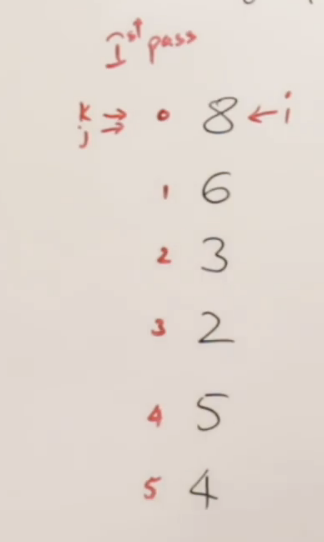
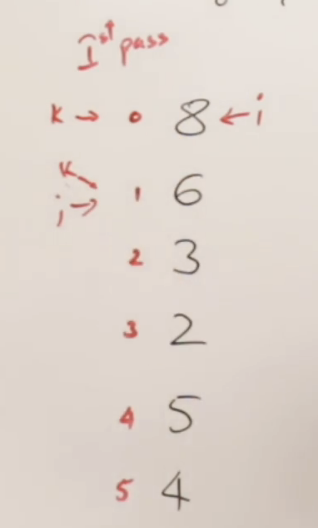
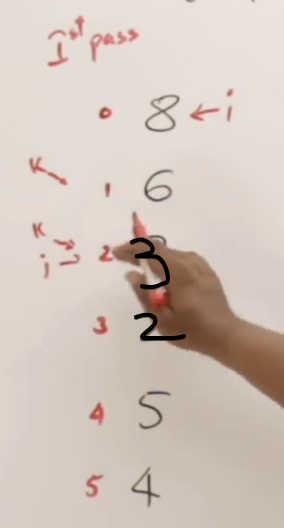
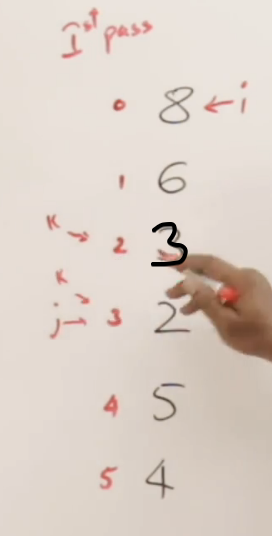
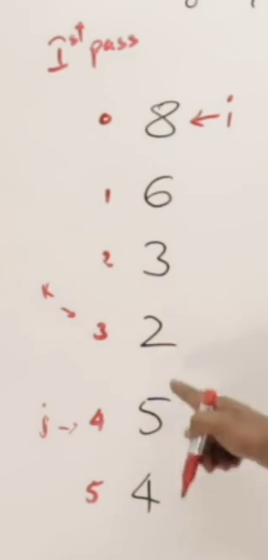
J will find the minimum element, it will pull k towards to it.

Move j, and compare j and k.  
Compare 8 and 6.  
j<k, 8<6 -> No, So j is smaller, So bring k to j.

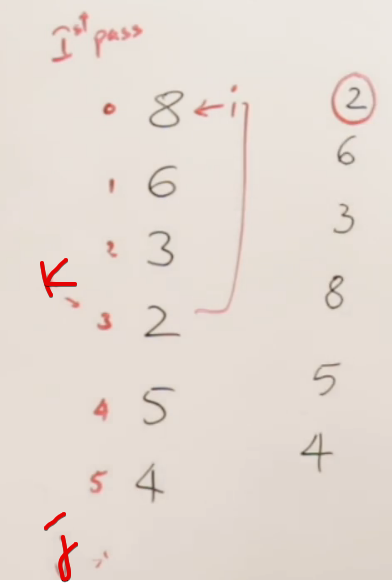
  A picture containing scatter chart

Description automatically generated A picture containing text

Description automatically generated

Now we reached to the end, and found the smallest element. (k=2)

We have not swapped any elements, just got the index of the smallest element.



Now we are swapping ith index and kth index.  
So we got the smallest element in the 0th index.

Selecting a index, and then finding out the element.

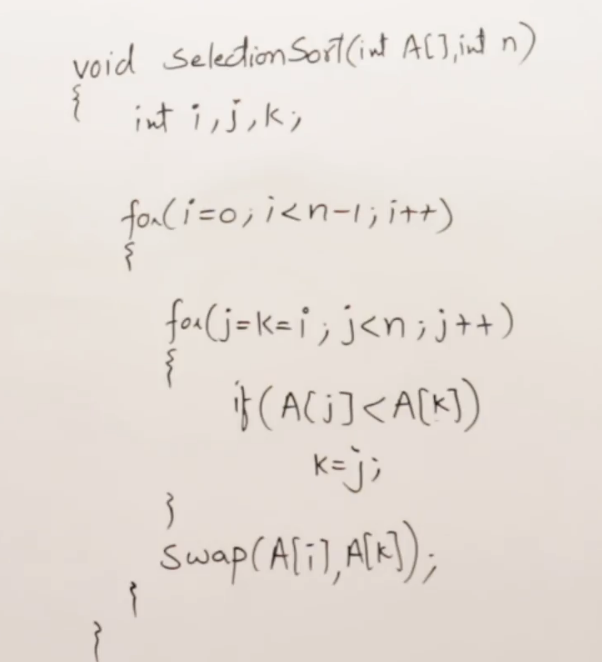
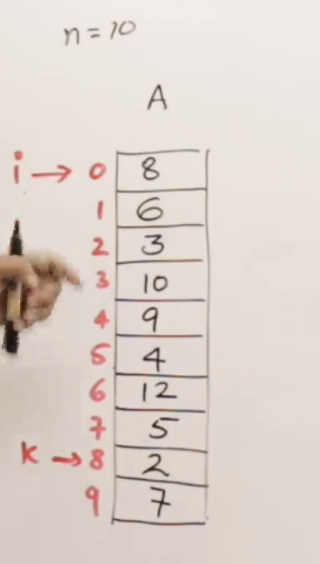
This is the only algortihm that allows minimum number of swaps.

After every pass

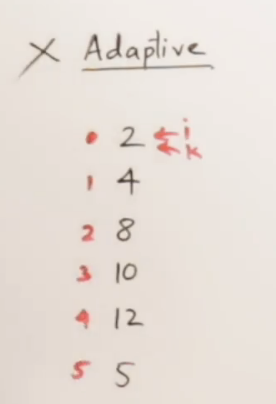
Bubble Sort 🡪 Finds the largest element at the last  
Selection Sort 🡪 Find the smalles element in the beginning

# **Program**

j will scan through all the elements upto N.  
If smaller elements found, j will bring down k towards it.

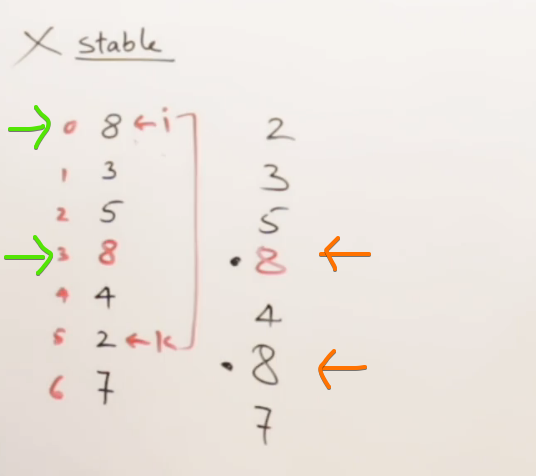


# **Adaptive**



By Nature, Selection Sort is not adaptive.  
Whether the array is sorted or not, it is going to take **O(n^2)**

# **Stable**



By Nature, Selection Sort is not stable. Since the order is not preserved.

# **Key Points**

🡪 Minimum number of swaps.

🡪 Use-ful intermediate result (after k-passes, will get k-smallest element)