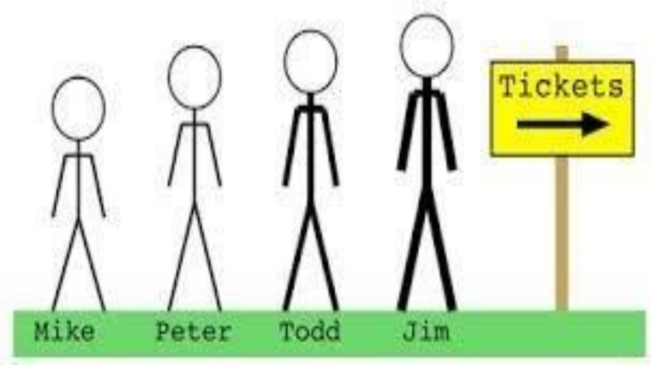
Sorting Techniques

PSAT 2021

Objectives

- ❖Note the need for re-arrangement of a collection of data.
- ❖Learn selection sorting technique.
- ❖Learn bubble sorting technique.



Sorting

- Sorting is the re-arrangement of a collection of data in either ascending or descending order.
- It is a common activity in data management.

- *Input:* A sequence of *n* elements
  - $a_1, a_2, \ldots, a_n$

- *Output:* A reordered list of *n* elements
  - $a_1', a_2', \ldots, a_n'$ of the input sequence
  - such that $a_1' \leq a_2' \leq \cdots \leq a_n'$ or $a_1' >= a_2' >= \cdots >= a_n'$

✓*Can you recollect where you have encountered sorted items / lists?*

Sorting techniques

- Selection sort
- Bubble sort
- Insertion sort
- Merge sort
- Quick sort

Selection sort

- Selection sort (ascending order) involves
  - *scanning* through the list to find (or select) the smallest element.
  - *swapping* it with the first element.
- The rest of the list is then searched for the next smallest and it is swapped with the second element.
- This process is repeated until the rest of the list reduces to one element, by which time the list is sorted.
- The list is rearranged from smallest to largest.

General strategy

- The list is divided into two sub lists: *sorted* and *unsorted* which are divided by an imaginary wall.
- We *select* (find) the smallest element from the unsorted sub list
- We *swap* it with the element at the beginning of the unsorted data.
- After each selection and swapping,
  - The imaginary wall between the two sub lists move one element ahead.
  - Increasing the number of sorted elements and decreasing the number of unsorted ones.
- Each time we move one element from the unsorted sub list to the sorted sub list, we say that we have completed a *pass*.
- A list of *n* elements requires *n-1* passes to completely rearrange the data.

General strategy *contd*---

- Repeatedly *scan* through the list of elements.
- *Select* the smallest element (*S in figure below*).
- *Swap* the smallest element with the first element (*F in figure below*) in the remaining list of elements.

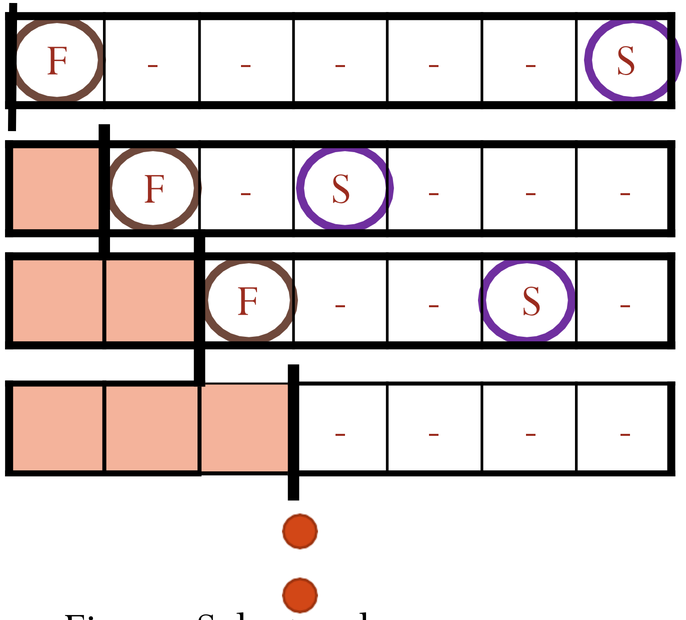
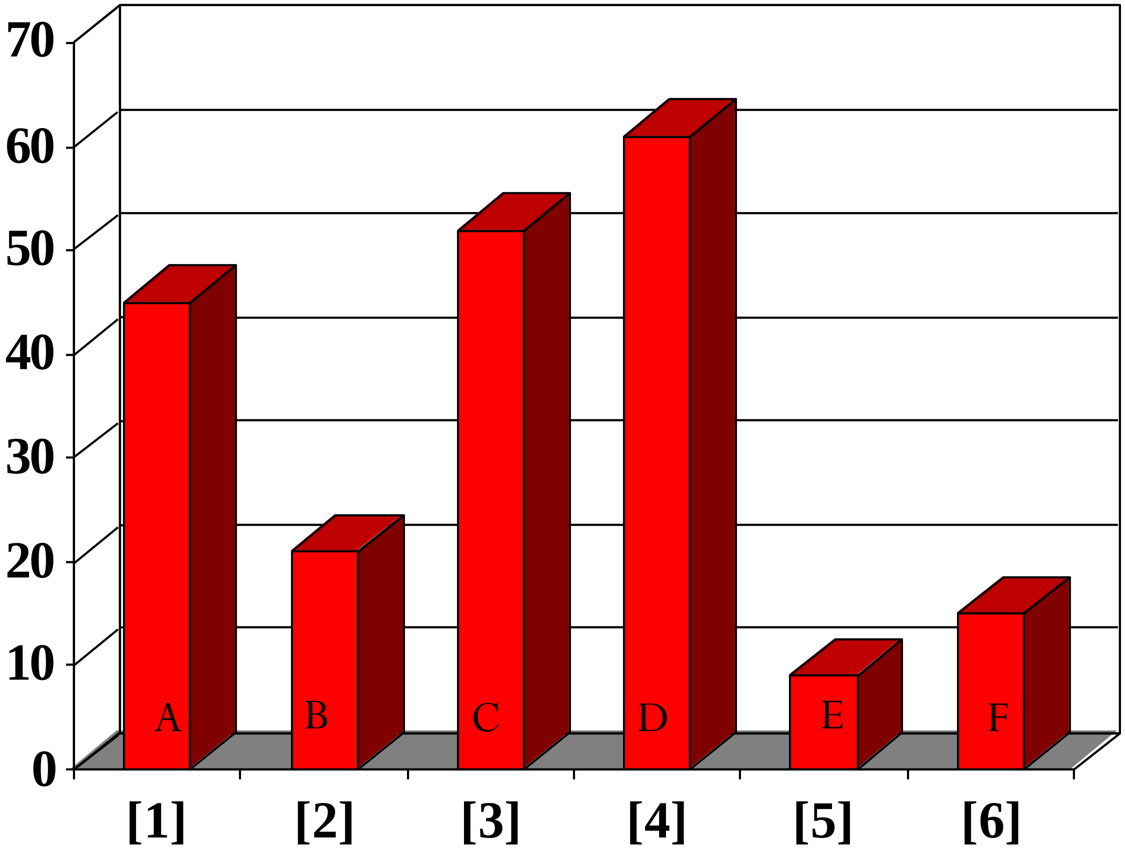
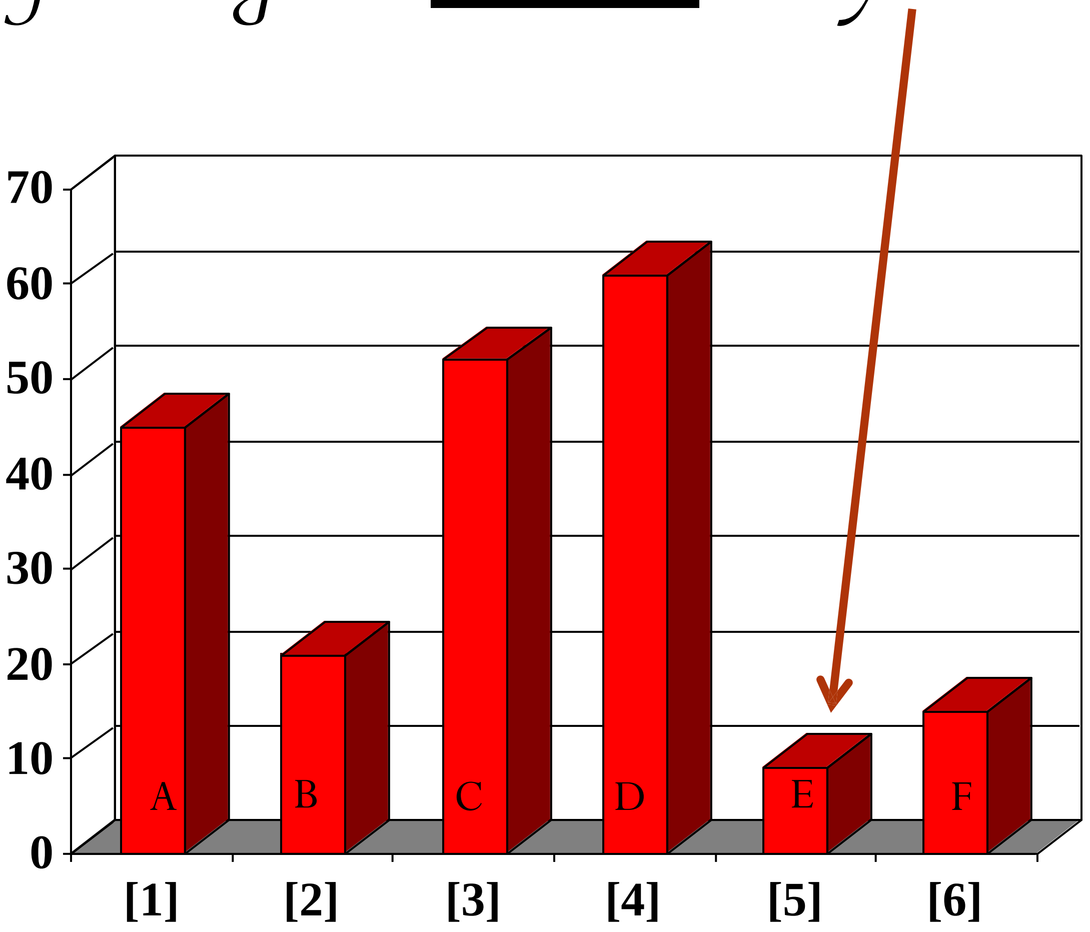


Figure: Select and swapprocess

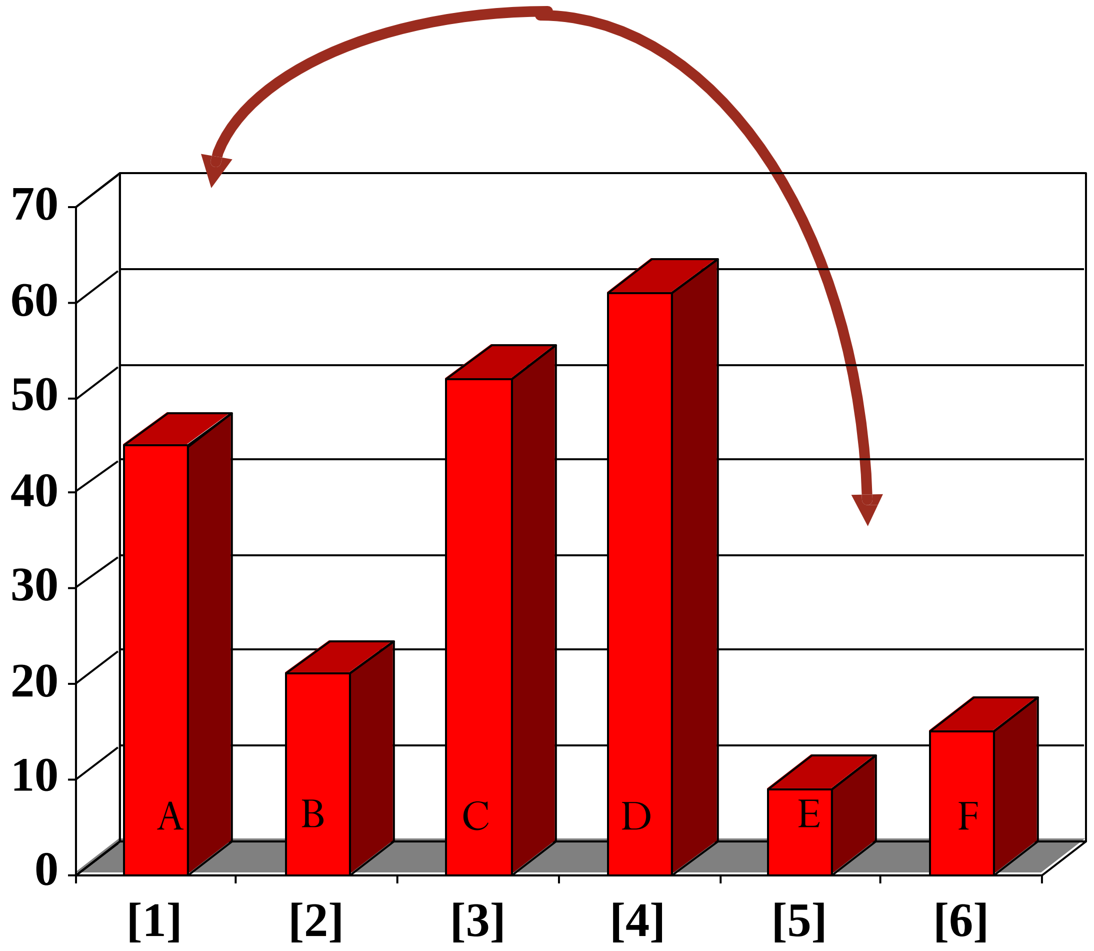
Eg: Sort a list of 6 elements from smallest to largest



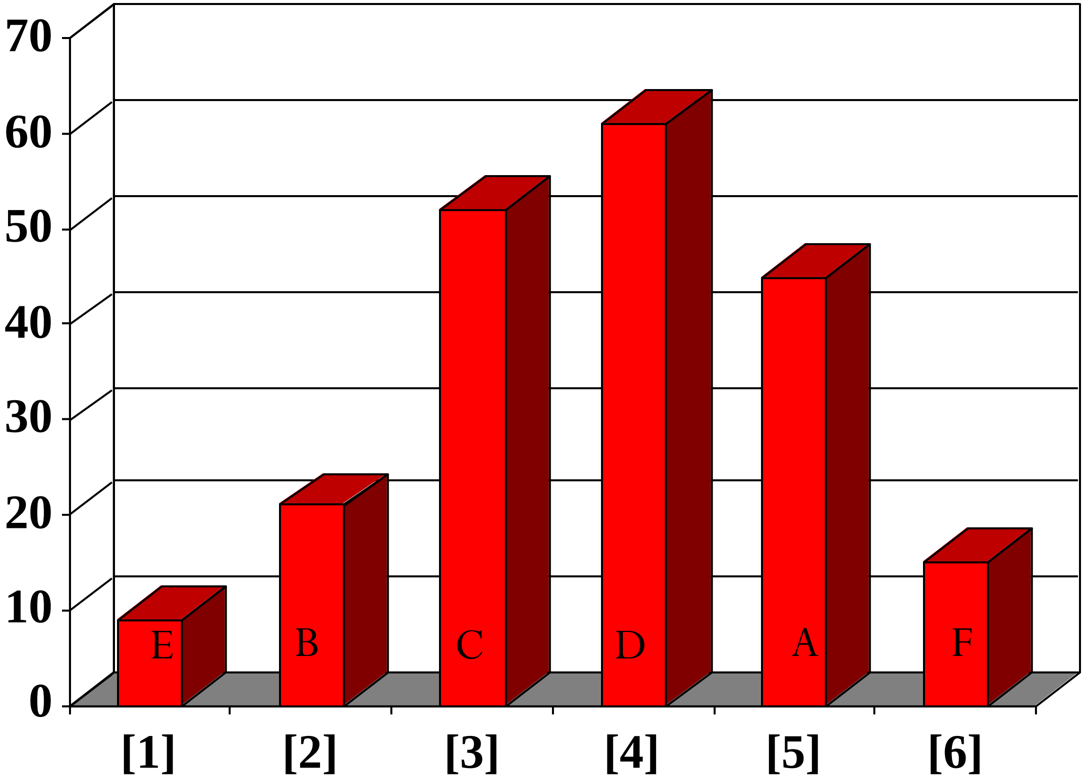
Start by finding the <u>smallest</u> entry.



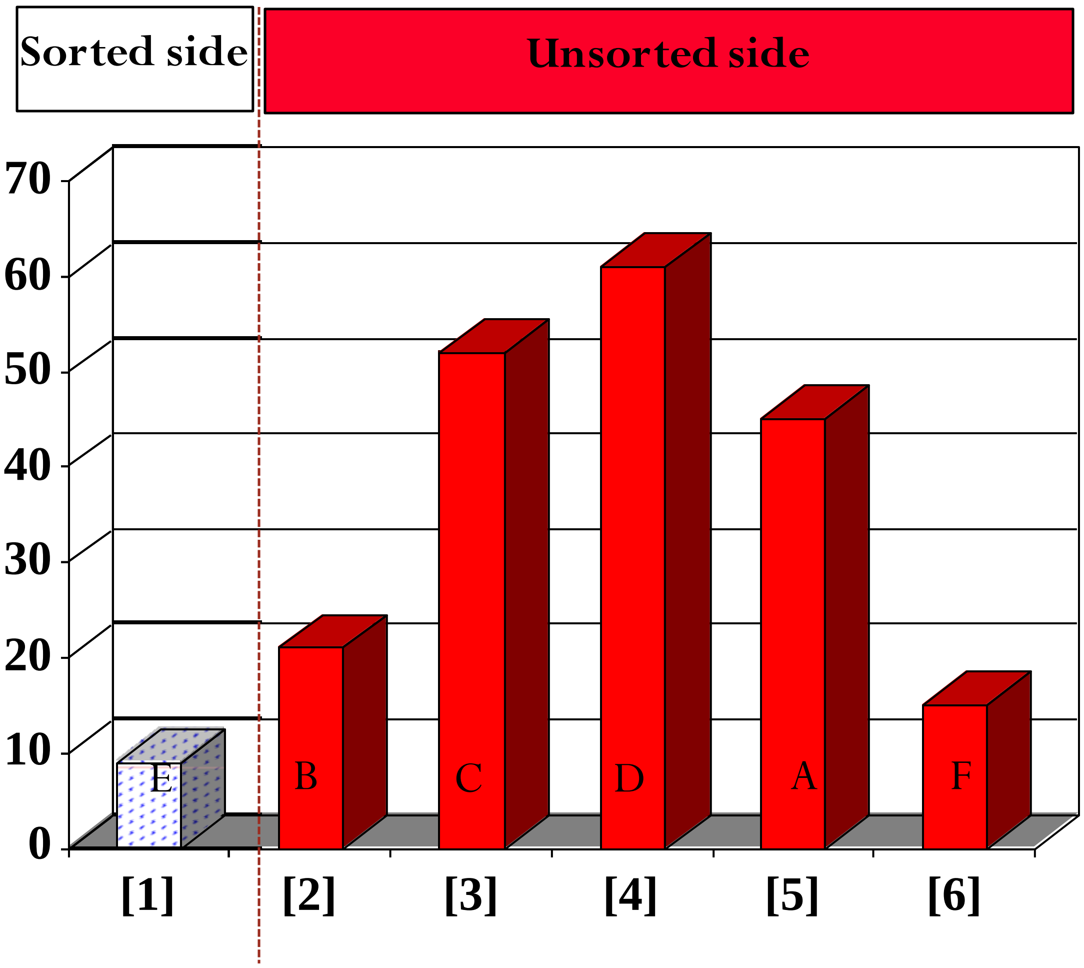
Swap the smallest entry with the first entry.



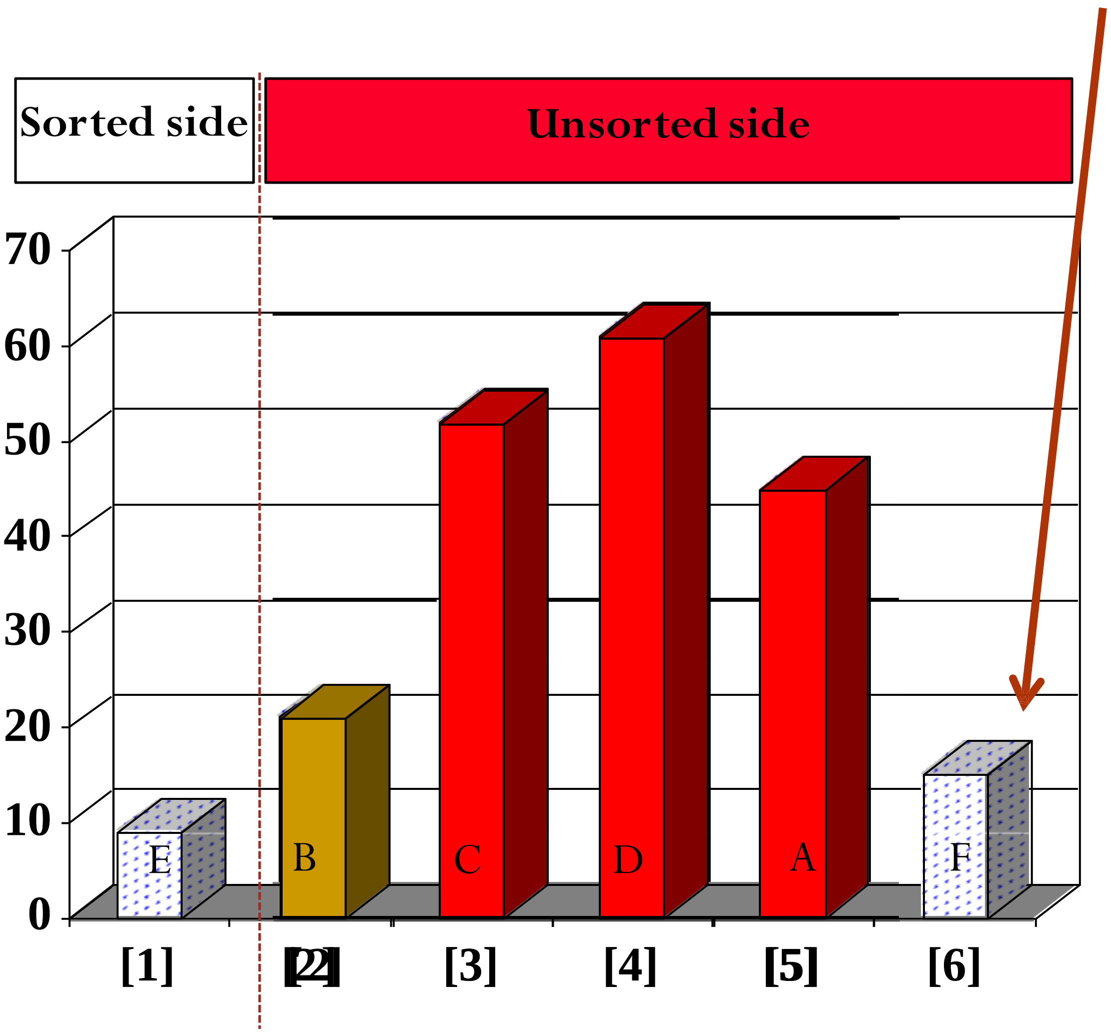
✓ *We have swapped the smallest entry with the first entry.*



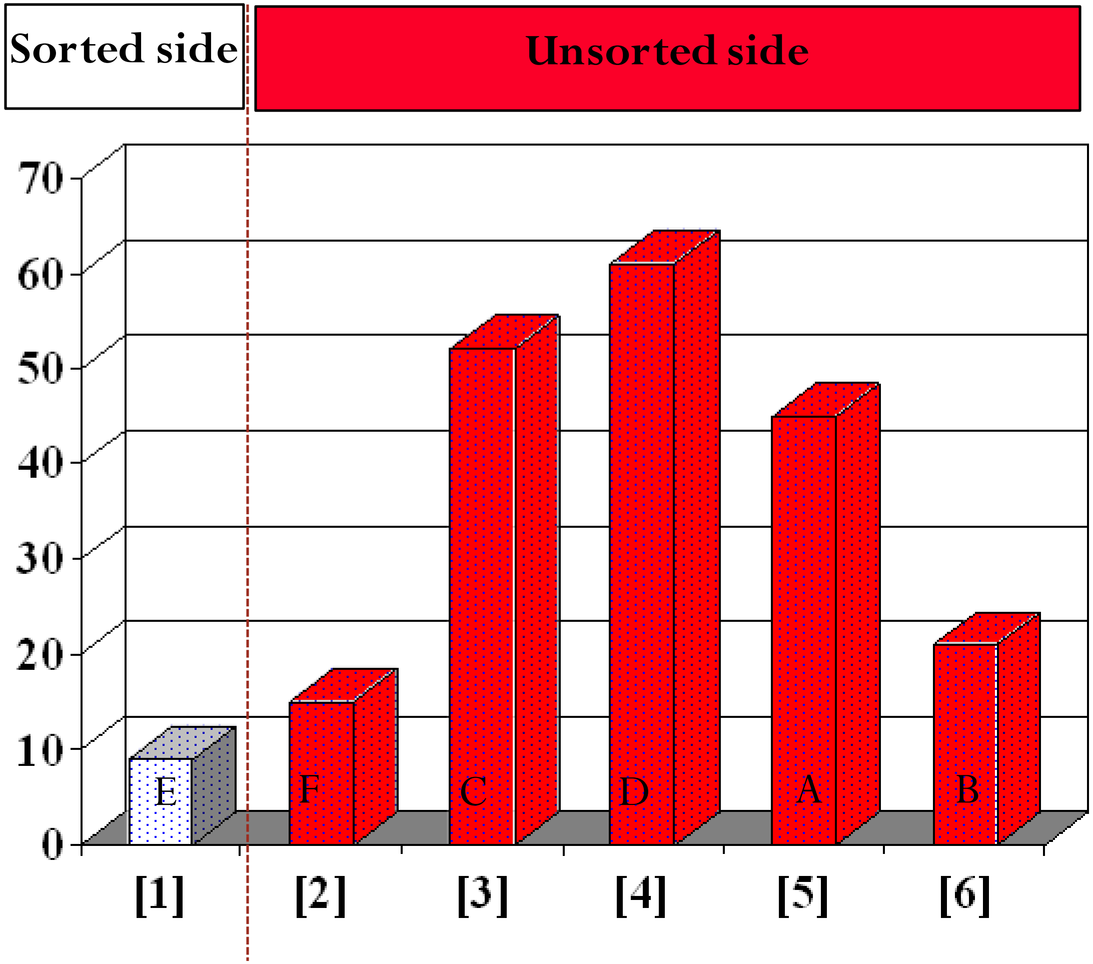
✓ *Part of the array is now sorted…*



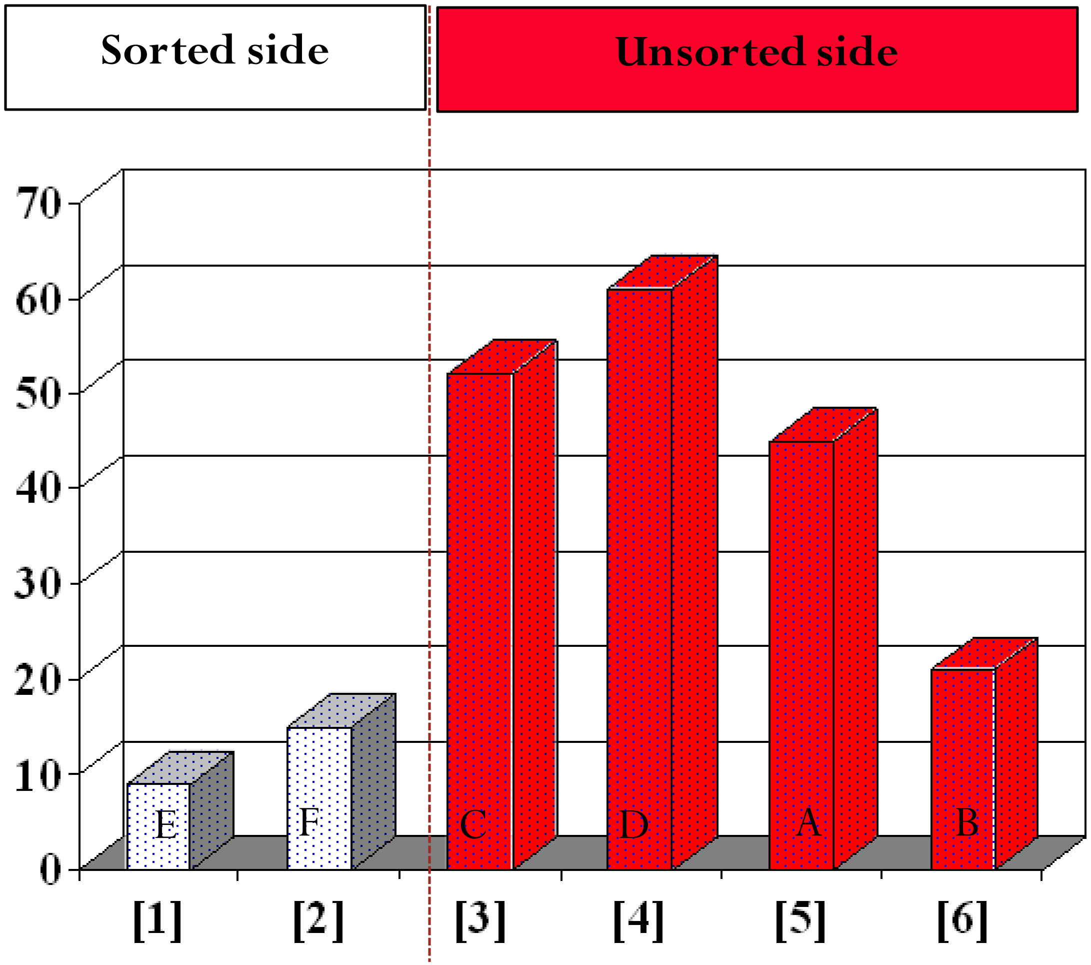
Now, find the smallest entry in the unsorted side.



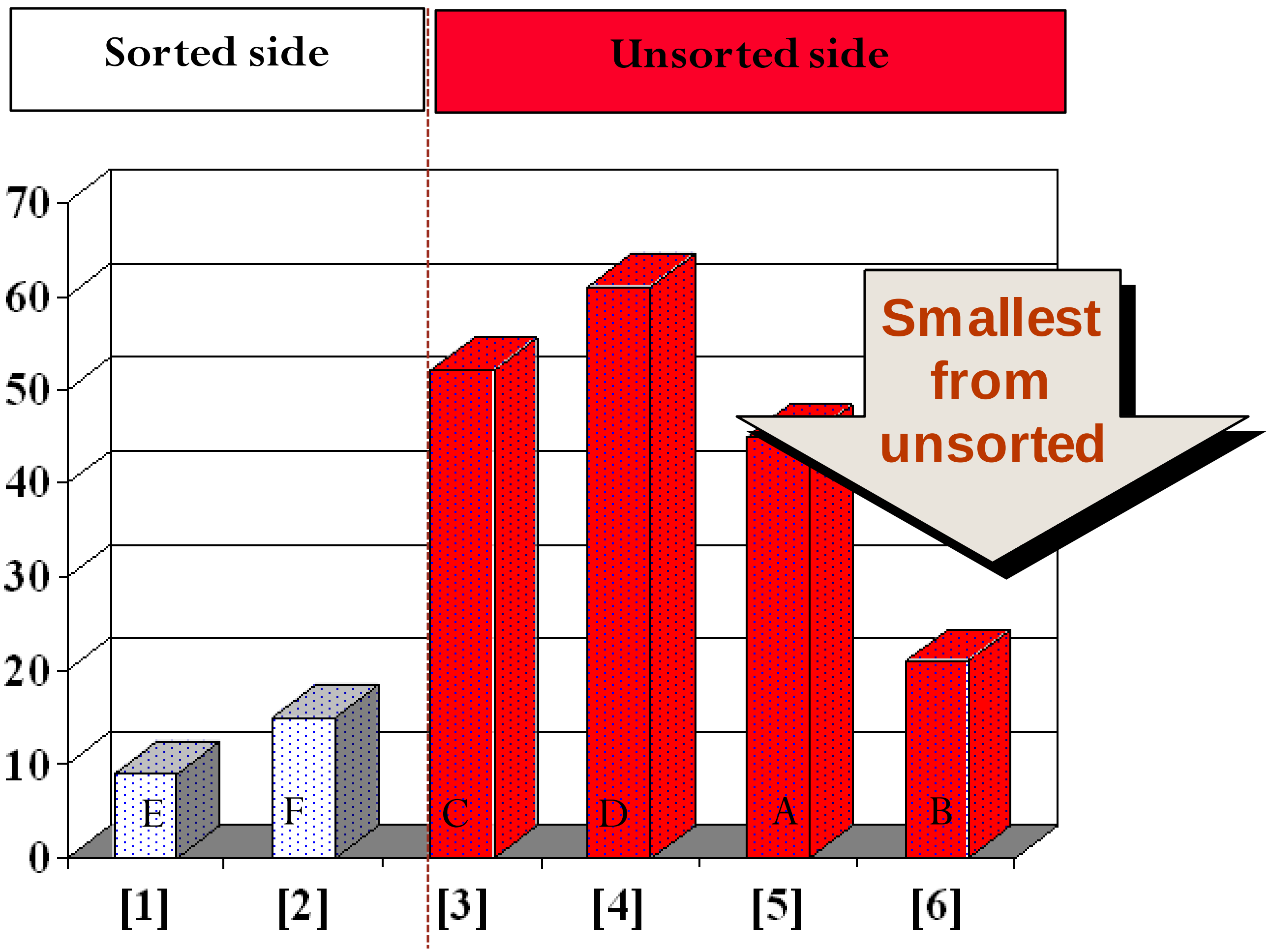
Swap this smallest entry with the front of the unsorted side.



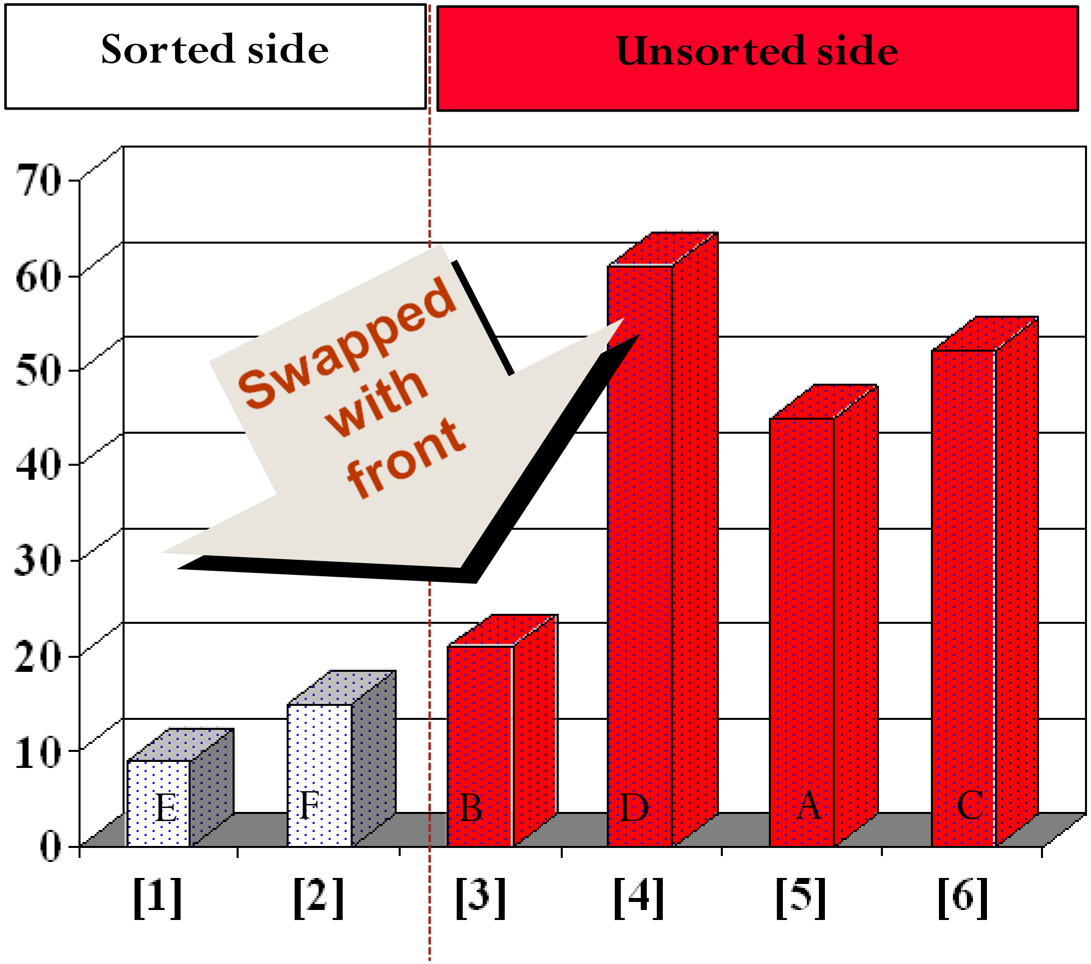
We have increased the size of the sorted side by one entry.



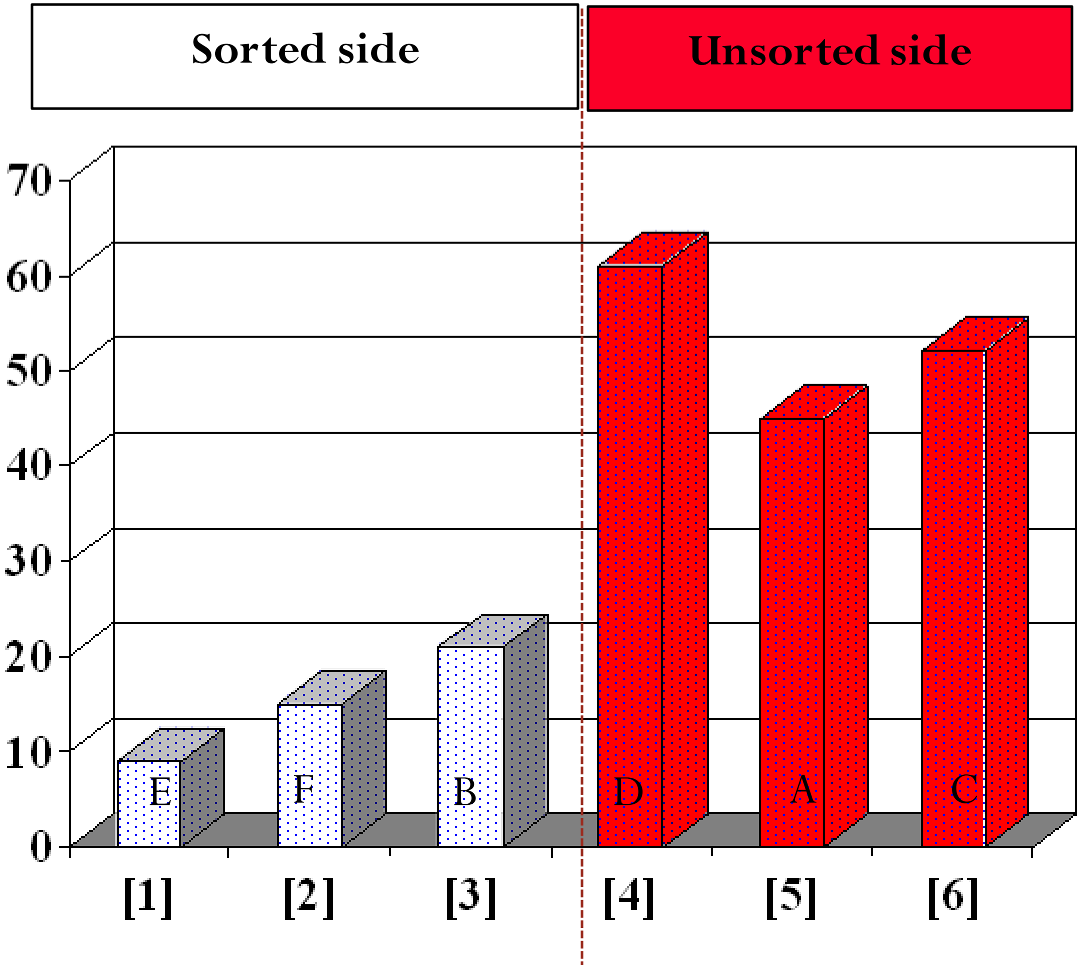
✓ The process continues…



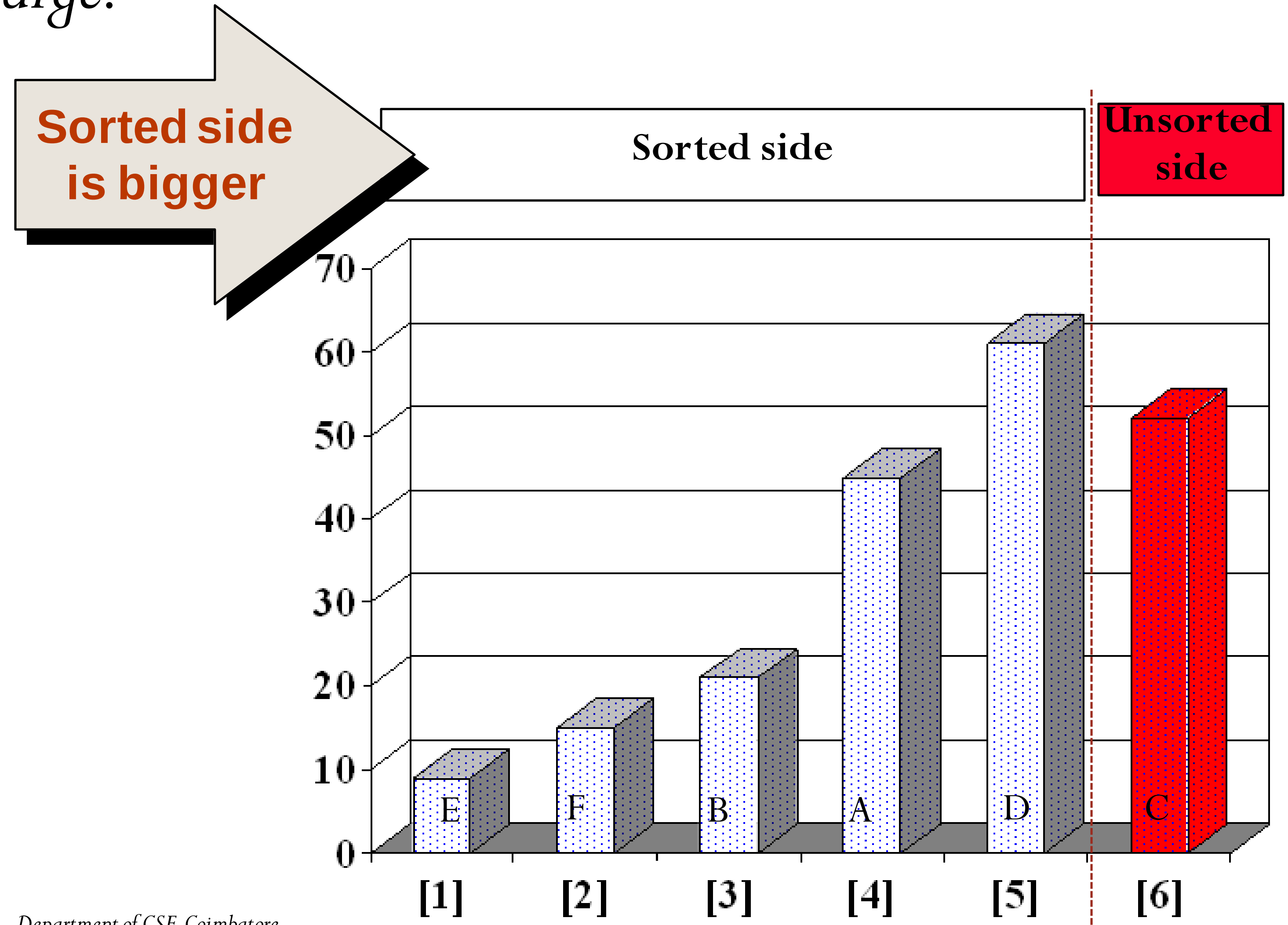
✓ The process continues…



✓ *The process continues…*



- ✓ The process keeps adding one more entry to the sorted side.
- ✓ The sorted side has the smallest entries, arranged from small to large.

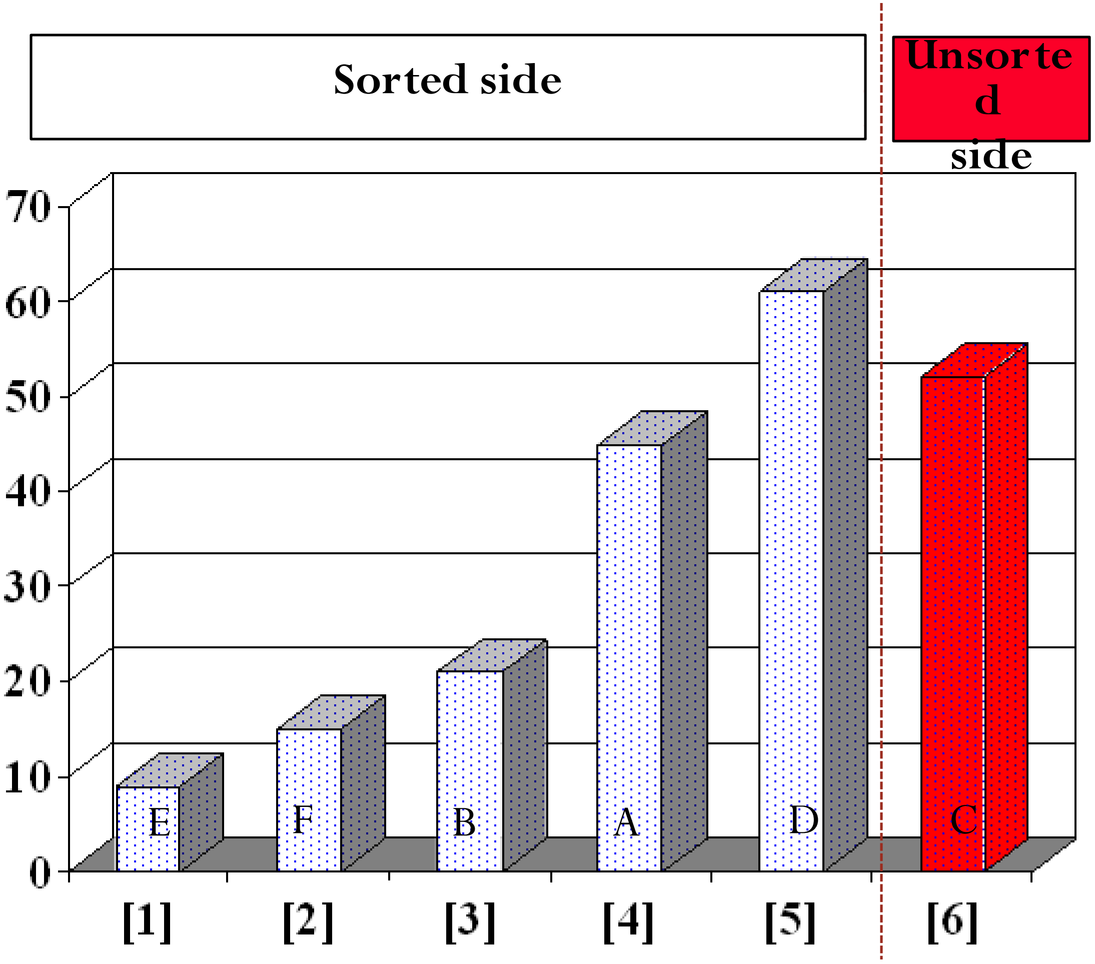


Sorted side is bigger

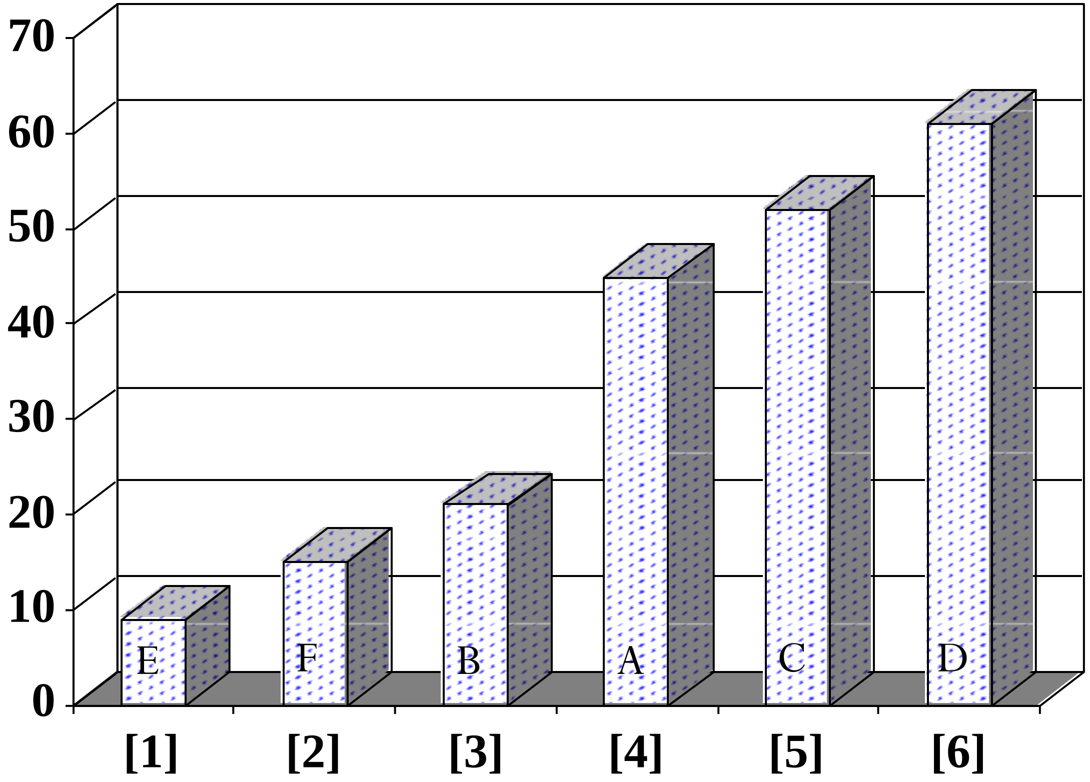
Sorted side

Unsorted side

✓ *We can stop when the unsorted side has just one entry, since that entry must be the largest entry.*

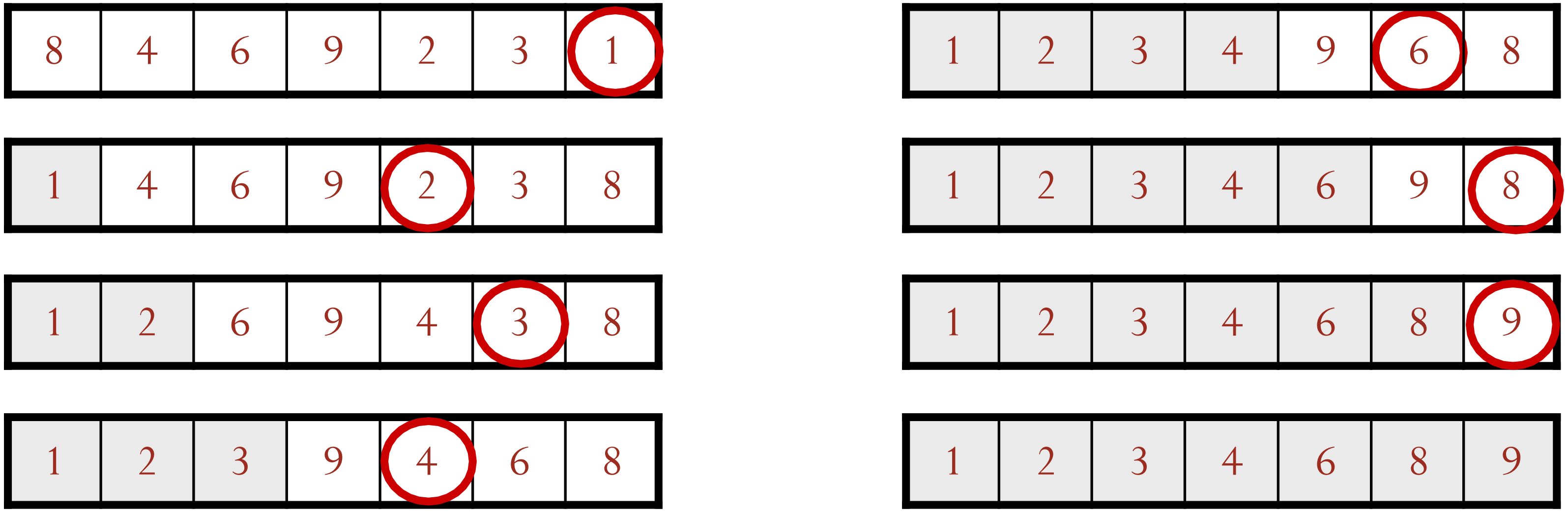


✓ *The list is now sorted…*



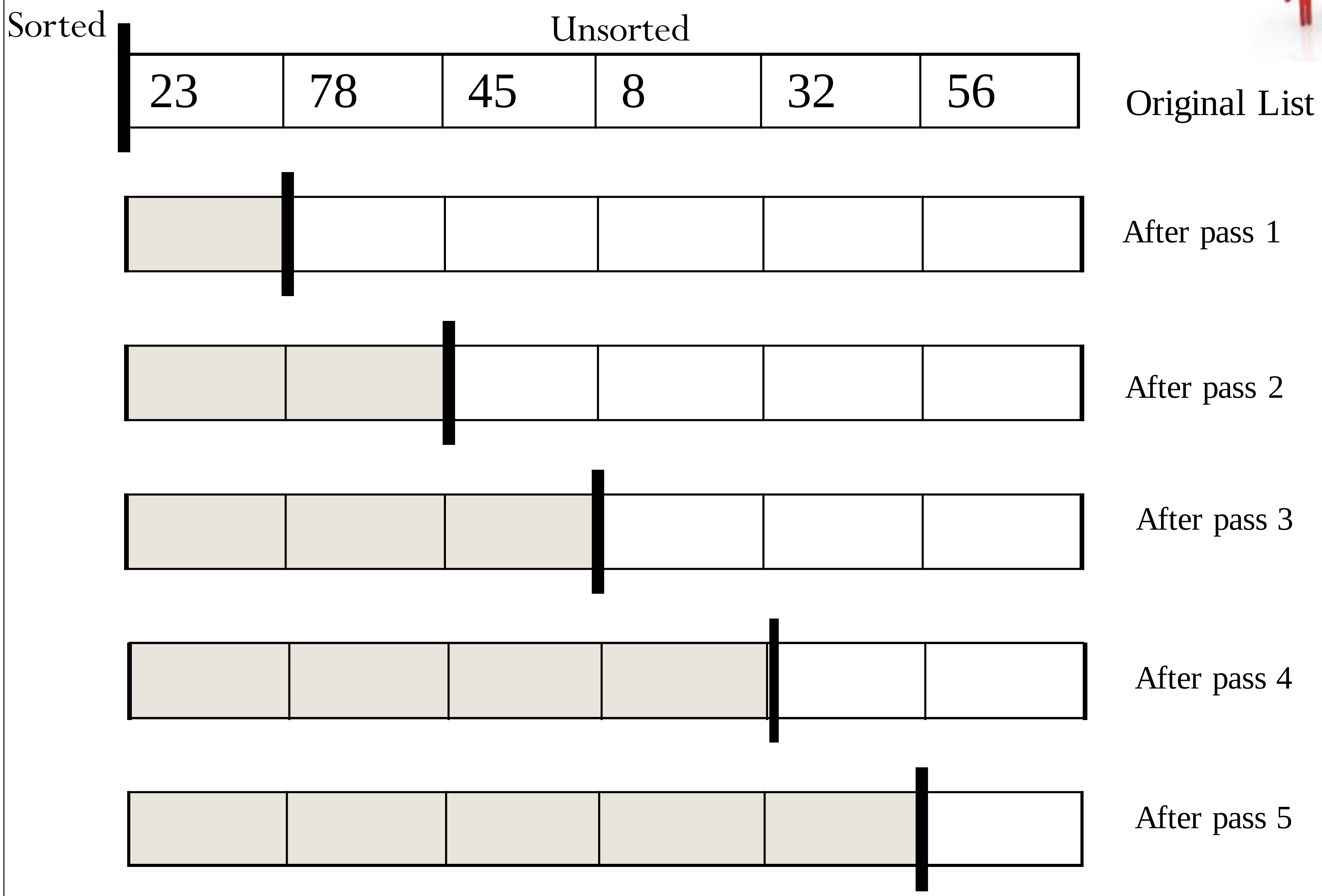
✓ *We repeatedly* ***selected*** *the smallest entry, and* ***moved*** *this entry to the front of the unsorted side.*

Eg:The scan, select and swap process



8	4	6	9	2	3	1
1	4	6	9	2	3	8
1	2	6	9	4	3	8
1	2	3	9	4	6	8
1	2	3	4	9	6	8
1	2	3	4	6	9	8
1	2	3	4	6	8	9
1	2	3	4	6	8	9

✓ *Perform selection sort*



Sorted

Unsorted

23	78	45	8	32	56

Original List

After pass 1

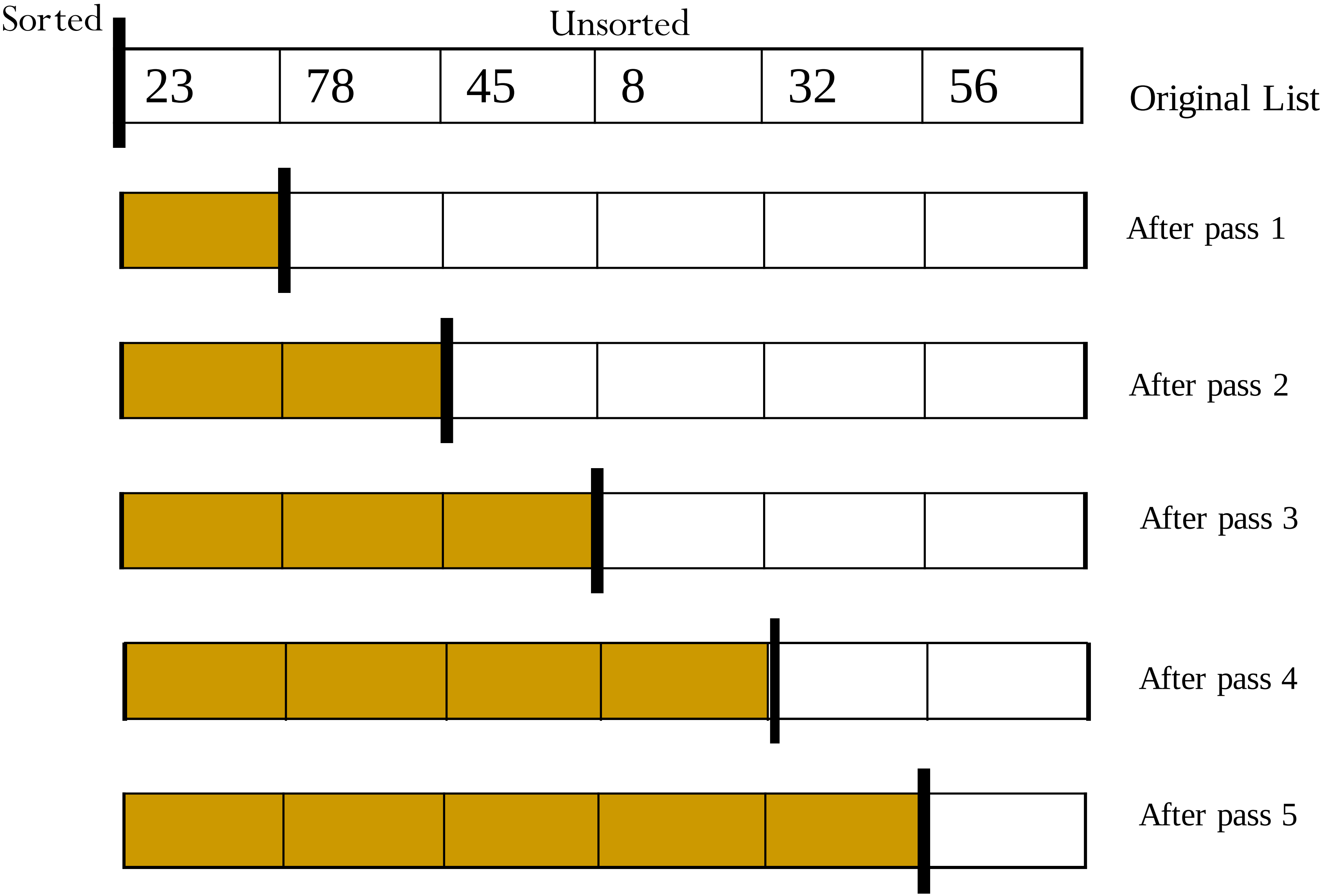
After pass 2

After pass 3

After pass 4

After pass 5

Solution



Sorted

Unsorted

23	78	45	8	32	56

Original List

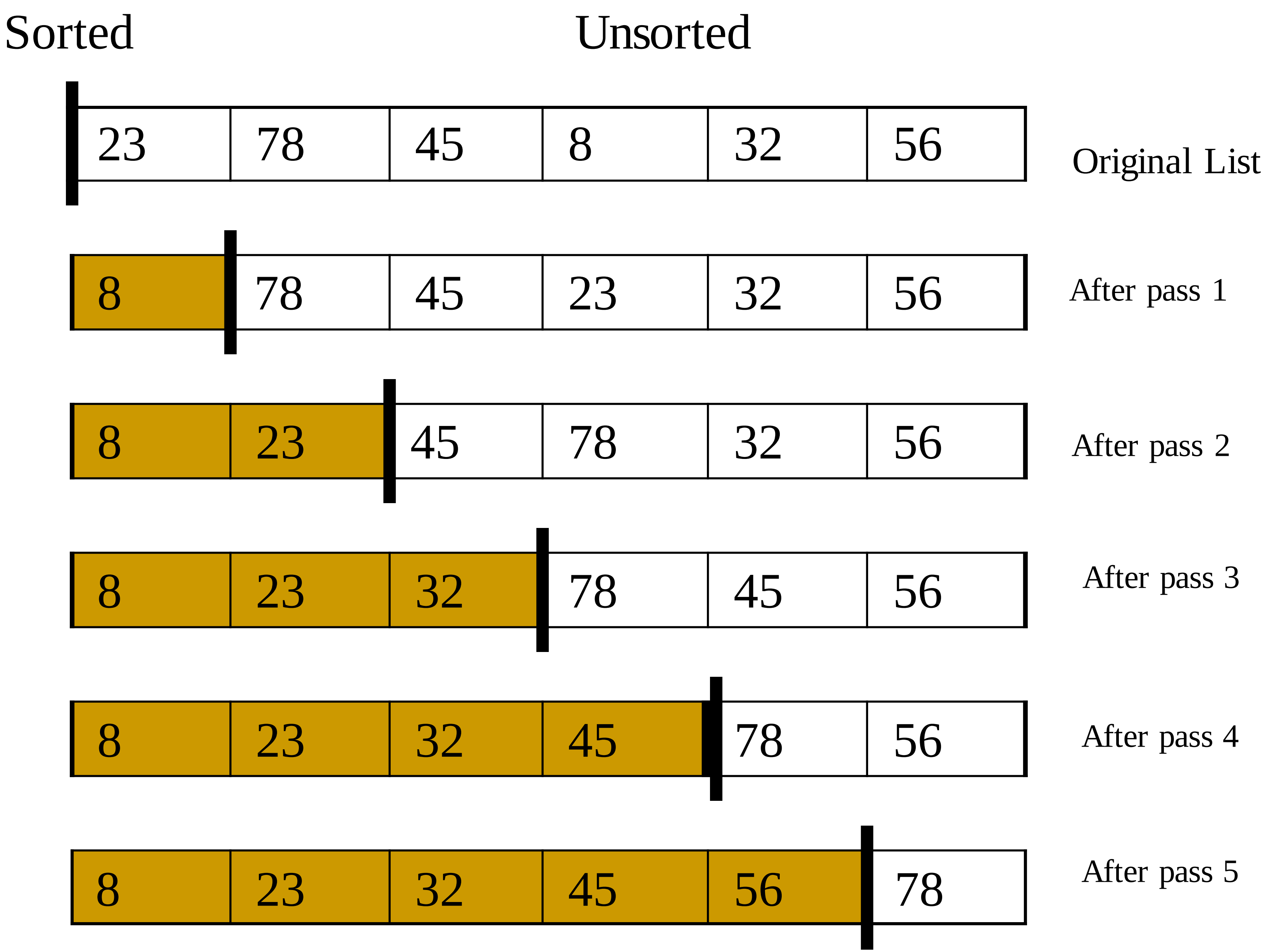
After pass 1

After pass 2

After pass 3

After pass 4

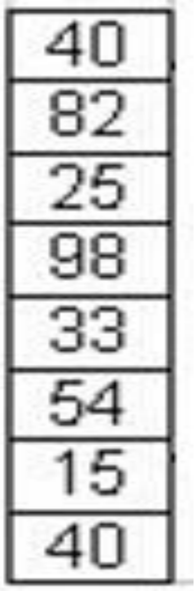
After pass 5



Sorted Unsorted

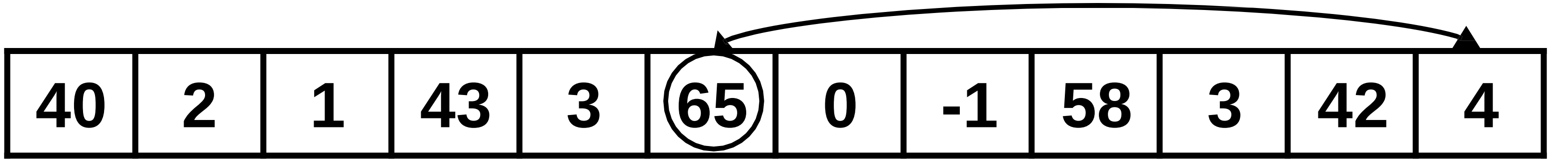
23	78	45	8	32	56	Original List
8	78	45	23	32	56	After pass 1
8	23	45	78	32	56	After pass 2
8	23	32	78	45	56	After pass 3
8	23	32	45	78	56	After pass 4
8	23	32	45	56	78	After pass 5

✓*Perform selection sort : show the scan, selection and swap process.*



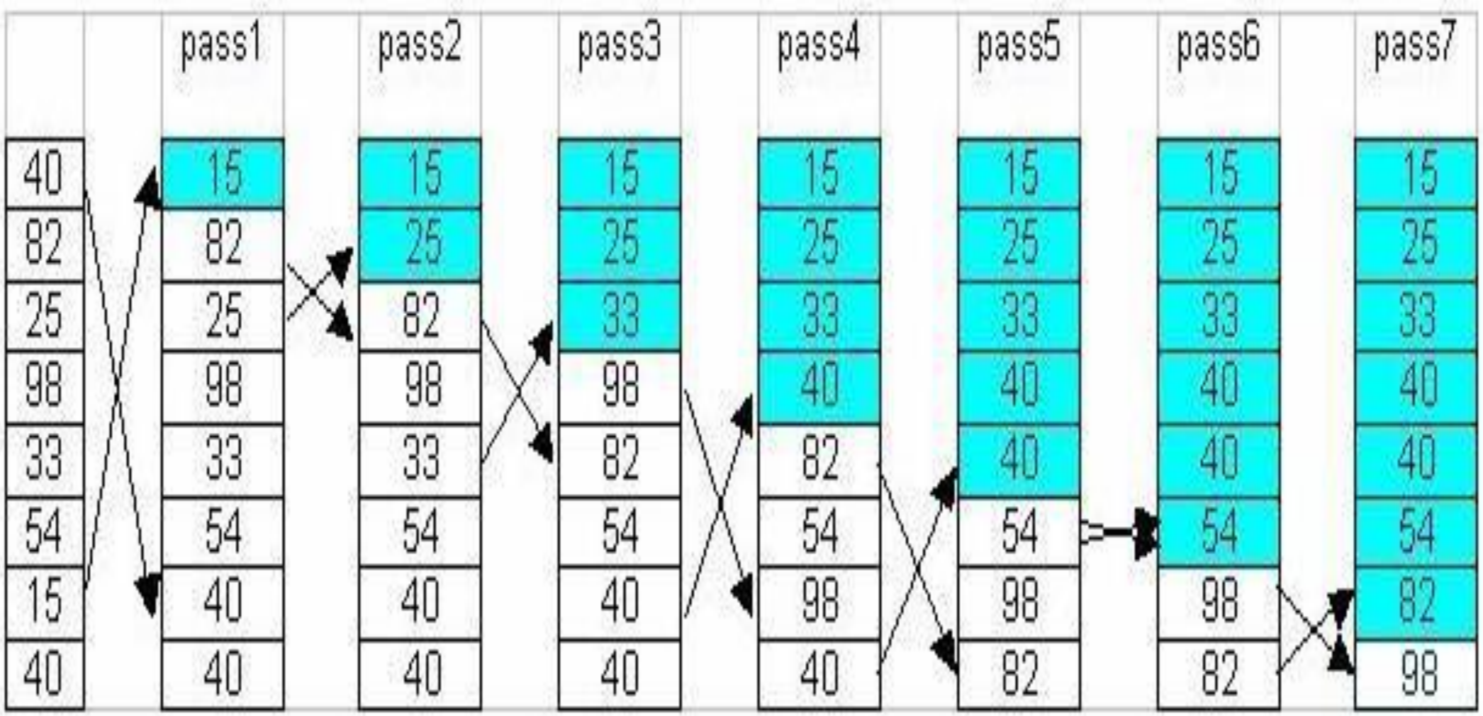
40
82
25
98
33
54
15
40

✓*Perform sort by selecting the largest element and swapping it.*



40	2	1	43	3	65	0	-1	58	3	42	4

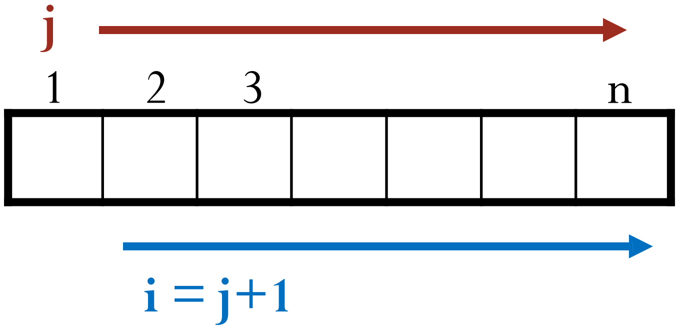
Solution

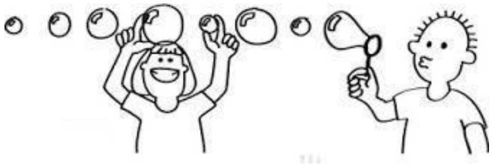


	pass1	pass2	pass3	pass4	pass5	pass6	pass7
40	15	15	15	15	15	15	15
82	82	25	25	25	25	25	25
25	25	82	33	33	33	33	33
98	98	98	98	40	40	40	40
33	33	33	82	82	40	40	40
54	54	54	54	54	54	54	54
15	40	40	40	98	98	98	82
40	40	40	40	40	82	82	98

Selection sort algorithm

```
n ← length of the list A
    Repeat from $j = 1$ to $n$
        let smallest ← $j$
        Repeat from $i = j + 1$ to $n$
            if listA[i] < listA[smallest]
            then smallest ← i
        exchange listA[j] $\boxed{\leftrightarrow}$ listA[smallest]
```





Bubble Sort

- Idea: Bubble in water.
  - Bubble in water moves upward.
  - When a bubble moves upward, the water from above will move downward to fill in the space left by the bubble.

General strategy

- Repeatedly *scan* through the list of elements.
- *Swap* adjacent elements that are out of order.
  - looks at pairs of entries in the list, and swaps their order if needed.

Bubble Sort

9, 6, 2, 12, 11, 9, 3, 7

Bubblesort compares the numbers in pairs from left to right exchanging when necessary. Here the first number is compared to the second and as it is larger they are exchanged.

Now the next pair of numbers are compared. Again the 9 is the larger and so this pair is also exchanged.

In the third comparison, the 9 is not larger than the 12 so no exchange is made. We move on to compare the next pair without any change to the list.

The 12 is larger than the 11 so they are exchanged.

The twelve is greater than the 9 so they are exchanged

The end of the list has been reached so this is the end of the first pass. The twelve at the end of the list must be largest number in the list and so is now in the correct position. We now start a new pass from left to right.

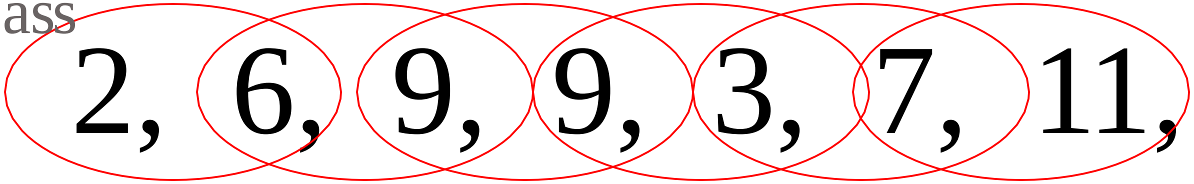
6 2 9 11 9 3 7 12

The 12 is greater than the 7 so they are exchanged.

Bubble Sort Example

First Pass 6, 2, 9, 11, 9, 3, 7, 12

Second Pass



2, 6, 9, 9, 3, 7, 11, 12

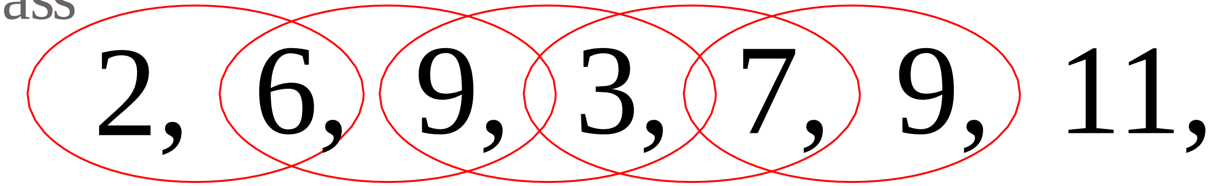
Notice that this time we do not have to compare the last two numbers as we know the 12 is in position. This pass therefore only requires 6 comparisons.

Bubble Sort Example

First Pass 6, 2, 9, 11, 9, 3, 7, 12

Second Pass 2, 6, 9, 9, 3, 7, 11, 12

Third Pass



2, 6, 9, 3, 7, 9, 11, 12

This time the 11 and 12 are in position. This pass therefore only requires 5 comparisons.

Bubble Sort Example

First Pass

6, 2, 9, 11, 9, 3, 7, 12

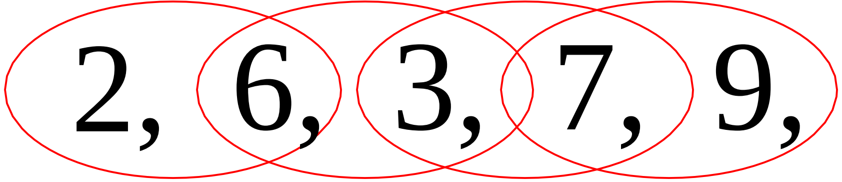
Second Pass

2, 6, 9, 9, 3, 7, 11, 12

Third Pass

2, 6, 9, 3, 7, 9, 11, 12

Fourth Pass



2, 6, 3, 7, 9, 9, 11, 12

Each pass requires fewer comparisons. This time only 4 are needed.

Bubble Sort Example

First Pass

6, 2, 9, 11, 9, 3, 7, 12

Second Pass

2, 6, 9, 9, 3, 7, 11, 12

Third Pass

2, 6, 9, 3, 7, 9, 11, 12

Fourth Pass

2, 6, 3, 7, 9, 9, 11, 12

Fifth Pass

2, 3, 6, 7, 9, 9, 11, 12

The list is now sorted but the algorithm does not know this until it completes a pass with no exchanges.

Bubble Sort Example

First Pass

6, 2, 9, 11, 9, 3, 7, 12

Second Pass

2, 6, 9, 9, 3, 7, 11, 12

Third Pass

2, 6, 9, 3, 7, 9, 11, 12

Fourth Pass

2, 6, 3, 7, 9, 9, 11, 12

Fifth Pa

This pass no exchanges are made so the algorithm knows the list is sorted. It can therefore save time by not doing the final pass. With other lists this check could save much more work.

Sixth Pass

2, 3, 6, 7, 9, 9, 11, 12

Bubble sort compares the numbers in pairs from left to right exchanging when necessary.

Eg: Bubble Sort

9, 6, 2, 12, 11, 9, 3, 7

✓ *The first number is compared to the second and as it is larger they are exchanged.*

6, **9**, 2, 12, 11, 9, 3, 7

✓ *Now the next pair of numbers are compared. Again the 9 is the larger and so this pair is also exchanged.*

6, **2**, **9**, 12, 11, 9, 3, 7

✓ *In the third comparison, the 9 is not larger than the 12 so no exchange is made.*

✓ *We move on to compare the next pair without any change to the list.*

Bubble Sort *contd---*

6, 2, 9, 12, 11, 9, 3, 7

✓ *The 12 is larger than the 11 so they are exchanged.*

6, 2, 9, **11**, **12**, 9, 3, 7

✓ *The 12 is greater than the 9 so they are exchanged.*

6, 2, 9, 11, **9**, **12**, 3, 7

✓ *The 12 is greater than the 3 so they are exchanged.*

6, 2, 9, 11, 9, **3**, **12**, 7

✓ *The 12 is greater than the 7 so they are exchanged.*

Bubble Sort *contd---*

6, 2, 9, 11, 9, 3, **7**, **12**

✓The end of the list has been reached so this is the end of the first pass.

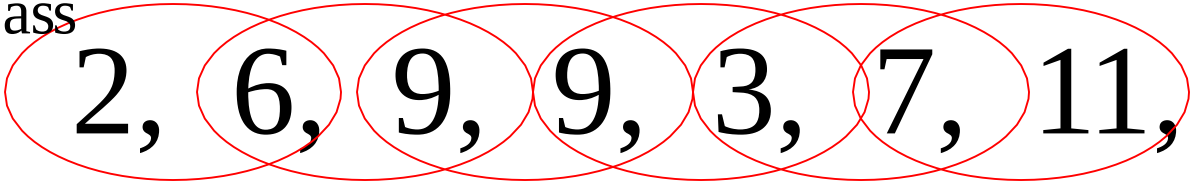
✓The 12 at the end of the list must be largest number in the list and so is now in the correct position.

✓ ***We now start a new pass from left to right.***

Bubble Sort : Second pass

First Pass 6, 2, 9, 11, 9, 3, 7, 12

Second Pass



2, 6, 9, 9, 3, 7, 11, 12

- ✓ *Notice that this time we do not have to compare the last two numbers as we know the 12 is in position.*
- ✓ *This pass therefore only requires 6 comparisons.*

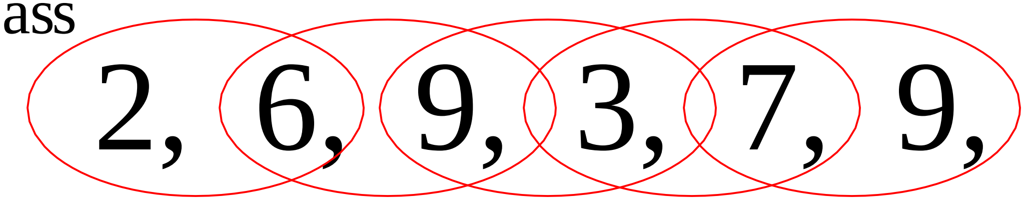
Bubble Sort :Third pass

First Pass 6, 2, 9, 11, 9, 3, 7, 12

Second Pass

2, 6, 9, 9, 3, 7, 11, 12

Third Pass



2, 6, 9, 3, 7, 9, 11, 12

- ✓ *This time the 11 and 12 are in position.*
- ✓ *This pass therefore only requires 5 comparisons.*

Bubble Sort : Fourth pass

First Pass 6, 2, 9, 11, 9, 3, 7, 12

Second Pass 2, 6, 9, 9, 3, 7, 11, 12

Third Pass 2, 6, 9, 3, 7, 9, 11, 12

Fourth Pass 2, 6, 3, 7, 9, 9, 11, 12

✓ *Each pass requires fewer comparisons.*

✓ *This time only 4 are needed.*

Bubble Sort : Fifth pass

First Pass

6, 2, 9, 11, 9, 3, 7, 12

Second Pass

2, 6, 9, 9, 3, 7, 11, 12

Third Pass

2, 6, 9, 3, 7, 9, 11, 12

Fourth Pass

2, 6, 3, 7, 9, 9, 11, 12

Fifth Pass

2, 3, 6, 7, 9, 9, 11, 12

✓ *The list is now sorted but the algorithm does not know this until it completes a pass with no exchanges.*

Bubble Sort : Sixth pass

Fifth Pass 2, 3, 6, 7, 9, 9, 11, 12

Sixth Pass

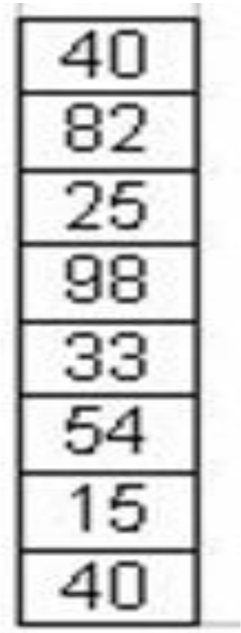
2, 3, 6, 7, 9, 9, 11, 12

✓*This pass no exchanges are made so the algorithm knows the list is sorted.*

✓*It can therefore save time by not doing the final pass.With other lists this check could save much more work.*

✓ *Perform bubble sort : show the scan and swap.*

List #1

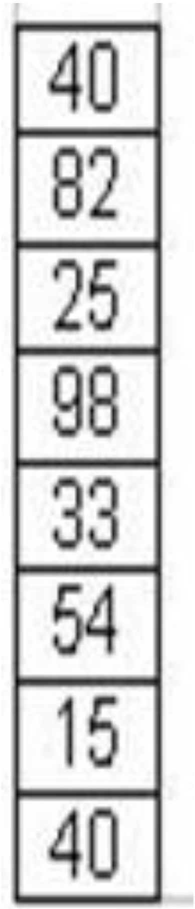


40
82
25
98
33
54
15
40

List #2

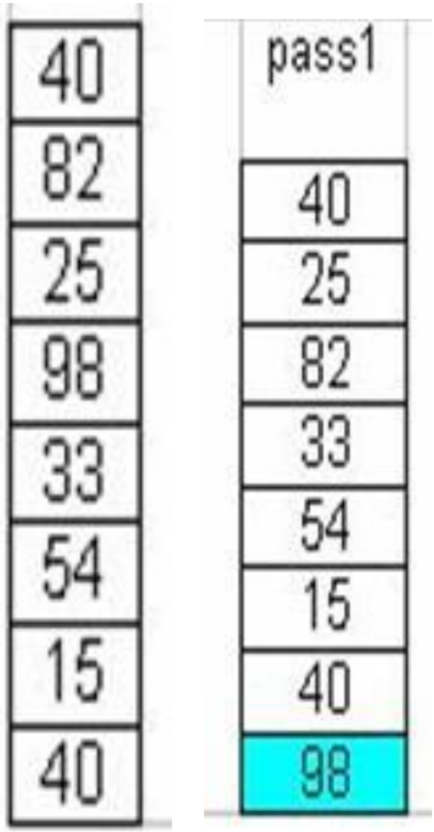
40	2	1	43	3	65	0	-1	58	3	42	4

Solution



40
82
25
98
33
54
15
40

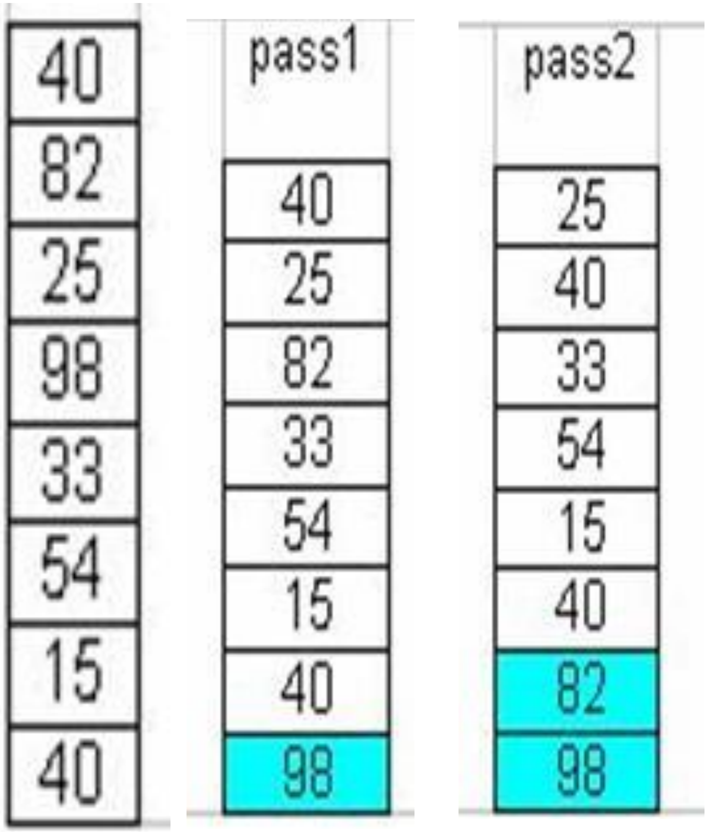
Pass 1



	pass1
40	40
82	25
25	82
98	33
33	54
54	15
15	40
40	98

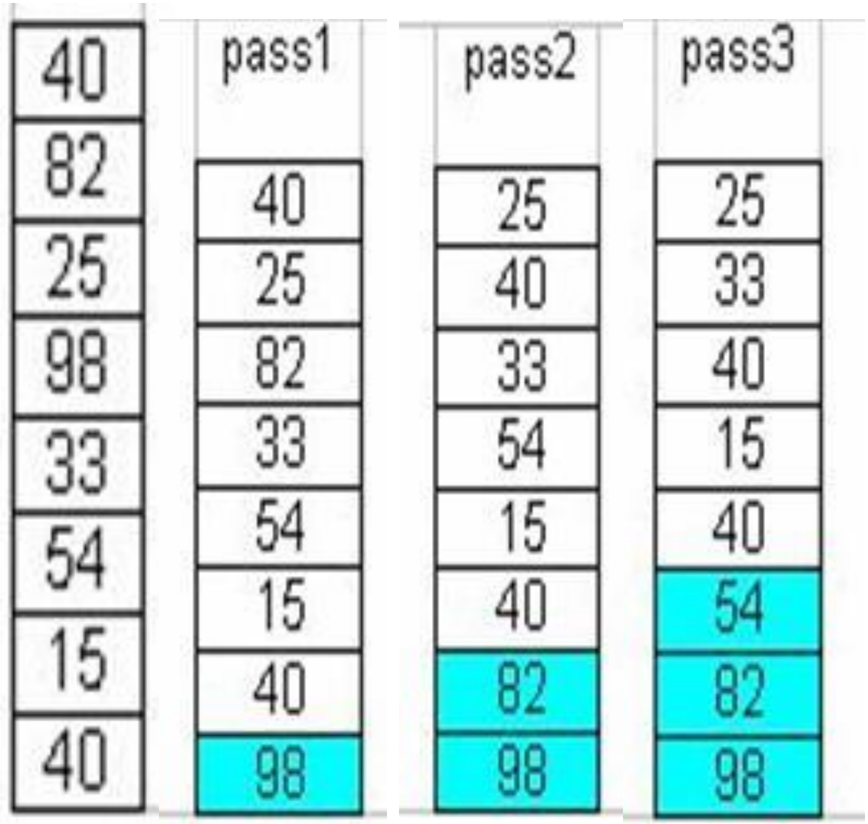
http://www.powershow.com/view/158a13-ZjNmN/ICS103_Programming_in_C_Lecture_14_Searching_and_Sorting_powerpoint_ppt_presentation

Pass 2



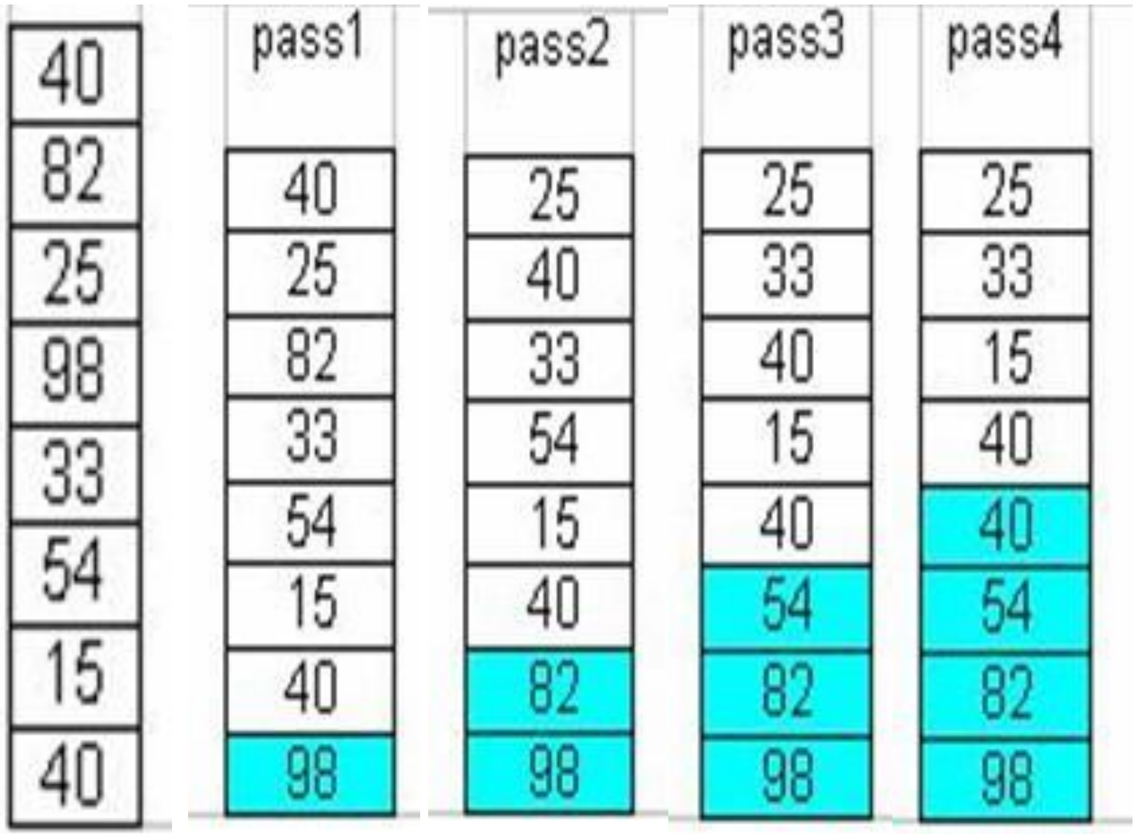
	pass1	pass2
40	40	25
82	25	40
25	82	33
98	33	54
33	54	15
54	15	40
15	40	82
40	98	98

Pass 3



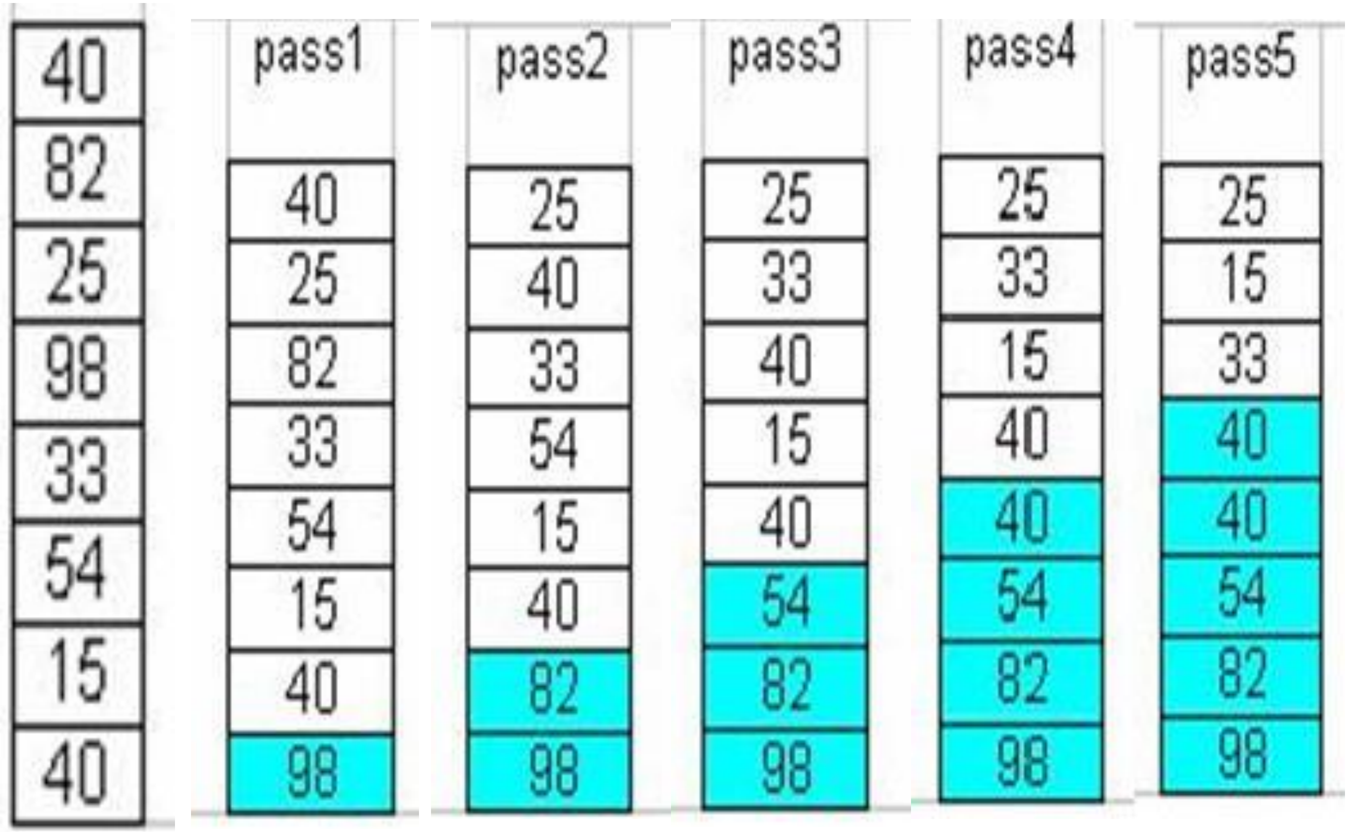
	pass1	pass2	pass3
40	40	25	25
82	25	40	33
25	82	33	40
98	33	54	15
33	54	15	40
54	15	40	54
15	40	82	82
40	98	98	98

Pass 4



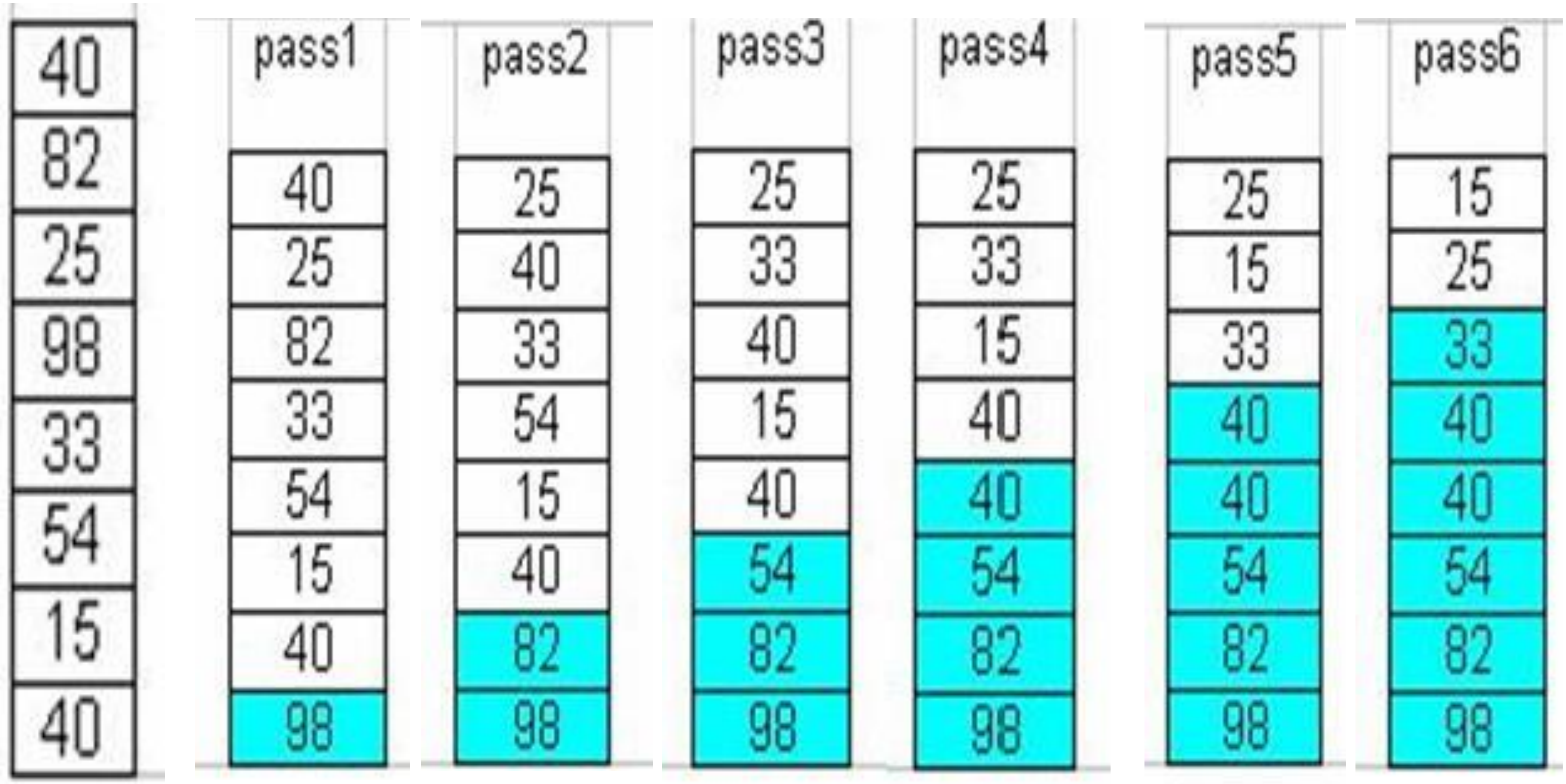
	pass1	pass2	pass3	pass4
40	40	25	25	25
82	25	40	33	33
25	82	33	40	15
98	33	54	15	40
33	54	15	40	40
54	15	40	54	54
15	40	82	82	82
40	98	98	98	98

Pass 5



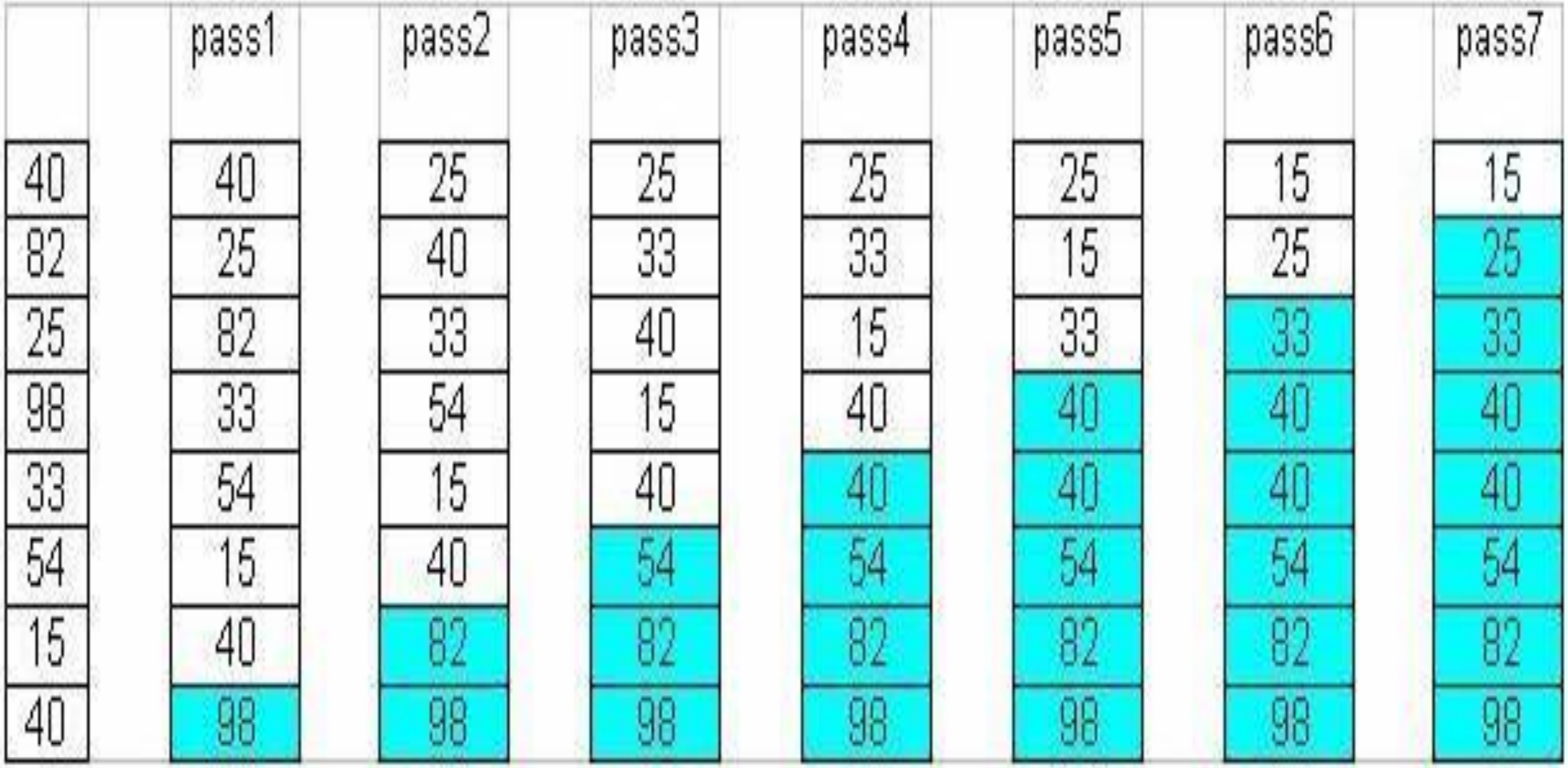
	pass1	pass2	pass3	pass4	pass5
40	40	25	25	25	25
82	25	40	33	33	15
25	82	33	40	15	33
98	33	54	15	40	40
33	54	15	40	40	40
54	15	40	54	54	54
15	40	82	82	82	82
40	98	98	98	98	98

Pass 6



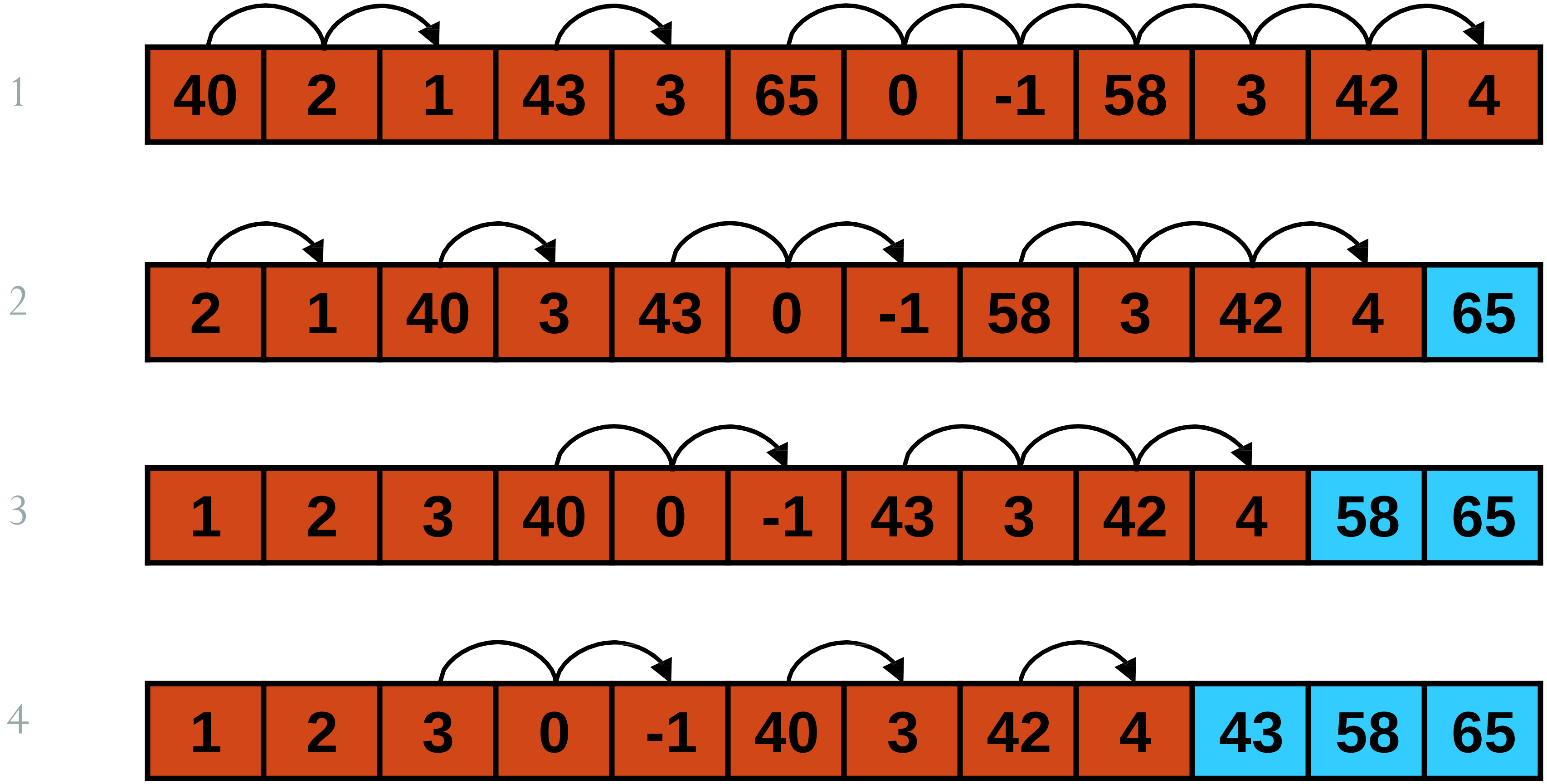
	pass1	pass2	pass3	pass4	pass5	pass6
40	40	25	25	25	25	15
82	25	40	33	33	15	25
25	82	33	40	15	33	33
98	33	54	15	40	40	40
33	54	15	40	40	40	40
54	15	40	54	54	54	54
15	40	82	82	82	82	82
40	98	98	98	98	98	98

Pass 7



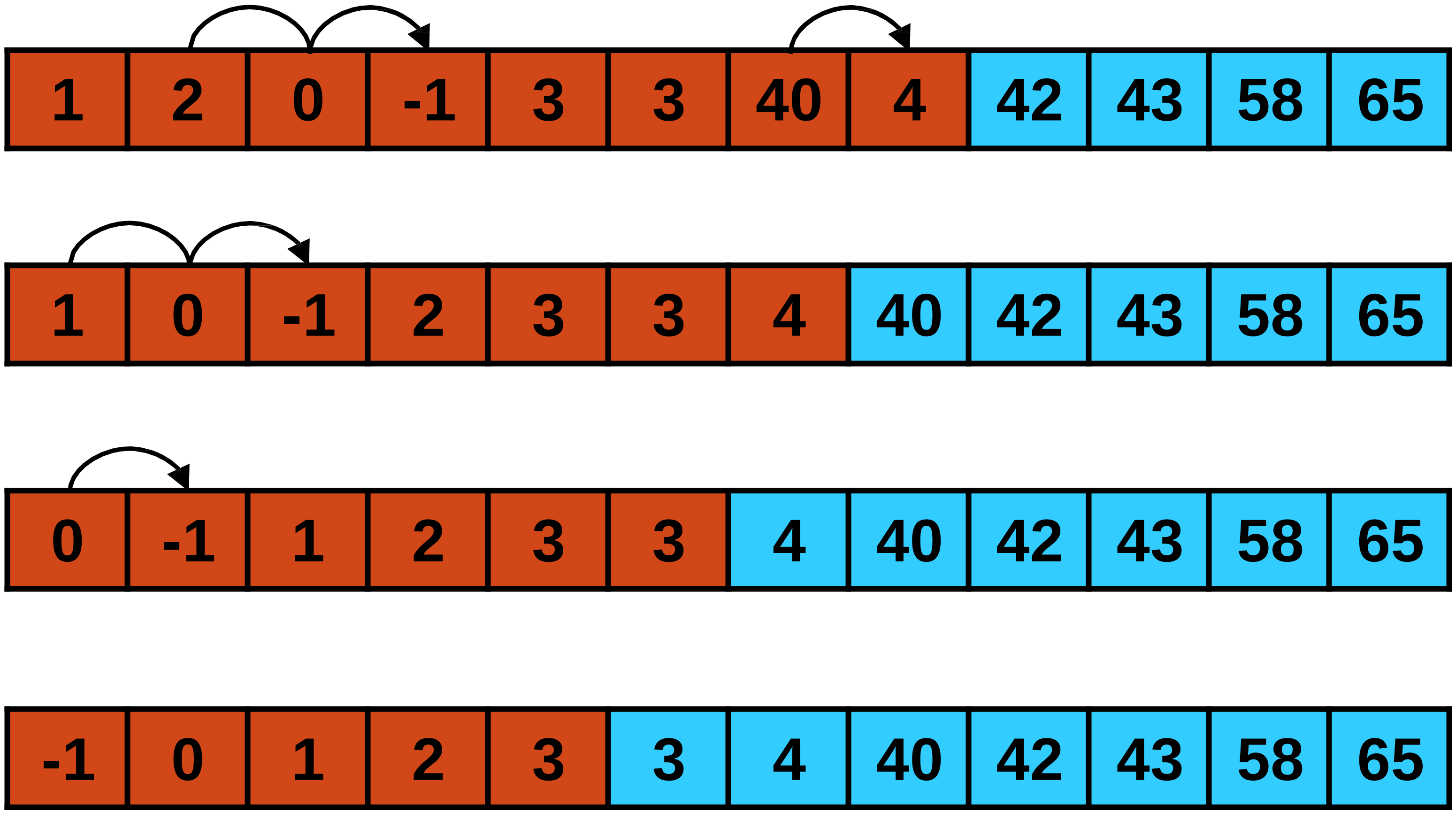
	pass1	pass2	pass3	pass4	pass5	pass6	pass7
40	40	25	25	25	25	15	15
82	25	40	33	33	15	25	25
25	82	33	40	15	33	33	33
98	33	54	15	40	40	40	40
33	54	15	40	40	40	40	40
54	15	40	54	54	54	54	54
15	40	82	82	82	82	82	82
40	98	98	98	98	98	98	98

Solution



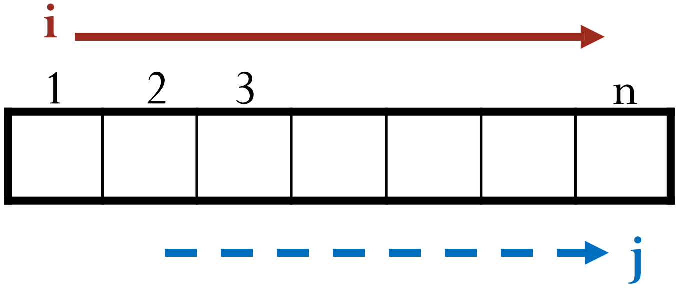
1	40	2	1	43	3	65	0	-1	58	3	42	4
2	2	1	40	3	43	0	-1	58	3	42	4	65
3	1	2	3	40	0	-1	43	3	42	4	58	65
4	1	2	3	0	-1	40	3	42	4	43	58	65

Solution *contd* ---



5	1	2	0	-1	3	3	40	4	42	43	58	65
6	1	0	-1	2	3	3	4	40	42	43	58	65
7	0	-1	1	2	3	3	4	40	42	43	58	65
8	-1	0	1	2	3	3	4	40	42	43	58	65

Bubble sort algorithm



```
n ← length of list A
    Repeat from i = 1 to n-1
        Repeat from j = 1 to (n-i)
            if  list A[j] > list A[j +1]
                then exchange A[j] ↔ A[j+1]
```

It's gonna get harder before it gets easier. But it will get better, you just gotta make it through the hard stuff first

Case study time …..

SORTING

How do you sort when you can only compare two things?

Sort

- Part One:
  - –Find the lightest bottle using the scale
  - –What is the easiest way to do this?
- PartTwo:
  - –Choose three bottles at random
  - –Sort them using the scale
  - –What is the fewest number of comparisons you can do?

Sort

- PartThree:
  - –Sort all the bottles from lightest to heaviest by repeatedly finding the lightest bottle remaining, and moving it into the sorted list at the top
  - –This is known as **Selection Sort**

Finding the Smallest Item

given: a list of data to find the minimum in

```
set currentSmallestItem to first item in list
while we haven't looked at all items in list:
    set compareItem to next item in list
    if compareItem < currentSmallestItem:
            save compareItem to currentSmallestItem
```

Selection Sort

```
given: a list of data to sort
set listSorted to an empty list
set listRemaining to the original list to be
sorted  while listRemaining is not empty:
   set smallestValue to the min value in listRemaining
   add smallestValue to the end of listSorted
   remove smallestValue from listRemaining
```

Insertion Sort

- Take the leftmost bottle from the bottom
- Insert it into the correct location in the sorted list
- –Do this by comparing the new item with each of the existing items (usually starting at the end) until the correct place is found
- This is known as **Insertion Sort**

Insertion Sort

```
given: a list of data to sort
set listSorted to an empty list
set listRemaining to the original list to be sorted
while listRemaining is not empty:
    set nextValueToSort to the first item in listRemaining
    add nextValueToSort to the end of listSorted
    while (nextValueToSort) <= (value to its left):
            swap nextValueToSort with its neighbour to the left
```

What has been described?

- The need for re-arrangement of a collection of data.
- Selection sorting technique.
- Bubble sorting technique.

Credits

- *112.196.5.130/edusat/engg/CompEngg/…/EDUSAT%20Lect.ppt*
- *Analysis of Algorithms, CS 477/677, George Bebis*
- *http://www.powershow.com/view/158a13-ZjNmN/ICS103_Programming_in_C_Lecture_14_Searching_and_Sorting_powerpoint_ppt_presentation*
- *https://www.coursehero.com/file/6355834/sorting/*
- *CENG 213 Data Structures, METU, CSE*
- *ww.cs.bu.edu/fac/gkollios/cs113/Slides/**Sorting**1.ppt*
- *Google images*