only one comparision to be fixed at its proper position.

Therefore, the total comparisons are;

$f(n) = (n-1) + (n-2) + (n-3) \cdots + (n-k) + \cdots$

$+ 2 + 1$

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

$f(n) = O(n^2)$