Gebze Technical University

CSE222 – Data Structures and Algorithms

HW6 – Part1

Subject = Sorting Step by Step

Nevzat Seferoglu

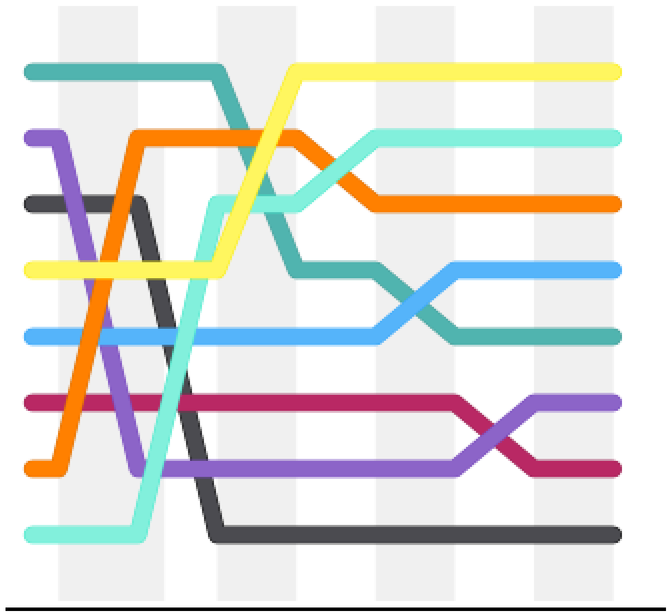
171044024

Shell Sort :

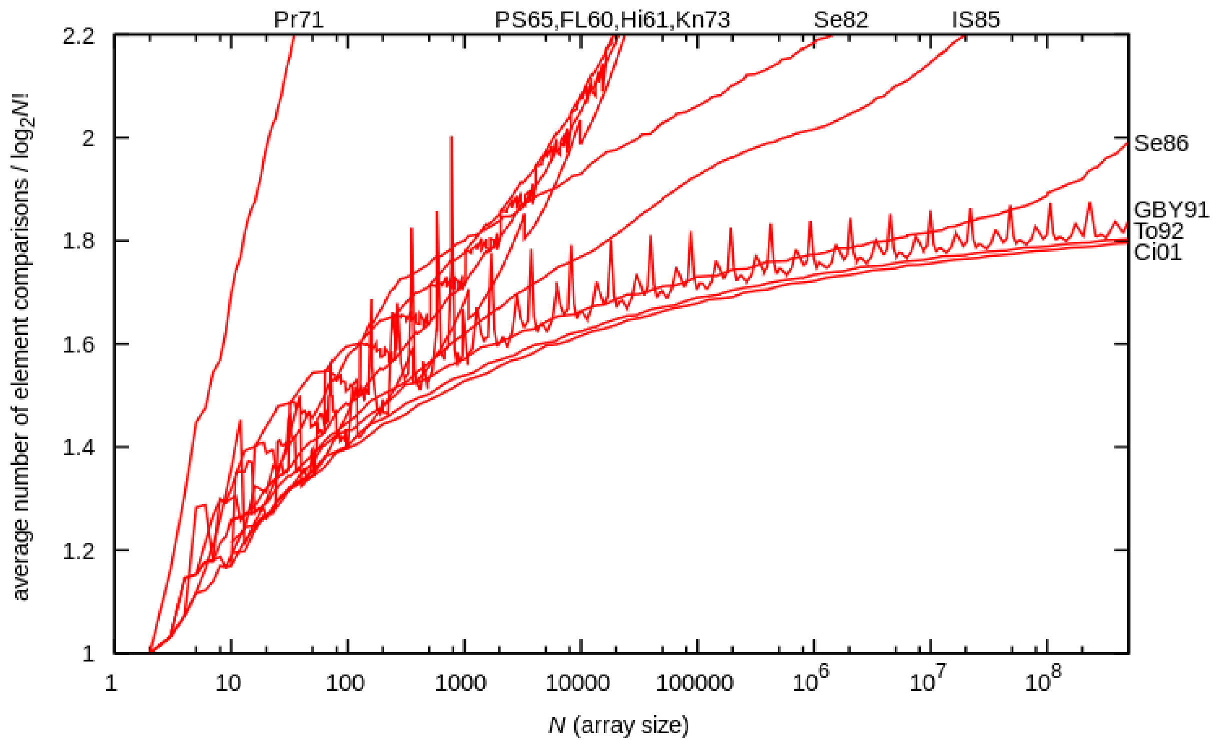
***Note :** Shell-sort use insertion sort as interval operation and insertion sort uses swap operation so , swapping the element admitted displacement amount.*

Description :

ShellSort is mainly a variation of insertion sort. In insertion sort, we move elements only one position ahead. When an element has to be moved far ahead, many movements are involved. The idea of shellSort is to allow exchange of far items. In shellSort, we make the array h-sorted for a large value of h. We keep reducing the value of h until it becomes 1. An array is said to be h-sorted if all sublists of every h'th element is sorted.



*Swapping pairs of items in successive steps of Shellsort with gaps 5, 3, 1



A =

1	2	3	4	5	6	7	8	9	10
0	1	2	3	4	5	6	7	8	9

gap = (initial gap)/2 = **5**

A[0] and A[5] are sorted. (best case)

A[1] and A[6] are sorted. (best case)

A[2] and A[7] are sorted. (best case)

A[3] and A[8] are sorted. (best case)

A[4] and A[9] are sorted. (best case)

1	2	3	4	5	6	7	8	9	10
0	1	2	3	4	5	6	7	8	9

gap = (gap) / 2 = **2**

A[0] , A[2] , A[4] , A[6] , A[8] are sorted. (best case)

A[1] , A[3] , A[5] , A[7] , A[9] are sorted.

1	2	3	4	5	6	7	8	9	10
0	1	2	3	4	5	6	7	8	9

gap = (gap) / 2 = **1**

A[0] , A[1] , A[2] , A[3] , A[4] , A[5] , A[6] , A[7] , A[8] , A[9] (best case) **C1** = 10 **, D1** = 0

***Calculated with computer.**

+ Total **comparison** amount = **22**

+ Total **displacement** amount = **0**

B =

10	9	8	7	6	5	4	3	2	1
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

gap = (initial gap)/2 = **5**

B[0] and B[5] are sorted. (worst case)

B [1] and B [6] are sorted. (worst case)

B [2] and B [7] are sorted. (worst case)

B [3] and B [8] are sorted. (worst case)

B [4] and B [9] are sorted. (worst case)

1	2	3	4	5	6	7	8	9	10
0	1	2	3	4	5	6	7	8	9

gap = (gap) / 2 = **2**

B [0] , B [2] , B [4] , B [6] , B [8] are sorted. (worst case)

B [1] , B [3] , B [5] , B [7] , B [9] are sorted. (worst case)

1	2	3	4	5	6	7	8	9	10
0	1	2	3	4	5	6	7	8	9

gap = (gap) / 2 = **1**

B [0] , B [1] , B [2] , B [3] , B [4] , B [5] , B [6] , B [7] , B [8] , B [9] (best case) **C1** = 10 **, D1** = 0

***Calculated with computer.**

+ Total **comparison** amount = **26**

+ Total **displacement** amount = **13**

C =

5	2	13	9	1	7	6	8	1	15	4	11
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**

gap = (initial gap)/2 = **6**

C [0] and C [6] are sorted.

C [1] and C [7] are sorted.

C [2] and C [8] are sorted.

C [3] and C [9] are sorted.

C [4] and C [10] are sorted.

C [5] and C [11] are sorted.

C =

5	1	1	6	2	7	9	4	11	15	8	13
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**

gap = (last gap) / 2 = **3**

C[0] , C[3] , C[6] , C[9] are sorted.

C[1] , C[4] , C[7] , C[10] are sorted.

C[2] , C[5] , C[8] , C[11] are sorted.

C =

5	2	1	9	1	7	6	8	13	15	4	11
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**

gap = (last gap) / 2 = **1**

C[0] , C[1] , C[2], C[3], C[4], C[5], C[6], C[7], C[8], C[9], C[10], C[11] are sorted.

***Calculated with computer.**

+ Total **comparison** amount = **40**

+ Total **displacement** amount = **17**

C =

83	66	73	77	72	81	67	76	82	69	80	75
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**

gap = (initial gap)/2 = **6**

C [0] and C [6] are sorted.

C [1] and C [7] are sorted.

C [2] and C [8] are sorted.

C [3] and C [9] are sorted.

C [4] and C [10] are sorted.

C [5] and C [11] are sorted.

C =

67	66	73	69	72	75	83	76	82	77	80	81
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**

gap = (last gap) / 2 = **3**

C[0] , C[3] , C[6] , C[9] are sorted.

C[1] , C[4] , C[7] , C[10] are sorted.

C[2] , C[5] , C[8] , C[11] are sorted.

C =

67	66	73	69	72	75	77	76	81	83	80	82
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**

gap = (last gap) / 2 = **1**

C[0] , C[1] , C[2], C[3], C[4], C[5], C[6], C[7], C[8], C[9], C[10], C[11] are sorted.

***Calculated with computer.**

+ Total **comparison** amount = 34

+ Total **displacement** amount = 12

Merge Sort :

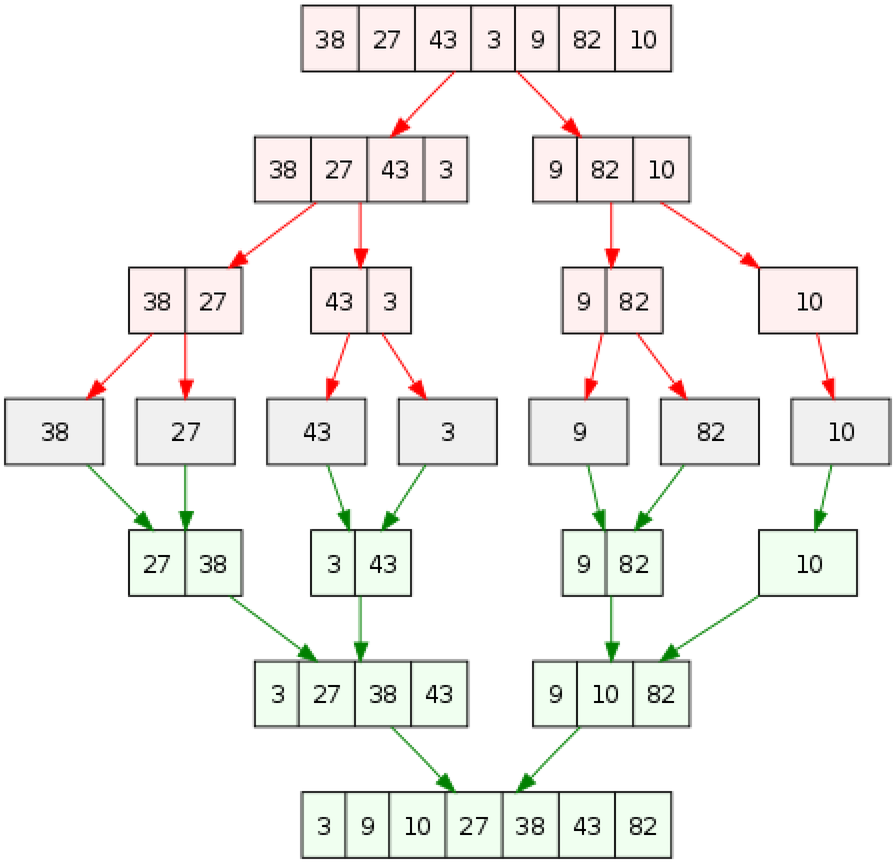
***Note :** Merge-sort does not use swap operation so , putting the element to to output array admitted displacement amount.*

Description :

Merge Sort is a divide and conquer. It divides input array in two halves, calls itself for the two halves and then merges the two sorted halves. **The merge() function** is used for merging two halves. The merge(arr, l, m, r) is key process that assumes that arr[l..m] and arr[m+1..r] are sorted and merges the two sorted sub-arrays into one. See following C implementation for details.

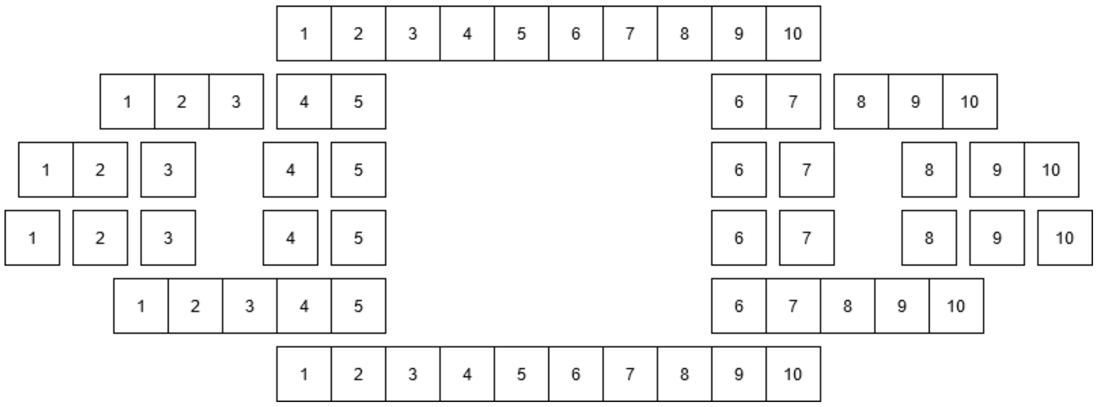
```
MergeSort(arr[], l,  r)
If r > l
     1. Find the middle point to divide the array into
two halves:
             middle m = (l+r)/2
     2. Call mergeSort for first half:
             Call mergeSort(arr, l, m)
     3. Call mergeSort for second half:
             Call mergeSort(arr, m+1, r)
     4. Merge the two halves sorted in step 2 and 3:
             Call merge(arr, l, m, r)
```

The following diagram from Wikipedia shows the complete merge sort process for an example array {38, 27, 43, 3, 9, 82, 10}. If we take a closer look at the diagram, we can see that the array is recursively divided in two halves till the size becomes 1. Once the size becomes 1, the merge processes comes into action and starts merging arrays back till the complete array is merged.


38 27 43 3 9 82 10
38 27 43 3
9 82 10
38 27
43 3
9 82
10
38
27
43
3
9
82
10
27 38
3 43
9 82
10
3 27 38 43
9 10 82
3 9 10 27 38 43 82

Merge sort type algorithms allowed large data sets to be sorted on early computers that had small random-access memories by modern standards. Records were stored on magnetic-tape and processed on banks of magnetic tape drives, such as these IBM 729s.

A :



Given array : 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10

Left Table : 1

Right Table : 2

1 compared with 2

1 Putted output array earlier

2 is remaining from the right and putted output array

Merged list : 1 2

Left Table : 3

Right Table : 4

3 compared with 4

3 Putted output array earlier

4 is remaining from the right and putted output array

Merged list : 3 4

```
Left Table : 1 2
Right Table : 3 4
1 compared with 3
1 Putted output array earlier
2 compared with 3
2 Putted output array earlier

3 is remaining from the right and putted output array
4 is remaining from the right and putted output array
Merged list : 1 2 3 4

Left Table : 5
Right Table : 6
5 compared with 6
5 Putted output array earlier

6 is remaining from the right and putted output array
Merged list : 5 6

Left Table : 8
Right Table : 10
8 compared with 10
8 Putted output array earlier

10 is remaining from the right and putted output array
Merged list : 8 10

Left Table : 7
```

```
Right Table : 8 10
7 compared with 8
7 Putted output array earlier

8 is remaining from the right and putted output array
10 is remaining from the right and putted output array
Merged list : 7 8 10

Left Table : 5 6
Right Table : 7 8 10
5 compared with 7
5 Putted output array earlier
6 compared with 7
6 Putted output array earlier

7 is remaining from the right and putted output array
8 is remaining from the right and putted output array
10 is remaining from the right and putted output array
Merged list : 5 6 7 8 10

Left Table : 1 2 3 4
Right Table : 5 6 7 8 10
1 compared with 5
1 Putted output array earlier
2 compared with 5
2 Putted output array earlier
3 compared with 5
3 Putted output array earlier
4 compared with 5
```

4 Putted output array earlier

5 is remaining from the right and putted output array

6 is remaining from the right and putted output array

7 is remaining from the right and putted output array

8 is remaining from the right and putted output array

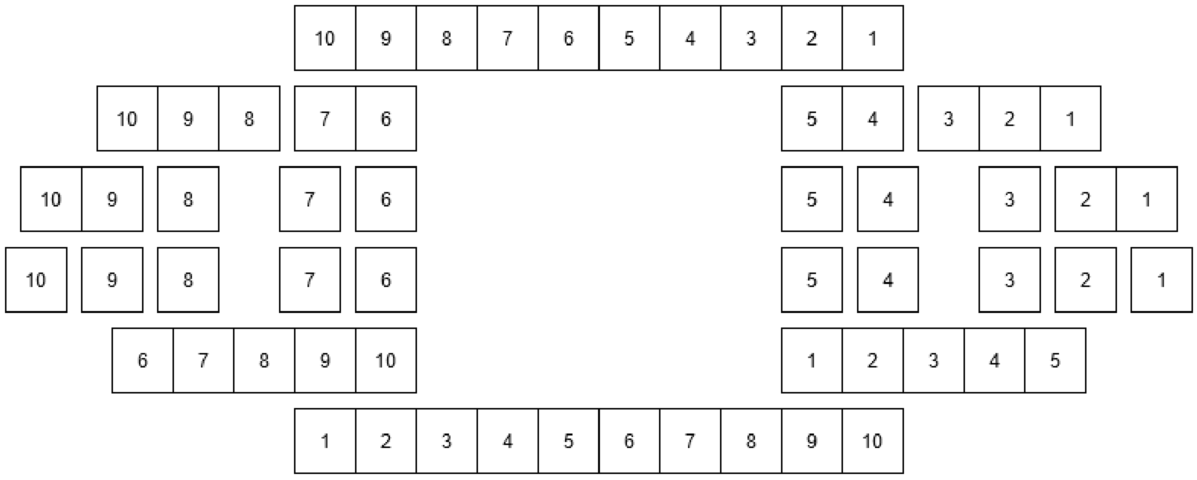
10 is remaining from the right and putted output array

Merged list : 1 2 3 4 5 6 7 8 10

+ Total **comparison** amount = 13

+ Total **displacement** amount = 29

B :



Given array : 10 , 9 ,8 ,7 6, 5, 4, 3, 2, 1

Left Table : 10

Right Table : 9

10 compared with 9

9 Putted output array earlier

10 is remaining from the left and putted output array

Merged list : 9 10

Left Table : 7

Right Table : 6

7 compared with 6

6 Putted output array earlier

7 is remaining from the left and putted output array

Merged list : 6 7

Left Table : 8

Right Table : 6 7

8 compared with 6

6 Putted output array earlier

8 compared with 7

7 Putted output array earlier

8 is remaining from the left and putted output array

Merged list : 6 7 8

Left Table : 9 10

Right Table : 6 7 8

```
9 compared with 6

6 Putted output array earlier

9 compared with 7

7 Putted output array earlier

9 compared with 8

8 Putted output array earlier


9 is remaining from the left and putted output array

10 is remaining from the left and putted output array

Merged list : 6 7 8 9 10


Left Table : 5

Right Table : 4

5 compared with 4

4 Putted output array earlier


5 is remaining from the left and putted output array

Merged list : 4 5


Left Table : 2

Right Table : 1

2 compared with 1

1 Putted output array earlier


2 is remaining from the left and putted output array

Merged list : 1 2


Left Table : 3

Right Table : 1 2
```

```
3 compared with 1
1 Putted output array earlier
3 compared with 2
2 Putted output array earlier

3 is remaining from the left and putted output array
Merged list : 1 2 3

Left Table : 4 5
Right Table : 1 2 3
4 compared with 1
1 Putted output array earlier
4 compared with 2
2 Putted output array earlier
4 compared with 3
3 Putted output array earlier

4 is remaining from the left and putted output array
5 is remaining from the left and putted output array
Merged list : 1 2 3 4 5

Left Table : 6 7 8 9 10
Right Table : 1 2 3 4 5
6 compared with 1
1 Putted output array earlier
6 compared with 2
2 Putted output array earlier
6 compared with 3
3 Putted output array earlier
```

6 compared with 4

4 Putted output array earlier

6 compared with 5

5 Putted output array earlier

6 is remaining from the left and putted output array

7 is remaining from the left and putted output array

8 is remaining from the left and putted output array

9 is remaining from the left and putted output array

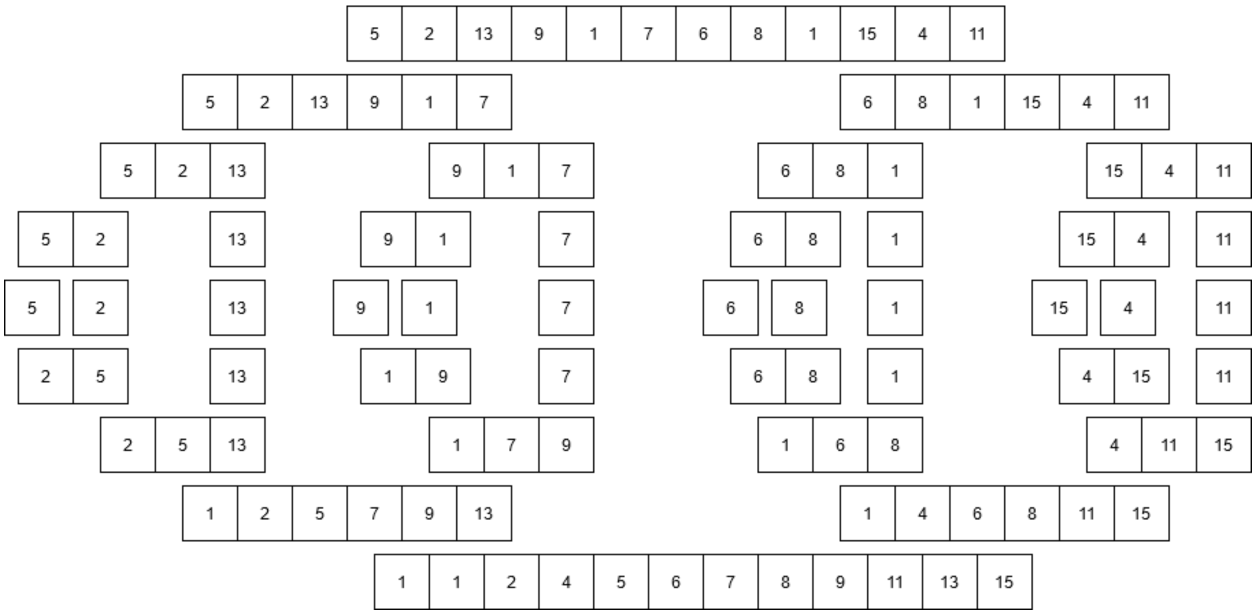
10 is remaining from the left and putted output array

Merged list : 1 2 3 4 5 6 7 8 9 10

+ Total **comparison** amount = 19

+ Total **displacement** amount = 34

C:



Given array : 5, 2, 13, 9, 1, 7, 6, 8, 1, 15, 4, 11

Left Table : 2

```
Right Table : 13

2 compared with 13

2 Putted output array earlier


13 is remaining from the right and putted output array

Merged list : 2 13


Left Table : 5

Right Table : 2 13

5 compared with 2

2 Putted output array earlier

5 compared with 13

5 Putted output array earlier


13 is remaining from the right and putted output array

Merged list : 2 5 13


Left Table : 1

Right Table : 7

1 compared with 7

1 Putted output array earlier


7 is remaining from the right and putted output array

Merged list : 1 7


Left Table : 9

Right Table : 1 7

9 compared with 1

1 Putted output array earlier
```

```
9 compared with 7
7 Putted output array earlier

9 is remaining from the left and putted output array
Merged list : 1 7 9

Left Table : 2 5 13
Right Table : 1 7 9
2 compared with 1
1 Putted output array earlier
2 compared with 7
2 Putted output array earlier
5 compared with 7
5 Putted output array earlier
13 compared with 7
7 Putted output array earlier
13 compared with 9
9 Putted output array earlier

13 is remaining from the left and putted output array
Merged list : 1 2 5 7 9 13

Left Table : 8
Right Table : 1
8 compared with 1
1 Putted output array earlier

8 is remaining from the left and putted output array
Merged list : 1 8
```

Left Table : 6

Right Table : 1 8

6 compared with 1

1 Putted output array earlier

6 compared with 8

6 Putted output array earlier

8 is remaining from the right and putted output array

Merged list : 1 6 8

Left Table : 4

Right Table : 11

4 compared with 11

4 Putted output array earlier

11 is remaining from the right and putted output array

Merged list : 4 11

Left Table : 15

Right Table : 4 11

15 compared with 4

4 Putted output array earlier

15 compared with 11

11 Putted output array earlier

15 is remaining from the left and putted output array

Merged list : 4 11 15

```
Left Table : 1 6 8

Right Table : 4 11 15

1 compared with 4

1 Putted output array earlier

6 compared with 4

4 Putted output array earlier

6 compared with 11

6 Putted output array earlier

8 compared with 11

8 Putted output array earlier


11 is remaining from the right and putted output array

15 is remaining from the right and putted output array

Merged list : 1 4 6 8 11 15


Left Table : 1 2 5 7 9 13

Right Table : 1 4 6 8 11 15

1 compared with 1

1 Putted output array earlier

1 compared with 4

1 Putted output array earlier

2 compared with 4

2 Putted output array earlier

5 compared with 4

4 Putted output array earlier

5 compared with 6

5 Putted output array earlier

7 compared with 6

6 Putted output array earlier
```

7 compared with 8

7 Putted output array earlier

9 compared with 8

8 Putted output array earlier

9 compared with 11

9 Putted output array earlier

13 compared with 11

11 Putted output array earlier

13 compared with 15

13 Putted output array earlier

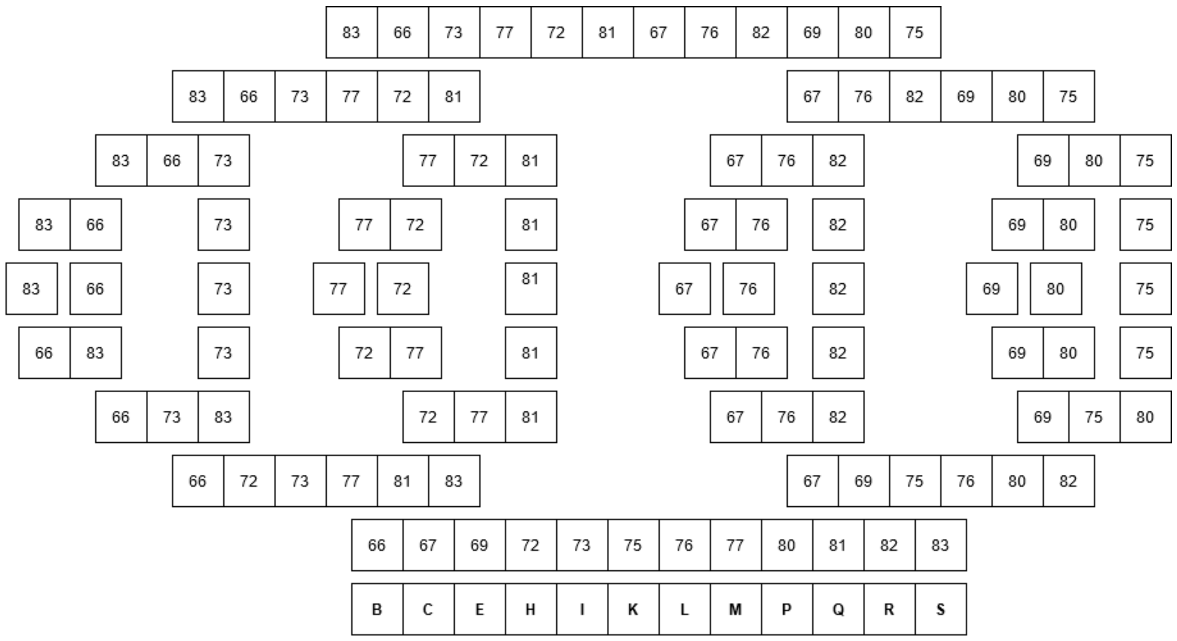
15 is remaining from the right and putted output array

Merged list : 1 1 2 4 5 6 7 8 9 11 13 15

+ Total **comparison** amount = 32

+ Total **displacement** amount = 44

D:



Given Array = ‘S’, ‘B’, ‘I’, ‘M’, ‘H’, ‘Q’, ‘C’, ‘L’, ‘R’, ‘E’, ‘P’, ‘K’

Left Table : B

Right Table : I

B compared with I

B Putted output array earlier

I is remaining from the right and putted output array

Merged list : B I

Left Table : S

Right Table : B I

S compared with B

B Putted output array earlier

S compared with I

I Putted output array earlier

S is remaining from the left and putted output array

Merged list : B I S

Left Table : H

Right Table : Q

H compared with Q

H Putted output array earlier

Q is remaining from the right and putted output array

Merged list : H Q

Left Table : M

Right Table : H Q

M compared with H

H Putted output array earlier

M compared with Q

M Putted output array earlier

Q is remaining from the right and putted output array

Merged list : H M Q

Left Table : B I S

Right Table : H M Q

B compared with H

B Putted output array earlier

I compared with H

H Putted output array earlier

I compared with M

I Putted output array earlier

S compared with M

M Putted output array earlier

S compared with Q

Q Putted output array earlier

S is remaining from the left and putted output array

Merged list : B H I M Q S

Left Table : L

Right Table : R

L compared with R

L Putted output array earlier

R is remaining from the right and putted output array

Merged list : L R

Left Table : C

Right Table : L R

C compared with L

C Putted output array earlier

L is remaining from the right and putted output array

R is remaining from the right and putted output array

Merged list : C L R

Left Table : P

Right Table : K

P compared with K

K Putted output array earlier

P is remaining from the left and putted output array

Merged list : K P

Left Table : E

Right Table : K P

E compared with K

E Putted output array earlier

K is remaining from the right and putted output array

P is remaining from the right and putted output array

Merged list : E K P

Left Table : C L R

Right Table : E K P

C compared with E

C Putted output array earlier

L compared with E

E Putted output array earlier

L compared with K

K Putted output array earlier

L compared with P

L Putted output array earlier

R compared with P

P Putted output array earlier

R is remaining from the left and putted output array

Merged list : C E K L P R

Left Table : B H I M Q S

Right Table : C E K L P R

B compared with C

B Putted output array earlier

H compared with C

C Putted output array earlier

H compared with E

E Putted output array earlier

H compared with K

H Putted output array earlier

I compared with K

I Putted output array earlier

M compared with K

K Putted output array earlier

M compared with L

L Putted output array earlier

M compared with P

M Putted output array earlier

Q compared with P

P Putted output array earlier

Q compared with R

Q Putted output array earlier

S compared with R

R Putted output array earlier

S is remaining from the left and putted output array

Merged list : B C E H I K L M P Q R S

+ Total **comparison** amount = 31

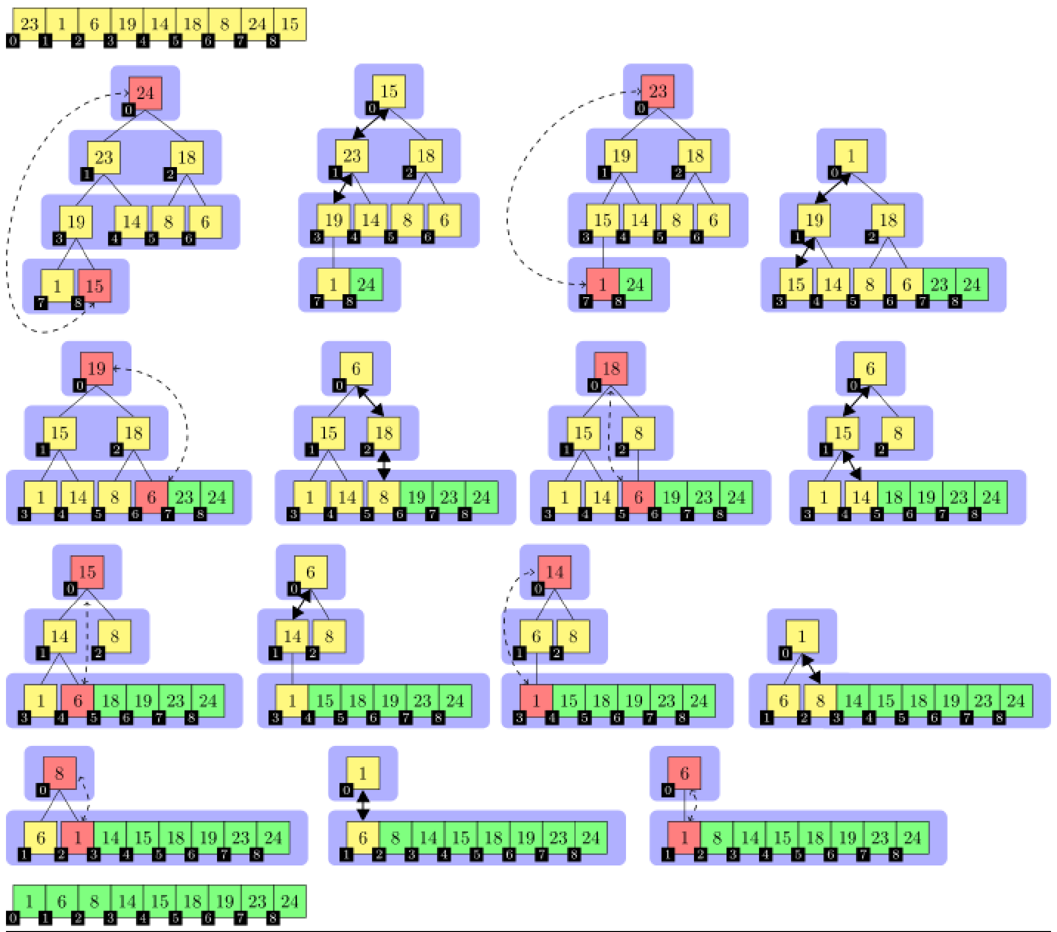
+ Total **displacement** amount = 44

Heap Sort:

***Note :** Heap-sort uses swap operation displacement amount has been admitted as swap operation amount.*

Description :

Heap Sort is a popular and efficient sorting algorithm in computer programming. Learning how to write the heap sort algorithm requires knowledge of two types of data structures - arrays and trees.



Algorithm :

1. Build a max-heap with the original data

2. Now the maximal element is at the root of the tree, take this element at switch it with the last element of the tree.

3. Look at the new tree, that you get by ignoring the last position (this is already sorted). The new tree might not be a max-heap any more, so we transform it into one by sifting the root down.

4. Go to step 2 and repeat until tree is empty.

Given Array = 1 , 2, ,3 ,4, 5, 6, 7, 8, 9, 10

1 compared with 2 and swapped

Current table in built process :

2 1 3 4 5 6 7 8 9 10

2 compared with 3 and swapped

Current table in built process :

3 1 2 4 5 6 7 8 9 10

1 compared with 4 and swapped

Current table in built process :

3 4 2 1 5 6 7 8 9 10

3 compared with 4 and swapped

Current table in built process :

4 3 2 1 5 6 7 8 9 10

3 compared with 5 and swapped

Current table in built process :

4 5 2 1 3 6 7 8 9 10

4 compared with 5 and swapped

Current table in built process :

5 4 2 1 3 6 7 8 9 10

2 compared with 6 and swapped

```
Current table in built process :

5 4 6 1 3 2 7 8 9 10


5 compared with 6 and swapped

Current table in built process :

6 4 5 1 3 2 7 8 9 10


5 compared with 7 and swapped

Current table in built process :

6 4 7 1 3 2 5 8 9 10


6 compared with 7 and swapped

Current table in built process :

7 4 6 1 3 2 5 8 9 10


1 compared with 8 and swapped

Current table in built process :

7 4 6 8 3 2 5 1 9 10


4 compared with 8 and swapped

Current table in built process :

7 8 6 4 3 2 5 1 9 10


7 compared with 8 and swapped

Current table in built process :

8 7 6 4 3 2 5 1 9 10


4 compared with 9 and swapped

Current table in built process :
```

8 7 6 9 3 2 5 1 4 10

7 compared with 9 and swapped

Current table in built process :

8 9 6 7 3 2 5 1 4 10

8 compared with 9 and swapped

Current table in built process :

9 8 6 7 3 2 5 1 4 10

3 compared with 10 and swapped

Current table in built process :

9 8 6 7 10 2 5 1 4 3

8 compared with 10 and swapped

Current table in built process :

9 10 6 7 8 2 5 1 4 3

9 compared with 10 and swapped

Current table in built process :

10 9 6 7 8 2 5 1 4 3

Built table :

10 9 6 7 8 2 5 1 4 3

10 and 3 are swapped

9 compared with 6

3 compared with 9

3 and 9 are swapped

```
Current table in shrink process :
9 3 6 7 8 2 5 1 4 10

7 compared with 8
3 compared with 8
3 and 8 are swapped
Current table in shrink process :
9 8 6 7 3 2 5 1 4 10

9 and 4 are swapped
8 compared with 6
4 compared with 8
4 and 8 are swapped
Current table in shrink process :
8 4 6 7 3 2 5 1 9 10

7 compared with 3
4 compared with 7
4 and 7 are swapped
Current table in shrink process :
8 7 6 4 3 2 5 1 9 10

1 compared with 9
4 compared with 1
8 and 1 are swapped
7 compared with 6
1 compared with 7
1 and 7 are swapped
Current table in shrink process :
```

```
7 1 6 4 3 2 5 8 9 10

4 compared with 3
1 compared with 4
1 and 4 are swapped
Current table in shrink process :
7 4 6 1 3 2 5 8 9 10

7 and 5 are swapped
4 compared with 6
5 compared with 6
5 and 6 are swapped
Current table in shrink process :
6 4 5 1 3 2 7 8 9 10

2 compared with 7
5 compared with 2
6 and 2 are swapped
4 compared with 5
2 compared with 5
2 and 5 are swapped
Current table in shrink process :
5 4 2 1 3 6 7 8 9 10

5 and 3 are swapped
4 compared with 2
3 compared with 4
3 and 4 are swapped
Current table in shrink process :
```

4 3 2 1 5 6 7 8 9 10

1 compared with 5

3 compared with 1

4 and 1 are swapped

3 compared with 2

1 compared with 3

1 and 3 are swapped

Current table in shrink process :

3 1 2 4 5 6 7 8 9 10

3 and 2 are swapped

1 compared with 3

2 compared with 1

2 and 1 are swapped

1 and 1 are swapped

Shrink table :

1 2 3 4 5 6 7 8 9 10

+ Total **comparison** amount = 47

+ Total **displacement** amount = 39 (Swap Amount in heap sort)

Given Array = 10 , 9 , 8 , 7 , 6 , 5 , 4 , 3 , 2 , 1

Built table :

10 9 8 7 6 5 4 3 2 1

10 and 1 are swapped

9 compared with 8

```
1 compared with 9
1 and 9 are swapped
Current table in shrink process :
9 1 8 7 6 5 4 3 2 10

7 compared with 6
1 compared with 7
1 and 7 are swapped
Current table in shrink process :
9 7 8 1 6 5 4 3 2 10

3 compared with 2
1 compared with 3
1 and 3 are swapped
Current table in shrink process :
9 7 8 3 6 5 4 1 2 10

9 and 2 are swapped
7 compared with 8
2 compared with 8
2 and 8 are swapped
Current table in shrink process :
8 7 2 3 6 5 4 1 9 10

5 compared with 4
2 compared with 5
2 and 5 are swapped
Current table in shrink process :
8 7 5 3 6 2 4 1 9 10
```

```
8 and 1 are swapped
7 compared with 5
1 compared with 7
1 and 7 are swapped
Current table in shrink process :
7 1 5 3 6 2 4 8 9 10

3 compared with 6
1 compared with 6
1 and 6 are swapped
Current table in shrink process :
7 6 5 3 1 2 4 8 9 10

7 and 4 are swapped
6 compared with 5
4 compared with 6
4 and 6 are swapped
Current table in shrink process :
6 4 5 3 1 2 7 8 9 10

3 compared with 1
4 compared with 3
6 and 2 are swapped
4 compared with 5
2 compared with 5
2 and 5 are swapped
Current table in shrink process :
5 4 2 3 1 6 7 8 9 10
```

```
5 and 1 are swapped
4 compared with 2
1 compared with 4
1 and 4 are swapped
Current table in shrink process :
4 1 2 3 5 6 7 8 9 10

3 compared with 5
1 compared with 3
1 and 3 are swapped
Current table in shrink process :
4 3 2 1 5 6 7 8 9 10

4 and 1 are swapped
3 compared with 2
1 compared with 3
1 and 3 are swapped
Current table in shrink process :
3 1 2 4 5 6 7 8 9 10

3 and 2 are swapped
1 compared with 3
2 compared with 1
2 and 1 are swapped
1 and 1 are swapped
Shrink table :
1 2 3 4 5 6 7 8 9 10
```

+ Total **comparison** amount = 28

+ Total **displacement** amount = 22

Given Array = 5, 2, 13, 9, 1, 7, 6, 8, 1, 15, 4, 1

5 compared with 13 and swapped

Current table in built process :

13 2 5 9 1 7 6 8 1 15 4 11

2 compared with 9 and swapped

Current table in built process :

13 9 5 2 1 7 6 8 1 15 4 11

5 compared with 7 and swapped

Current table in built process :

13 9 7 2 1 5 6 8 1 15 4 11

2 compared with 8 and swapped

Current table in built process :

13 9 7 8 1 5 6 2 1 15 4 11

1 compared with 15 and swapped

Current table in built process :

13 9 7 8 15 5 6 2 1 1 4 11

9 compared with 15 and swapped

Current table in built process :

13 15 7 8 9 5 6 2 1 1 4 11

```
13 compared with 15 and swapped

Current table in built process :

15 13 7 8 9 5 6 2 1 1 4 11


5 compared with 11 and swapped

Current table in built process :

15 13 7 8 9 11 6 2 1 1 4 5


7 compared with 11 and swapped

Current table in built process :

15 13 11 8 9 7 6 2 1 1 4 5


Built table :

15 13 11 8 9 7 6 2 1 1 4 5


15 and 5 are swapped

13 compared with 11

5 compared with 13

5 and 13 are swapped

Current table in shrink process :

13 5 11 8 9 7 6 2 1 1 4 15


8 compared with 9

5 compared with 9

5 and 9 are swapped

Current table in shrink process :

13 9 11 8 5 7 6 2 1 1 4 15
```

```
1 compared with 4
5 compared with 4
13 and 4 are swapped
9 compared with 11
4 compared with 11
4 and 11 are swapped
Current table in shrink process :
11 9 4 8 5 7 6 2 1 1 13 15

7 compared with 6
4 compared with 7
4 and 7 are swapped
Current table in shrink process :
11 9 7 8 5 4 6 2 1 1 13 15

11 and 1 are swapped
9 compared with 7
1 compared with 9
1 and 9 are swapped
Current table in shrink process :
9 1 7 8 5 4 6 2 1 11 13 15

8 compared with 5
1 compared with 8
1 and 8 are swapped
Current table in shrink process :
9 8 7 1 5 4 6 2 1 11 13 15

2 compared with 1
```

```
1 compared with 2

1 and 2 are swapped

Current table in shrink process :

9 8 7 2 5 4 6 1 1 11 13 15


9 and 1 are swapped

8 compared with 7

1 compared with 8

1 and 8 are swapped

Current table in shrink process :

8 1 7 2 5 4 6 1 9 11 13 15


2 compared with 5

1 compared with 5

1 and 5 are swapped

Current table in shrink process :

8 5 7 2 1 4 6 1 9 11 13 15


8 and 1 are swapped

5 compared with 7

1 compared with 7

1 and 7 are swapped

Current table in shrink process :

7 5 1 2 1 4 6 8 9 11 13 15


4 compared with 6

1 compared with 6

1 and 6 are swapped

Current table in shrink process :
```

```
7 5 6 2 1 4 1 8 9 11 13 15


7 and 1 are swapped

5 compared with 6

1 compared with 6

1 and 6 are swapped

Current table in shrink process :

6 5 1 2 1 4 7 8 9 11 13 15


4 compared with 7

1 compared with 4

1 and 4 are swapped

Current table in shrink process :

6 5 4 2 1 1 7 8 9 11 13 15


6 and 1 are swapped

5 compared with 4

1 compared with 5

1 and 5 are swapped

Current table in shrink process :

5 1 4 2 1 6 7 8 9 11 13 15


2 compared with 1

1 compared with 2

1 and 2 are swapped

Current table in shrink process :

5 2 4 1 1 6 7 8 9 11 13 15


5 and 1 are swapped
```

2 compared with 4

1 compared with 4

1 and 4 are swapped

Current table in shrink process :

4 2 1 1 5 6 7 8 9 11 13 15

4 and 1 are swapped

2 compared with 1

1 compared with 2

1 and 2 are swapped

Current table in shrink process :

2 1 1 4 5 6 7 8 9 11 13 15

2 and 1 are swapped

1 compared with 2

1 compared with 1

1 and 1 are swapped

1 and 1 are swapped

Shrink table :

1 1 2 4 5 6 7 8 9 11 13 15

+ Total **comparison** amount = 47

+ Total **displacement** amount = 38

Given Array = 'S', 'B', 'I', 'M', 'H', 'Q', 'C', 'L', 'R', 'E', 'P', 'K'

B compared with M and swapped

Current table in built process :

S M I B H Q C L R E P K

I compared with Q and swapped

Current table in built process :

S M Q B H I C L R E P K

B compared with L and swapped

Current table in built process :

S M Q L H I C B R E P K

L compared with R and swapped

Current table in built process :

S M Q R H I C B L E P K

M compared with R and swapped

Current table in built process :

S R Q M H I C B L E P K

H compared with P and swapped

Current table in built process :

S R Q M P I C B L E H K

I compared with K and swapped

Current table in built process :

S R Q M P K C B L E H I

Built table :

S R Q M P K C B L E H I

S and I are swapped

R compared with Q

I compared with R

I and R are swapped

Current table in shrink process :

R I Q M P K C B L E H S

M compared with P

I compared with P

I and P are swapped

Current table in shrink process :

R P Q M I K C B L E H S

E compared with H

I compared with H

R and H are swapped

P compared with Q

H compared with Q

H and Q are swapped

Current table in shrink process :

Q P H M I K C B L E R S

K compared with C

H compared with K

H and K are swapped

Current table in shrink process :

Q P K M I H C B L E R S

Q and E are swapped

P compared with K

```
E compared with P

E and P are swapped

Current table in shrink process :

P E K M I H C B L Q R S


M compared with I

E compared with M

E and M are swapped

Current table in shrink process :

P M K E I H C B L Q R S


B compared with L

E compared with L

E and L are swapped

Current table in shrink process :

P M K L I H C B E Q R S


P and E are swapped

M compared with K

E compared with M

E and M are swapped

Current table in shrink process :

M E K L I H C B P Q R S


L compared with I

E compared with L

E and L are swapped

Current table in shrink process :

M L K E I H C B P Q R S
```

B compared with P

E compared with B

M and B are swapped

L compared with K

B compared with L

B and L are swapped

Current table in shrink process :

L B K E I H C M P Q R S

E compared with I

B compared with I

B and I are swapped

Current table in shrink process :

L I K E B H C M P Q R S

L and C are swapped

I compared with K

C compared with K

C and K are swapped

Current table in shrink process :

K I C E B H L M P Q R S

H compared with L

C compared with H

C and H are swapped

Current table in shrink process :

K I H E B C L M P Q R S

```
K and C are swapped

I compared with H

C compared with I

C and I are swapped

Current table in shrink process :

I C H E B K L M P Q R S


E compared with B

C compared with E

C and E are swapped

Current table in shrink process :

I E H C B K L M P Q R S


I and B are swapped

E compared with H

B compared with H

B and H are swapped

Current table in shrink process :

H E B C I K L M P Q R S


H and C are swapped

E compared with B

C compared with E

C and E are swapped

Current table in shrink process :

E C B H I K L M P Q R S


E and B are swapped

C compared with E
```

B compared with C

B and C are swapped

Current table in shrink process :

C B E H I K L M P Q R S

C and B are swapped

B and B are swapped

Shrink table :

B C E H I K L M P Q R S

+ Total **comparison** amount = 47

+ Total **displacement** amount = 37

Quick Sort:

***Note :** Quick uses swap operation displacement amount has been admitted as swap operation amount.*

Description :

QuickSort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot. There are many different versions of quickSort that pick pivot in different ways.

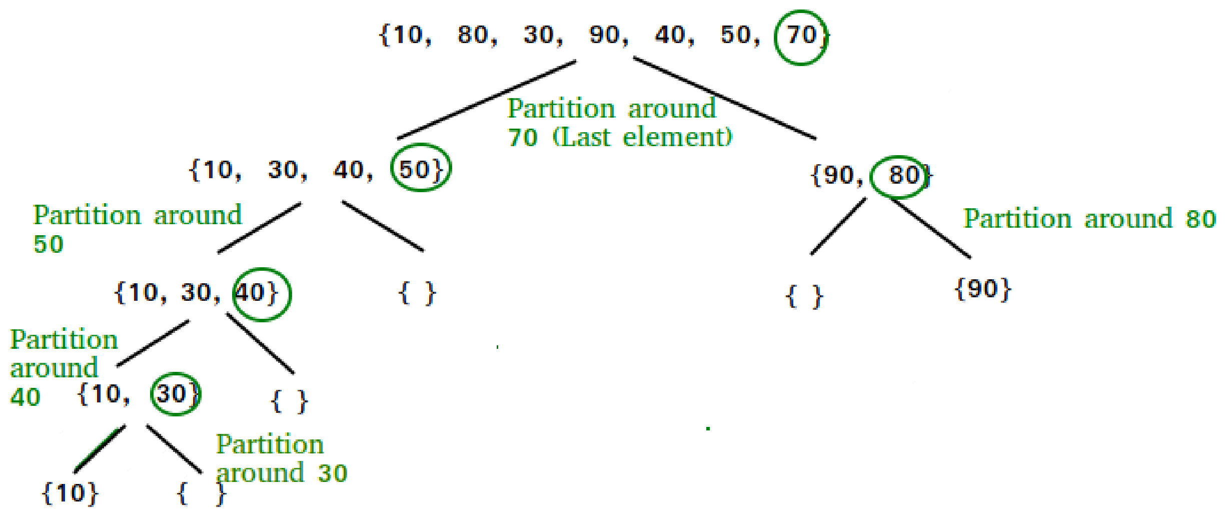
1. Always pick first element as pivot.
2. Always pick last element as pivot (implemented below)
3. Pick a random element as pivot.
4. Pick median as pivot.

The key process in quickSort is partition(). Target of partitions is, given an array and an element x of array as pivot, put x at its correct position in sorted array and put all smaller elements (smaller than x) before x, and put all greater elements (greater than x) after x. All this should be done in linear time.

Pseudo Code for recursive QuickSort function :

```
/* low  --> Starting index,  high  --> Ending index */
quickSort(arr[], low, high)
{
    if (low < high)
    {
        /* pi is partitioning index, arr[pi] is now
           at right place */
        pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1);  // Before pi
        quickSort(arr, pi + 1, high); // After pi
    }
}
```



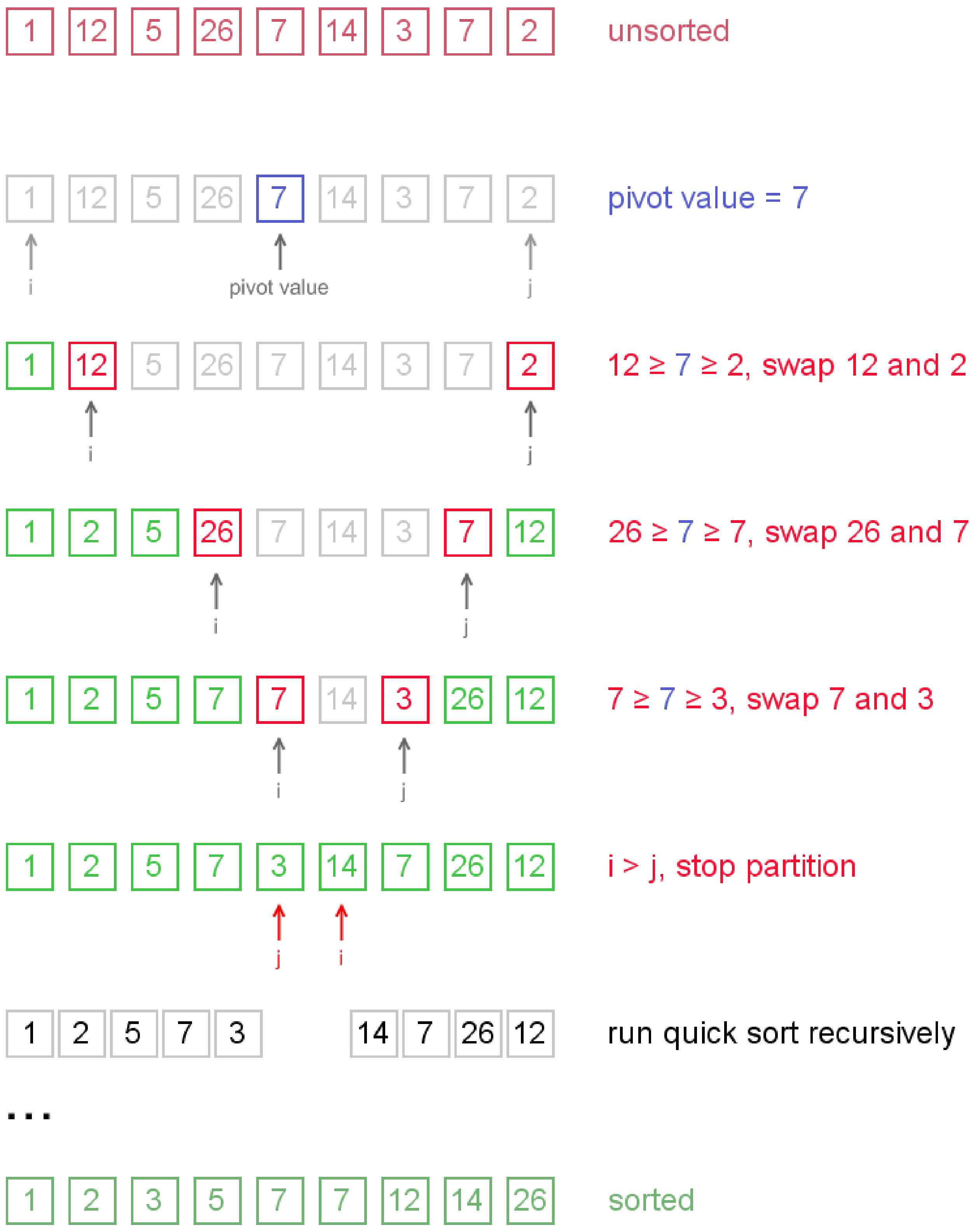
Partition Algorithm

There can be many ways to do partition, following pseudo code adopts the method given in CLRS book. The logic is simple, we start from the leftmost element and keep track of index of smaller (or equal to) elements as i. While traversing, if we find a smaller element, we swap current element with arr[i]. Otherwise we ignore current element.

```
/* This function takes last element as pivot, places
   the pivot element at its correct position in sorted
    array, and places all smaller (smaller than pivot)
   to left of pivot and all greater elements to right
   of pivot */
partition (arr[], low, high)
{
    // pivot (Element to be placed at right position)
    pivot = arr[high];

    i = (low - 1)  // Index of smaller element

    for (j = low; j <= high- 1; j++)
    {
        // If current element is smaller than the pivot
        if (arr[j] < pivot)
        {
            i++;    // increment index of smaller element
            swap arr[i] and arr[j]
        }
    }
    swap arr[i + 1] and arr[high])
    return (i + 1)
}
```


1 12 5 26 7 14 3 7 2
unsorted
1 12 5 26 7 14 3 7 2
pivot value = 7
i
pivot value
j
1 12 5 26 7 14 3 7 2
12 ≥ 7 ≥ 2, swap 12 and 2
i
j
1 2 5 26 7 14 3 7 12
26 ≥ 7 ≥ 7, swap 26 and 7
i
j
1 2 5 7 7 14 3 26 12
7 ≥ 7 ≥ 3, swap 7 and 3
i
j
1 2 5 7 3 14 7 26 12
i > j, stop partition
j
i
1 2 5 7 3 14 7 26 12
run quick sort recursively
. . .
1 2 3 5 7 7 12 14 26
sorted

Given Array = 1 , 2 , 3 , 4 , 5 , 6, 7 , 8 , 9 , 10

Current table before partition :

1 2 3 4 5 6 7 8 9 10

After another sort method :

1 2 3 4 5 6 7 8 9 10

1 and 5 are swapped

Pivot that is chosen : 5

5 which is pivot compared with 5

5 which is pivot compared with 2

5 which is pivot compared with 3

5 which is pivot compared with 4

5 which is pivot compared with 1

5 which is pivot compared with 10

5 which is pivot compared with 9

5 which is pivot compared with 8

5 which is pivot compared with 7

5 which is pivot compared with 6

5 and 1 are swapped

Current table after partition :

1 2 3 4 5 6 7 8 9 10

Sorting of numbers which are smaller than pivot

Current table before partition :

1 2 3 4 5 6 7 8 9 10

```
After another sort method :

1 2 3 4 5 6 7 8 9 10


1 and 2 are swapped


Pivot that is chosen : 2


2 which is pivot  compared with 2

2 which is pivot  compared with 1

2 which is pivot  compared with 4

2 which is pivot  compared with 3



2 and 1 are swapped


Current table after partition :

1 2 3 4 5 6 7 8 9 10

Sorting of numbers which are smaller than pivot


After lower pivot sort

1 2 3 4 5 6 7 8 9 10


Sorting of numbers which are smaller than pivot

Current table before partition :

1 2 3 4 5 6 7 8 9 10

After another sort method :

1 2 3 4 5 6 7 8 9 10


3 and 3 are swapped
```

Pivot that is chosen : 3

3 which is pivot compared with 3

3 which is pivot compared with 4

3 and 3 are swapped

Current table after partition :

1 2 3 4 5 6 7 8 9 10

Sorting of numbers which are smaller than pivot

After lower pivot sort

1 2 3 4 5 6 7 8 9 10

Sorting of numbers which are smaller than pivot

After upper pivot sort

1 2 3 4 5 6 7 8 9 10

After upper pivot sort

1 2 3 4 5 6 7 8 9 10

After lower pivot sort

1 2 3 4 5 6 7 8 9 10

```
Sorting of numbers which are smaller than pivot

Current table before partition :

1 2 3 4 5 6 7 8 9 10

After another sort method :

1 2 3 4 5 6 7 8 9 10


6 and 8 are swapped


Pivot that is chosen : 8


8 which is pivot  compared with 8

8 which is pivot  compared with 7

8 which is pivot  compared with 6

8 which is pivot  compared with 10

8 which is pivot  compared with 9



8 and 6 are swapped


Current table after partition :

1 2 3 4 5 6 7 8 9 10

Sorting of numbers which are smaller than pivot

Current table before partition :

1 2 3 4 5 6 7 8 9 10

After another sort method :

1 2 3 4 5 6 7 8 9 10


6 and 6 are swapped
```

```
Pivot that is chosen : 6


6 which is pivot  compared with 6

6 which is pivot  compared with 7



6 and 6 are swapped


Current table after partition :

1 2 3 4 5 6 7 8 9 10

Sorting of numbers which are smaller than pivot


After lower pivot sort

1 2 3 4 5 6 7 8 9 10


Sorting of numbers which are smaller than pivot


After upper pivot sort

1 2 3 4 5 6 7 8 9 10



After lower pivot sort

1 2 3 4 5 6 7 8 9 10


Sorting of numbers which are smaller than pivot

Current table before partition :

1 2 3 4 5 6 7 8 9 10

After another sort method :

1 2 3 4 5 6 7 8 9 10
```

9 and 9 are swapped

Pivot that is chosen : 9

9 which is pivot compared with 9

9 which is pivot compared with 10

9 and 9 are swapped

Current table after partition :

1 2 3 4 5 6 7 8 9 10

Sorting of numbers which are smaller than pivot

After lower pivot sort

1 2 3 4 5 6 7 8 9 10

Sorting of numbers which are smaller than pivot

After upper pivot sort

1 2 3 4 5 6 7 8 9 10

+ Total **comparison** amount   = 43

+ Total **displacement** amount   = 12

Given Array = 10 , 9 , 8 , 7 , 6 , 5 , 4 , 3 , 2 , 1

Current table before partition :

```
10 9 8 7 6 5 4 3 2 1

6 and 10 are swapped

1 and 10 are swapped

1 and 6 are swapped

After another sort method :

1 9 8 7 6 5 4 3 2 10


1 and 6 are swapped


Pivot that is chosen : 6


6 which is pivot  compared with 6

6 which is pivot  compared with 10

9 and 2 are swapped


6 which is pivot  compared with 2

6 which is pivot  compared with 9

8 and 3 are swapped


6 which is pivot  compared with 3

6 which is pivot  compared with 8

7 and 4 are swapped


6 which is pivot  compared with 4

6 which is pivot  compared with 1

6 which is pivot  compared with 5

6 which is pivot  compared with 7
```

```
6 and 5 are swapped


Current table after partition :

5 2 3 4 1 6 7 8 9 10

Sorting of numbers which are smaller than pivot

Current table before partition :

5 2 3 4 1 6 7 8 9 10

3 and 5 are swapped

1 and 5 are swapped

1 and 3 are swapped


After another sort method :

1 2 3 4 5 6 7 8 9 10


1 and 3 are swapped


Pivot that is chosen : 3


3 which is pivot  compared with 3

3 which is pivot  compared with 2

3 which is pivot  compared with 1

3 which is pivot  compared with 5

3 which is pivot  compared with 4

3 and 1 are swapped


Current table after partition :

1 2 3 4 5 6 7 8 9 10

Sorting of numbers which are smaller than pivot

Current table before partition :
```

```
1 2 3 4 5 6 7 8 9 10

After another sort method :

1 2 3 4 5 6 7 8 9 10


1 and 1 are swapped


Pivot that is chosen : 1


1 which is pivot  compared with 1

1 which is pivot  compared with 2



1 and 1 are swapped


Current table after partition :

1 2 3 4 5 6 7 8 9 10

Sorting of numbers which are smaller than pivot


After lower pivot sort

1 2 3 4 5 6 7 8 9 10


Sorting of numbers which are smaller than pivot



After upper pivot sort

1 2 3 4 5 6 7 8 9 10


After lower pivot sort

1 2 3 4 5 6 7 8 9 10
```

```
Sorting of numbers which are smaller than pivot

Current table before partition :

1 2 3 4 5 6 7 8 9 10

After another sort method :

1 2 3 4 5 6 7 8 9 10


4 and 4 are swapped

Pivot that is chosen : 4


4 which is pivot  compared with 4

4 which is pivot  compared with 5


4 and 4 are swapped


Current table after partition :

1 2 3 4 5 6 7 8 9 10

Sorting of numbers which are smaller than pivot


After lower pivot sort

1 2 3 4 5 6 7 8 9 10




Sorting of numbers which are smaller than pivot


After upper pivot sort

1 2 3 4 5 6 7 8 9 10
```

```
After upper pivot sort

1 2 3 4 5 6 7 8 9 10


After lower pivot sort

1 2 3 4 5 6 7 8 9 10


Sorting of numbers which are smaller than pivot


Current table before partition :

1 2 3 4 5 6 7 8 9 10

After another sort method :

1 2 3 4 5 6 7 8 9 10


7 and 8 are swapped


Pivot that is chosen : 8

8 which is pivot  compared with 8

8 which is pivot  compared with 7

8 which is pivot  compared with 10

8 which is pivot  compared with 9


8 and 7 are swapped

Current table after partition :

1 2 3 4 5 6 7 8 9 10

Sorting of numbers which are smaller than pivot

After lower pivot sort

1 2 3 4 5 6 7 8 9 10


Sorting of numbers which are smaller than pivot
```

Current table before partition :

1 2 3 4 5 6 7 8 9 10

After another sort method :

1 2 3 4 5 6 7 8 9 10

9 and 9 are swapped

Pivot that is chosen : 9

9 which is pivot compared with 9

9 which is pivot compared with 10

9 and 9 are swapped

Current table after partition :

1 2 3 4 5 6 7 8 9 10

Sorting of numbers which are smaller than pivot

After lower pivot sort

1 2 3 4 5 6 7 8 9 10

Sorting of numbers which are smaller than pivot

After upper pivot sort

1 2 3 4 5 6 7 8 9 10

+ Total **comparison** amount = 43

+ Total **displacement** amount = 21

**Given Array = ** 5, 2, 13, 9, 1, 7, 6, 8, 1, 15, 4, 11

Current table before partition :

5 2 13 9 1 7 6 8 1 15 4 11

After another sort method :

5 2 13 9 1 7 6 8 1 15 4 11

5 and 7 are swapped

Pivot that is chosen : 7

7 which is pivot compared with 7

7 which is pivot compared with 2

7 which is pivot compared with 11

13 and 4 are swapped

7 which is pivot compared with 4

7 which is pivot compared with 13

7 which is pivot compared with 15

9 and 1 are swapped

7 which is pivot compared with 1

7 which is pivot compared with 1

7 which is pivot compared with 5

7 which is pivot compared with 6

7 which is pivot compared with 9

7 which is pivot compared with 8

```
7 and 6 are swapped


Current table after partition :

6 2 4 1 1 5 7 8 9 15 13 11

Sorting of numbers which are smaller than pivot

Current table before partition :

6 2 4 1 1 5 7 8 9 15 13 11

4 and 6 are swapped

5 and 6 are swapped

After another sort method :

4 2 5 1 1 6 7 8 9 15 13 11


4 and 5 are swapped


Pivot that is chosen : 5


5 which is pivot  compared with 5

5 which is pivot  compared with 2

5 which is pivot  compared with 4

5 which is pivot  compared with 1

5 which is pivot  compared with 1

5 which is pivot  compared with 6



5 and 1 are swapped


Current table after partition :

1 2 4 1 5 6 7 8 9 15 13 11
```

```
Sorting of numbers which are smaller than pivot

Current table before partition :

1 2 4 1 5 6 7 8 9 15 13 11

1 and 2 are swapped

After another sort method :

1 1 4 2 5 6 7 8 9 15 13 11


1 and 1 are swapped


Pivot that is chosen : 1


1 which is pivot  compared with 1

1 which is pivot  compared with 1

1 which is pivot  compared with 2

1 which is pivot  compared with 4



1 and 1 are swapped


Current table after partition :

1 1 4 2 5 6 7 8 9 15 13 11

Sorting of numbers which are smaller than pivot


After lower pivot sort

1 1 4 2 5 6 7 8 9 15 13 11


Sorting of numbers which are smaller than pivot

Current table before partition :

1 1 4 2 5 6 7 8 9 15 13 11
```

```
2 and 4 are swapped
After another sort method :
1 1 2 4 5 6 7 8 9 15 13 11

2 and 2 are swapped

Pivot that is chosen : 2

2 which is pivot  compared with 2
2 which is pivot  compared with 4


2 and 2 are swapped

Current table after partition :
1 1 2 4 5 6 7 8 9 15 13 11
Sorting of numbers which are smaller than pivot

After lower pivot sort
1 1 2 4 5 6 7 8 9 15 13 11

Sorting of numbers which are smaller than pivot

After upper pivot sort
1 1 2 4 5 6 7 8 9 15 13 11


After upper pivot sort
1 1 2 4 5 6 7 8 9 15 13 11
```

```
After lower pivot sort

1 1 2 4 5 6 7 8 9 15 13 11


Sorting of numbers which are smaller than pivot


After upper pivot sort

1 1 2 4 5 6 7 8 9 15 13 11



After lower pivot sort

1 1 2 4 5 6 7 8 9 15 13 11


Sorting of numbers which are smaller than pivot

Current table before partition :

1 1 2 4 5 6 7 8 9 15 13 11

11 and 15 are swapped

After another sort method :

1 1 2 4 5 6 7 8 9 11 13 15


8 and 11 are swapped


Pivot that is chosen : 11


11 which is pivot  compared with 11

11 which is pivot  compared with 9

11 which is pivot  compared with 8

11 which is pivot  compared with 15
```

```
11 which is pivot  compared with 13



11 and 8 are swapped


Current table after partition :

1 1 2 4 5 6 7 8 9 11 13 15

Sorting of numbers which are smaller than pivot

Current table before partition :

1 1 2 4 5 6 7 8 9 11 13 15

After another sort method :

1 1 2 4 5 6 7 8 9 11 13 15


8 and 8 are swapped


Pivot that is chosen : 8


8 which is pivot  compared with 8

8 which is pivot  compared with 9



8 and 8 are swapped


Current table after partition :

1 1 2 4 5 6 7 8 9 11 13 15

Sorting of numbers which are smaller than pivot


After lower pivot sort

1 1 2 4 5 6 7 8 9 11 13 15
```

```
Sorting of numbers which are smaller than pivot


After upper pivot sort

1 1 2 4 5 6 7 8 9 11 13 15



After lower pivot sort

1 1 2 4 5 6 7 8 9 11 13 15


Sorting of numbers which are smaller than pivot

Current table before partition :

1 1 2 4 5 6 7 8 9 11 13 15

After another sort method :

1 1 2 4 5 6 7 8 9 11 13 15


13 and 13 are swapped


Pivot that is chosen : 13


13 which is pivot  compared with 13

13 which is pivot  compared with 15



13 and 13 are swapped


Current table after partition :

1 1 2 4 5 6 7 8 9 11 13 15

Sorting of numbers which are smaller than pivot
```

After lower pivot sort

1 1 2 4 5 6 7 8 9 11 13 15

Sorting of numbers which are smaller than pivot

After upper pivot sort

1 1 2 4 5 6 7 8 9 11 13 15

+ Total **comparison** amount = 54

+ Total **displacement** amount = 21

Given Array = ‘S’, ‘B’, ‘I’, ‘M’, ‘H’, ‘Q’, ‘C’, ‘L’, ‘R’, ‘E’, ‘P’, ‘K’

Current table before partition :

S B I M H Q C L R E P K

Q and S are swapped

K and S are swapped

K and Q are swapped

After another sort method :

K B I M H Q C L R E P S

K and Q are swapped

Pivot that is chosen : Q

Q which is pivot compared with Q

Q which is pivot compared with B

Q which is pivot compared with I

Q which is pivot compared with M

Q which is pivot compared with H

Q which is pivot compared with K

Q which is pivot compared with C

Q which is pivot compared with L

Q which is pivot compared with S

R and P are swapped

Q which is pivot compared with P

Q which is pivot compared with E

Q which is pivot compared with R

Q and E are swapped

Current table after partition :

E B I M H K C L P Q R S

Sorting of numbers which are smaller than pivot

Current table before partition :

E B I M H K C L P Q R S

After another sort method :

E B I M H K C L P Q R S

E and H are swapped

Pivot that is chosen : H

H which is pivot compared with H

H which is pivot compared with B

H which is pivot compared with P

H which is pivot compared with L

I and C are swapped

H which is pivot compared with C

H which is pivot compared with I

H which is pivot compared with K

M and E are swapped

H which is pivot compared with E

H which is pivot compared with M

H and E are swapped

Current table after partition :

E B C H M K I L P Q R S

Sorting of numbers which are smaller than pivot

Current table before partition :

E B C H M K I L P Q R S

B and E are swapped

C and E are swapped

After another sort method :

B C E H M K I L P Q R S

B and C are swapped

Pivot that is chosen : C

```
C which is pivot  compared with C

C which is pivot  compared with B

C which is pivot  compared with E



C and B are swapped


Current table after partition :

B C E H M K I L P Q R S

Sorting of numbers which are smaller than pivot


After lower pivot sort

B C E H M K I L P Q R S


Sorting of numbers which are smaller than pivot


After upper pivot sort

B C E H M K I L P Q R S



After lower pivot sort

B C E H M K I L P Q R S


Sorting of numbers which are smaller than pivot

Current table before partition :

B C E H M K I L P Q R S

I and M are swapped

After another sort method :
```

B C E H I K M L P Q R S

I and M are swapped

Pivot that is chosen : M

M which is pivot compared with M

M which is pivot compared with K

M which is pivot compared with I

M which is pivot compared with L

M which is pivot compared with P

M and L are swapped

Current table after partition :

B C E H L K I M P Q R S

Sorting of numbers which are smaller than pivot

Current table before partition :

B C E H L K I M P Q R S

K and L are swapped

I and L are swapped

I and K are swapped

After another sort method :

B C E H I K L M P Q R S

I and K are swapped

Pivot that is chosen : K

K which is pivot compared with K

K which is pivot compared with I

K which is pivot compared with L

K and I are swapped

Current table after partition :

B C E H I K L M P Q R S

Sorting of numbers which are smaller than pivot

After lower pivot sort

B C E H I K L M P Q R S

Sorting of numbers which are smaller than pivot

After upper pivot sort

B C E H I K L M P Q R S

After lower pivot sort

B C E H I K L M P Q R S

Sorting of numbers which are smaller than pivot

After upper pivot sort

B C E H I K L M P Q R S

After upper pivot sort

B C E H I K L M P Q R S

After lower pivot sort

B C E H I K L M P Q R S

Sorting of numbers which are smaller than pivot

Current table before partition :

B C E H I K L M P Q R S

After another sort method :

B C E H I K L M P Q R S

R and R are swapped

Pivot that is chosen : R

R which is pivot compared with R

R which is pivot compared with S

R and R are swapped

Current table after partition :

B C E H I K L M P Q R S

Sorting of numbers which are smaller than pivot

After lower pivot sort

B C E H I K L M P Q R S

Sorting of numbers which are smaller than pivot

After upper pivot sort

B C E H I K L M P Q R S

+ Total **comparison** amount = 52

+ Total **displacement** amount = 24