Shell sort

Performing shell sort on a sorted array

Shell sort being executed for array: 1 2 3 4 5 6 7 8 9 10

Current 'gap' is: 5

Since 6 at position 5 is bigger or equal than 1 at position 0 we will not perform the swapping

Since 7 at position 6 is bigger or equal than 2 at position 1 we will not perform the swapping

Since 8 at position 7 is bigger or equal than 3 at position 2 we will not perform the swapping

Since 9 at position 8 is bigger or equal than 4 at position 3 we will not perform the swapping

Since 10 at position 9 is bigger or equal than 5 at position 4 we will not perform the swapping

Current 'gap' is: 2

Since 3 at position 2 is bigger or equal than 1 at position 0 we will not perform the swapping

Since 4 at position 3 is bigger or equal than 2 at position 1 we will not perform the swapping

Since 5 at position 4 is bigger or equal than 3 at position 2 we will not perform the swapping

Since 6 at position 5 is bigger or equal than 4 at position 3 we will not perform the swapping

Since 7 at position 6 is bigger or equal than 5 at position 4 we will not perform the swapping

Since 8 at position 7 is bigger or equal than 6 at position 5 we will not perform the swapping

Since 9 at position 8 is bigger or equal than 7 at position 6 we will not perform the swapping

Since 10 at position 9 is bigger or equal than 8 at position 7 we will not perform the swapping

Current 'gap' is: 1

Since 2 at position 1 is bigger or equal than 1 at position 0 we will not perform the swapping

Since 3 at position 2 is bigger or equal than 2 at position 1 we will not perform the swapping

Since 4 at position 3 is bigger or equal than 3 at position 2 we will not perform the swapping

Since 5 at position 4 is bigger or equal than 4 at position 3 we will not perform the swapping

Since 6 at position 5 is bigger or equal than 5 at position 4 we will not perform the swapping

Since 7 at position 6 is bigger or equal than 6 at position 5 we will not perform the swapping

Since 8 at position 7 is bigger or equal than 7 at position 6 we will not perform the swapping

Since 9 at position 8 is bigger or equal than 8 at position 7 we will not perform the swapping

Since 10 at position 9 is bigger or equal than 9 at position 8 we will not perform the swapping

Gap will be reduced again and will be less than 1 so the sorting will be finished

In the end total number of comparisons was: 22

Total number of displacements was: 0

Final array is:

1 2 3 4 5 6 7 8 9 10

Performing shell sort on a reversely sorted array

Shell sort being executed for array: 10 9 8 7 6 5 4 3 2 1

Current 'gap' is: 5

Since 5 at position 5 is smaller than 10 at position 0 we will swap them

Our array is now

5 9 8 7 6 10 4 3 2 1

Total number of displacements: 1

Total number of comparisons: 1

Since 4 at position 6 is smaller than 9 at position 1 we will swap them

Our array is now

5 4 8 7 6 10 9 3 2 1

Total number of displacements: 2

Total number of comparisons: 2

Since 3 at position 7 is smaller than 8 at position 2 we will swap them

Our array is now

5 4 3 7 6 10 9 8 2 1

Total number of displacements: 3

Total number of comparisons: 3

Since 2 at position 8 is smaller than 7 at position 3 we will swap them

Our array is now

5 4 3 2 6 10 9 8 7 1

Total number of displacements: 4

Total number of comparisons: 4

Since 1 at position 9 is smaller than 6 at position 4 we will swap them

Our array is now

5 4 3 2 1 10 9 8 7 6

Total number of displacements: 5

Total number of comparisons: 5

Current 'gap' is: 2

Since 3 at position 2 is smaller than 5 at position 0 we will swap them

Our array is now

3 4 5 2 1 10 9 8 7 6

Total number of displacements: 6

Total number of comparisons: 6

Since 2 at position 3 is smaller than 4 at position 1 we will swap them

Our array is now

3 2 5 4 1 10 9 8 7 6

Total number of displacements: 7

Total number of comparisons: 7

Since 1 at position 4 is smaller than 5 at position 2 we will swap them

Our array is now

3 2 1 4 5 10 9 8 7 6

Total number of displacements: 8

Total number of comparisons: 8

Since 1 at position 2 is smaller than 3 at position 0 we will swap them

Our array is now

1 2 3 4 5 10 9 8 7 6

Total number of displacements: 9

Total number of comparisons: 9

Since 10 at position 5 is bigger or equal than 4 at position 3 we will not perform the swapping

Since 9 at position 6 is bigger or equal than 5 at position 4 we will not perform the swapping

Since 8 at position 7 is smaller than 10 at position 5 we will swap them

Our array is now

1 2 3 4 5 8 9 10 7 6

Total number of displacements: 10

Total number of comparisons: 12

Since 8 at position 5 is bigger or equal than 4 at position 3 we will not perform the swapping

Since 7 at position 8 is smaller than 9 at position 6 we will swap them

Our array is now

1 2 3 4 5 8 7 10 9 6

Total number of displacements: 11

Total number of comparisons: 14

Since 7 at position 6 is bigger or equal than 5 at position 4 we will not perform the swapping

Since 6 at position 9 is smaller than 10 at position 7 we will swap them

Our array is now

1 2 3 4 5 8 7 6 9 10

Total number of displacements: 12

Total number of comparisons: 16

Since 6 at position 7 is smaller than 8 at position 5 we will swap them

Our array is now

1 2 3 4 5 6 7 8 9 10

Total number of displacements: 13

Total number of comparisons: 17

Since 6 at position 5 is bigger or equal than 4 at position 3 we will not perform the swapping

Current 'gap' is: 1

Since 2 at position 1 is bigger or equal than 1 at position 0 we will not perform the swapping

Since 3 at position 2 is bigger or equal than 2 at position 1 we will not perform the swapping

Since 4 at position 3 is bigger or equal than 3 at position 2 we will not perform the swapping

Since 5 at position 4 is bigger or equal than 4 at position 3 we will not perform the swapping

Since 6 at position 5 is bigger or equal than 5 at position 4 we will not perform the swapping

Since 7 at position 6 is bigger or equal than 6 at position 5 we will not perform the swapping

Since 8 at position 7 is bigger or equal than 7 at position 6 we will not perform the swapping

Since 9 at position 8 is bigger or equal than 8 at position 7 we will not perform the swapping

Since 10 at position 9 is bigger or equal than 9 at position 8 we will not perform the swapping

Gap will be reduced again and will be less than 1 so the sorting will be finished

In the end total number of comparisons was: 27

Total number of displacements was: 13

Final array is:

1 2 3 4 5 6 7 8 9 10

Performing shell sort on a given array {5, 2, 13, 9, 1, 7, 6, 8, 1, 15, 4, 11}

Shell sort being executed for array: 5 2 13 9 1 7 6 8 1 15 4 11

Current 'gap' is: 6

Since 6 at position 6 is bigger or equal than 5 at position 0 we will not perform the swapping

Since 8 at position 7 is bigger or equal than 2 at position 1 we will not perform the swapping

Since 1 at position 8 is smaller than 13 at position 2 we will swap them

Our array is now

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

Total number of displacements: 1

Total number of comparisons: 3

Since 15 at position 9 is bigger or equal than 9 at position 3 we will not perform the swapping

Since 4 at position 10 is bigger or equal than 1 at position 4 we will not perform the swapping

Since 11 at position 11 is bigger or equal than 7 at position 5 we will not perform the swapping

Current 'gap' is: 3

Since 9 at position 3 is bigger or equal than 5 at position 0 we will not perform the swapping

Since 1 at position 4 is smaller than 2 at position 1 we will swap them

Our array is now

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

Total number of displacements: 2

Total number of comparisons: 8

Since 7 at position 5 is bigger or equal than 1 at position 2 we will not perform the swapping

Since 6 at position 6 is smaller than 9 at position 3 we will swap them

Our array is now

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

Total number of displacements: 3

Total number of comparisons: 10

Since 6 at position 3 is bigger or equal than 5 at position 0 we will not perform the swapping

Since 8 at position 7 is bigger or equal than 2 at position 4 we will not perform the swapping

Since 13 at position 8 is bigger or equal than 7 at position 5 we will not perform the swapping

Since 15 at position 9 is bigger or equal than 9 at position 6 we will not perform the swapping

Since 4 at position 10 is smaller than 8 at position 7 we will swap them

Our array is now

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

Total number of displacements: 4

Total number of comparisons: 15

Since 4 at position 7 is bigger or equal than 2 at position 4 we will not perform the swapping

Since 11 at position 11 is smaller than 13 at position 8 we will swap them

Our array is now

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

Total number of displacements: 5

Total number of comparisons: 17

Since 11 at position 8 is bigger or equal than 7 at position 5 we will not perform the swapping

Current 'gap' is: 1

Since 1 at position 1 is smaller than 5 at position 0 we will swap them

Our array is now

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

Total number of displacements: 6

Total number of comparisons: 19

Since 1 at position 2 is smaller than 5 at position 1 we will swap them

Our array is now

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

Total number of displacements: 7

Total number of comparisons: 20

Since 1 at position 1 is bigger or equal than 1 at position 0 we will not perform the swapping

Since 6 at position 3 is bigger or equal than 5 at position 2 we will not perform the swapping

Since 2 at position 4 is smaller than 6 at position 3 we will swap them

Our array is now

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

Total number of displacements: 8

Total number of comparisons: 23

Since 2 at position 3 is smaller than 5 at position 2 we will swap them

Our array is now

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

Total number of displacements: 9

Total number of comparisons: 24

Since 2 at position 2 is bigger or equal than 1 at position 1 we will not perform the swapping

Since 7 at position 5 is bigger or equal than 6 at position 4 we will not perform the swapping

Since 9 at position 6 is bigger or equal than 7 at position 5 we will not perform the swapping

Since 4 at position 7 is smaller than 9 at position 6 we will swap them

Our array is now

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

Total number of displacements: 10

Total number of comparisons: 28

Since 4 at position 6 is smaller than 7 at position 5 we will swap them

Our array is now

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

Total number of displacements: 11

Total number of comparisons: 29

Since 4 at position 5 is smaller than 6 at position 4 we will swap them

Our array is now

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

Total number of displacements: 12

Total number of comparisons: 30

Since 4 at position 4 is smaller than 5 at position 3 we will swap them

Our array is now

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

Total number of displacements: 13

Total number of comparisons: 31

Since 4 at position 3 is bigger or equal than 2 at position 2 we will not perform the swapping

Since 11 at position 8 is bigger or equal than 9 at position 7 we will not perform the swapping

Since 15 at position 9 is bigger or equal than 11 at position 8 we will not perform the swapping

Since 8 at position 10 is smaller than 15 at position 9 we will swap them

Our array is now

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

Total number of displacements: 14

Total number of comparisons: 35

Since 8 at position 9 is smaller than 11 at position 8 we will swap them

Our array is now

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

Total number of displacements: 15

Total number of comparisons: 36

Since 8 at position 8 is smaller than 9 at position 7 we will swap them

Our array is now

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

Total number of displacements: 16

Total number of comparisons: 37

Since 8 at position 7 is bigger or equal than 7 at position 6 we will not perform the swapping

Since 13 at position 11 is smaller than 15 at position 10 we will swap them

Our array is now

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

Total number of displacements: 17

Total number of comparisons: 39

Since 13 at position 10 is bigger or equal than 11 at position 9 we will not perform the swapping

Gap will be reduced again and will be less than 1 so the sorting will be finished

In the end total number of comparisons was: 40

Total number of displacements was: 17

Final array is:

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

Performing shell sort on a given array of characters {'S', 'B', 'I', 'M', 'H', 'Q', 'C', 'L', 'R', 'E', 'P', 'K'}

Shell sort being executed for array: S B I M H Q C L R E P K

Current 'gap' is: 6

Since C at position 6 is smaller than S at position 0 we will swap them

Our array is now

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

Total number of displacements: 1

Total number of comparisons: 1

Since L at position 7 is bigger or equal than B at position 1 we will not perform the swapping

Since R at position 8 is bigger or equal than I at position 2 we will not perform the swapping

Since E at position 9 is smaller than M at position 3 we will swap them

Our array is now

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

Total number of displacements: 2

Total number of comparisons: 4

Since P at position 10 is bigger or equal than H at position 4 we will not perform the swapping

Since K at position 11 is smaller than Q at position 5 we will swap them

Our array is now

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

Total number of displacements: 3

Total number of comparisons: 6

Current 'gap' is: 3

Since E at position 3 is bigger or equal than C at position 0 we will not perform the swapping

Since H at position 4 is bigger or equal than B at position 1 we will not perform the swapping

Since K at position 5 is bigger or equal than I at position 2 we will not perform the swapping

Since S at position 6 is bigger or equal than E at position 3 we will not perform the swapping

Since L at position 7 is bigger or equal than H at position 4 we will not perform the swapping

Since R at position 8 is bigger or equal than K at position 5 we will not perform the swapping

Since M at position 9 is smaller than S at position 6 we will swap them

Our array is now

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

Total number of displacements: 4

Total number of comparisons: 13

Since M at position 6 is bigger or equal than E at position 3 we will not perform the swapping

Since P at position 10 is bigger or equal than L at position 7 we will not perform the swapping

Since Q at position 11 is smaller than R at position 8 we will swap them

Our array is now

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

Total number of displacements: 5

Total number of comparisons: 16

Since Q at position 8 is bigger or equal than K at position 5 we will not perform the swapping

Current 'gap' is: 1

Since B at position 1 is smaller than C at position 0 we will swap them

Our array is now

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

Total number of displacements: 6

Total number of comparisons: 18

Since I at position 2 is bigger or equal than C at position 1 we will not perform the swapping

Since E at position 3 is smaller than I at position 2 we will swap them

Our array is now

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

Total number of displacements: 7

Total number of comparisons: 20

Since E at position 2 is bigger or equal than C at position 1 we will not perform the swapping

Since H at position 4 is smaller than I at position 3 we will swap them

Our array is now

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

Total number of displacements: 8

Total number of comparisons: 22

Since H at position 3 is bigger or equal than E at position 2 we will not perform the swapping

Since K at position 5 is bigger or equal than I at position 4 we will not perform the swapping

Since M at position 6 is bigger or equal than K at position 5 we will not perform the swapping

Since L at position 7 is smaller than M at position 6 we will swap them

Our array is now

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

Total number of displacements: 9

Total number of comparisons: 26

Since L at position 6 is bigger or equal than K at position 5 we will not perform the swapping

Since Q at position 8 is bigger or equal than M at position 7 we will not perform the swapping

Since S at position 9 is bigger or equal than Q at position 8 we will not perform the swapping

Since P at position 10 is smaller than S at position 9 we will swap them

Our array is now

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

Total number of displacements: 10

Total number of comparisons: 30

Since P at position 9 is smaller than Q at position 8 we will swap them

Our array is now

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

Total number of displacements: 11

Total number of comparisons: 31

Since P at position 8 is bigger or equal than M at position 7 we will not perform the swapping

Since R at position 11 is smaller than S at position 10 we will swap them

Our array is now

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

Total number of displacements: 12

Total number of comparisons: 33

Since R at position 10 is bigger or equal than Q at position 9 we will not perform the swapping

Gap will be reduced again and will be less than 1 so the sorting will be finished

In the end total number of comparisons was: 34

Total number of displacements was: 12

Final array is:

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

Merge sort

Merge sort

NOTE: Since merge sort is not in place algorithm i won't talk about the number of 'displacements' . Also I just performed merge sort and didn't talk about the number of displacements for each merge of two arrays(separate linear funciton that merges two sorted arrays in one sorted array) instead I just calculated the number of comparisons during those merges and gave a final answer. If I did also write for merge it would just be way to long. Also I was executing it the same way the program would (recursively).

Merge sorting {1,2,3,4,5,6,7,8,9,10}

Our current array that will be separated in left and right array is: 1 2 3 4 5 6 7 8 9 10

Left array: 1 2 3 4 5

Right array: 6 7 8 9 10

Our current array that will be separated in left and right array is: 1 2 3 4 5

Left array: 1 2

Right array: 3 4 5

Our current array that will be separated in left and right array is: 1 2

Left array: 1

Right array: 2

Left array (merged): 1

Right array (merged): 2

After this we will merge and the merged array will be: 1 2

Our current array that will be separated in left and right array is: 3 4 5

Left array: 3

Right array: 4 5

Our current array that will be separated in left and right array is: 4 5

Left array: 4

Right array: 5

Left array (merged): 4

Right array (merged): 5

After this we will merge and the merged array will be: 4 5

Left array (merged): 3

Right array (merged): 4 5

After this we will merge and the merged array will be: 3 4 5

Left array (merged): 1 2

Right array (merged): 3 4 5

After this we will merge and the merged array will be: 1 2 3 4 5

Our current array that will be separated in left and right array is: 6 7 8 9 10

Left array: 6 7

Right array: 8 9 10

Our current array that will be separated in left and right array is: 6 7

Left array: 6

Right array: 7

Left array (merged): 6

Right array (merged): 7

After this we will merge and the merged array will be: 6 7

Our current array that will be separated in left and right array is: 8 9 10

Left array: 8

Right array: 9 10

Our current array that will be separated in left and right array is: 9 10

Left array: 9

Right array: 10

Left array (merged): 9

Right array (merged): 10

After this we will merge and the merged array will be: 9 10

Left array (merged): 8

Right array (merged): 9 10

After this we will merge and the merged array will be: 8 9 10

Left array (merged): 6 7

Right array (merged): 8 9 10

After this we will merge and the merged array will be: 6 7 8 9 10

Left array (merged): 1 2 3 4 5

Right array (merged): 6 7 8 9 10

After this we will merge and the merged array will be: 1 2 3 4 5 6 7 8 9 10

Numer of comparisons is: 15

So in the end after sorting array is:

1 2 3 4 5 6 7 8 9 10

Merge sorting {10,9,8,7,6,5,4,3,2,1}

Our current array that will be separated in left and right array is: 10 9 8 7 6 5 4 3 2 1

Left array: 10 9 8 7 6

Right array: 5 4 3 2 1

Our current array that will be separated in left and right array is: 10 9 8 7 6

Left array: 10 9

Right array: 8 7 6

Our current array that will be separated in left and right array is: 10 9

Left array: 10

Right array: 9

Left array (merged): 10

Right array (merged): 9

After this we will merge and the merged array will be: 9 10

Our current array that will be separated in left and right array is: 8 7 6

Left array: 8

Right array: 7 6

Our current array that will be separated in left and right array is: 7 6

Left array: 7

Right array: 6

Left array (merged): 7

Right array (merged): 6

After this we will merge and the merged array will be: 6 7

Left array (merged): 8

Right array (merged): 6 7

After this we will merge and the merged array will be: 6 7 8

Left array (merged): 9 10

Right array (merged): 6 7 8

After this we will merge and the merged array will be: 6 7 8 9 10

Our current array that will be separated in left and right array is: 5 4 3 2 1

Left array: 5 4

Right array: 3 2 1

Our current array that will be separated in left and right array is: 5 4

Left array: 5

Right array: 4

Left array (merged): 5

Right array (merged): 4

After this we will merge and the merged array will be: 4 5

Our current array that will be separated in left and right array is: 3 2 1

Left array: 3

Right array: 2 1

Our current array that will be separated in left and right array is: 2 1

Left array: 2

Right array: 1

Left array (merged): 2

Right array (merged): 1

After this we will merge and the merged array will be: 1 2

Left array (merged): 3

Right array (merged): 1 2

After this we will merge and the merged array will be: 1 2 3

Left array (merged): 4 5

Right array (merged): 1 2 3

After this we will merge and the merged array will be: 1 2 3 4 5

Left array (merged): 6 7 8 9 10

Right array (merged): 1 2 3 4 5

After this we will merge and the merged array will be: 1 2 3 4 5 6 7 8 9 10

Numer of comparisons is: 19

So in the end after sorting array is:

1 2 3 4 5 6 7 8 9 10

Merge sorting {5, 2, 13, 9, 1, 7, 6, 8, 1, 15, 4, 11}

Our current array that will be separated in left and right array is: 5 2 13 9 1 7 6 8 1 15 4 11

Left array: 5 2 13 9 1 7

Right array: 6 8 1 15 4 11

Our current array that will be separated in left and right array is: 5 2 13 9 1 7

Left array: 5 2 13

Right array: 9 1 7

Our current array that will be separated in left and right array is: 5 2 13

Left array: 5

Right array: 2 13

Our current array that will be separated in left and right array is: 2 13

Left array: 2

Right array: 13

Left array (merged): 2

Right array (merged): 13

After this we will merge and the merged array will be: 2 13

Left array (merged): 5

Right array (merged): 2 13

After this we will merge and the merged array will be: 2 5 13

Our current array that will be separated in left and right array is: 9 1 7

Left array: 9

Right array: 1 7

Our current array that will be separated in left and right array is: 1 7

Left array: 1

Right array: 7

Left array (merged): 1

Right array (merged): 7

After this we will merge and the merged array will be: 1 7

Left array (merged): 9

Right array (merged): 1 7

After this we will merge and the merged array will be: 1 7 9

Left array (merged): 2 5 13

Right array (merged): 1 7 9

After this we will merge and the merged array will be: 1 2 5 7 9 13

Our current array that will be separated in left and right array is: 6 8 1 15 4 11

Left array: 6 8 1

Right array: 15 4 11

Our current array that will be separated in left and right array is: 6 8 1

Left array: 6

Right array: 8 1

Our current array that will be separated in left and right array is: 8 1

Left array: 8

Right array: 1

Left array (merged): 8

Right array (merged): 1

After this we will merge and the merged array will be: 1 8

Left array (merged): 6

Right array (merged): 1 8

After this we will merge and the merged array will be: 1 6 8

Our current array that will be separated in left and right array is: 15 4 11

Left array: 15

Right array: 4 11

Our current array that will be separated in left and right array is: 4 11

Left array: 4

Right array: 11

Left array (merged): 4

Right array (merged): 11

After this we will merge and the merged array will be: 4 11

Left array (merged): 15

Right array (merged): 4 11

After this we will merge and the merged array will be: 4 11 15

Left array (merged): 1 6 8

Right array (merged): 4 11 15

After this we will merge and the merged array will be: 1 4 6 8 11 15

Left array (merged): 1 2 5 7 9 13

Right array (merged): 1 4 6 8 11 15

After this we will merge and the merged array will be: 1 1 2 4 5 6 7 8 9 11 13 15

Numer of comparisons is: 32

So in the end after sorting array is:

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

Merge sorting {'S', 'B', 'I', 'M', 'H', 'Q', 'C', 'L', 'R', 'E', 'P', 'K'}

Our current array that will be separated in left and right array is: S B I M H Q C L R E P K

Left array: S B I M H Q

Right array: C L R E P K

Our current array that will be separated in left and right array is: S B I M H Q

Left array: S B I

Right array: M H Q

Our current array that will be separated in left and right array is: S B I

Left array: S

Right array: B I

Our current array that will be separated in left and right array is: B I

Left array: B

Right array: I

Left array (merged): B

Right array (merged): I

After this we will merge and the merged array will be: B I

Left array (merged): S

Right array (merged): B I

After this we will merge and the merged array will be: B I S

Our current array that will be separated in left and right array is: M H Q

Left array: M

Right array: H Q

Our current array that will be separated in left and right array is: H Q

Left array: H

Right array: Q

Left array (merged): H

Right array (merged): Q

After this we will merge and the merged array will be: H Q

Left array (merged): M

Right array (merged): H Q

After this we will merge and the merged array will be: H M Q

Left array (merged): B I S

Right array (merged): H M Q

After this we will merge and the merged array will be: B H I M Q S

Our current array that will be separated in left and right array is: C L R E P K

Left array: C L R

Right array: E P K

Our current array that will be separated in left and right array is: C L R

Left array: C

Right array: L R

Our current array that will be separated in left and right array is: L R

Left array: L

Right array: R

Left array (merged): L

Right array (merged): R

After this we will merge and the merged array will be: L R

Left array (merged): C

Right array (merged): L R

After this we will merge and the merged array will be: C L R

Our current array that will be separated in left and right array is: E P K

Left array: E

Right array: P K

Our current array that will be separated in left and right array is: P K

Left array: P

Right array: K

Left array (merged): P

Right array (merged): K

After this we will merge and the merged array will be: K P

Left array (merged): E

Right array (merged): K P

After this we will merge and the merged array will be: E K P

Left array (merged): C L R

Right array (merged): E K P

After this we will merge and the merged array will be: C E K L P R

Left array (merged): B H I M Q S

Right array (merged): C E K L P R

After this we will merge and the merged array will be: B C E H I K L M P Q R S

Numer of comparisons is: 31

So in the end after sorting array is:

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

Quick sort

Perfrming quick sort now

Note that I took the first element as a pivot for each subarray here

Performing quick sort on a sorted array

First element of 'subarray' with lower and upper bound of: 0, 9 of the original array will be taken as pivot or array[lower_bound]:1 is the pivot

Partitioning the array(pivot:1)1 2 3 4 5 6 7 8 9 10

Swapping the pivot element: 0 with the element at 'end': 9 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 1 2 3 4 5 6 7 8 9 10

First element of 'subarray' with lower and upper bound of: 1, 9 of the original array will be taken as pivot or array[lower_bound]:2 is the pivot

Partitioning the array(pivot:2)2 3 4 5 6 7 8 9 10

Swapping the pivot element: 1 with the element at 'end': 9 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 2 3 4 5 6 7 8 9 10

First element of 'subarray' with lower and upper bound of: 2, 9 of the original array will be taken as pivot or array[lower_bound]:3 is the pivot

Partitioning the array(pivot:3)3 4 5 6 7 8 9 10

Swapping the pivot element: 2 with the element at 'end': 9 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 3 4 5 6 7 8 9 10

First element of 'subarray' with lower and upper bound of: 3, 9 of the original array will be taken as pivot or array[lower_bound]:4 is the pivot

Partitioning the array(pivot:4)4 5 6 7 8 9 10

Swapping the pivot element: 3 with the element at 'end': 9 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 4 5 6 7 8 9 10

First element of 'subarray' with lower and upper bound of: 4, 9 of the original array will be taken as pivot or array[lower_bound]:5 is the pivot

Partitioning the array(pivot:5)5 6 7 8 9 10

Swapping the pivot element: 4 with the element at 'end': 9 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 5 6 7 8 9 10

First element of 'subarray' with lower and upper bound of: 5, 9 of the original array will be taken as pivot or array[lower_bound]:6 is the pivot

Partitioning the array(pivot:6)6 7 8 9 10

Swapping the pivot element: 5 with the element at 'end': 9 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 6 7 8 9 10

First element of 'subarray' with lower and upper bound of: 6, 9 of the original array will be taken as pivot or array[lower_bound]:7 is the pivot

Partitioning the array(pivot:7)7 8 9 10

Swapping the pivot element: 6 with the element at 'end': 9 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 7 8 9 10

First element of 'subarray' with lower and upper bound of: 7, 9 of the original array will be taken as pivot or array[lower_bound]:8 is the pivot

Partitioning the array(pivot:8)8 9 10

Swapping the pivot element: 7 with the element at 'end': 9 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 8 9 10

First element of 'subarray' with lower and upper bound of: 8, 9 of the original array will be taken as pivot or array[lower_bound]:9 is the pivot

Partitioning the array(pivot:9)9 10

Swapping the pivot element: 8 with the element at 'end': 9 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 9 10

After sorting the total number of comparisons is: 54

Total number of displacements: 0

Array after sorting is:

1 2 3 4 5 6 7 8 9 10

Performing quick sort sort on a reversly sorted array

First element of 'subarray' with lower and upper bound of: 0, 9 of the original array will be taken as pivot or array[lower_bound]:10 is the pivot

Partitioning the array(pivot:10)10 9 8 7 6 5 4 3 2 1

Swapping the pivot element: 0 with the element at 'end': 9 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 1 9 8 7 6 5 4 3 2 10

First element of 'subarray' with lower and upper bound of: 0, 8 of the original array will be taken as pivot or array[lower_bound]:1 is the pivot

Partitioning the array(pivot:1)1 9 8 7 6 5 4 3 2

Swapping the pivot element: 0 with the element at 'end': 8 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 1 9 8 7 6 5 4 3 2

First element of 'subarray' with lower and upper bound of: 1, 8 of the original array will be taken as pivot or array[lower_bound]:9 is the pivot

Partitioning the array(pivot:9)9 8 7 6 5 4 3 2

Swapping the pivot element: 1 with the element at 'end': 8 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 2 8 7 6 5 4 3 9

First element of 'subarray' with lower and upper bound of: 1, 7 of the original array will be taken as pivot or array[lower_bound]:2 is the pivot

Partitioning the array(pivot:2)2 8 7 6 5 4 3

Swapping the pivot element: 1 with the element at 'end': 7 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 2 8 7 6 5 4 3

First element of 'subarray' with lower and upper bound of: 2, 7 of the original array will be taken as pivot or array[lower_bound]:8 is the pivot

Partitioning the array(pivot:8)8 7 6 5 4 3

Swapping the pivot element: 2 with the element at 'end': 7 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 3 7 6 5 4 8

First element of 'subarray' with lower and upper bound of: 2, 6 of the original array will be taken as pivot or array[lower_bound]:3 is the pivot

Partitioning the array(pivot:3)3 7 6 5 4

Swapping the pivot element: 2 with the element at 'end': 6 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 3 7 6 5 4

First element of 'subarray' with lower and upper bound of: 3, 6 of the original array will be taken as pivot or array[lower_bound]:7 is the pivot

Partitioning the array(pivot:7)7 6 5 4

Swapping the pivot element: 3 with the element at 'end': 6 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 4 6 5 7

First element of 'subarray' with lower and upper bound of: 3, 5 of the original array will be taken as pivot or array[lower_bound]:4 is the pivot

Partitioning the array(pivot:4)4 6 5

Swapping the pivot element: 3 with the element at 'end': 5 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 4 6 5

First element of 'subarray' with lower and upper bound of: 4, 5 of the original array will be taken as pivot or array[lower_bound]:6 is the pivot

Partitioning the array(pivot:6)6 5

Swapping the pivot element: 4 with the element at 'end': 5 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 5 6

After sorting the total number of comparisons is: 54

Total number of displacements: 5

Array after sorting is:

1 2 3 4 5 6 7 8 9 10

Performing quick sort on a given array {5, 2, 13, 9, 1, 7, 6, 8, 1, 15, 4, 11}

First element of 'subarray' with lower and upper bound of: 0, 11 of the original array will be taken as pivot or array[lower_bound]:5 is the pivot

Partitioning the array(pivot:5)5 2 13 9 1 7 6 8 1 15 4 11

Now swapping element in this subbaray at: 2 with the element at: 10

Now swapping element in this subbaray at: 3 with the element at: 8

Swapping the pivot element: 0 with the element at 'end': 11 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 1 2 4 1 5 7 6 8 9 15 13 11

First element of 'subarray' with lower and upper bound of: 0, 3 of the original array will be taken as pivot or array[lower_bound]:1 is the pivot

Partitioning the array(pivot:1)1 2 4 1

Now swapping element in this subbaray at: 1 with the element at: 3

Swapping the pivot element: 0 with the element at 'end': 3 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 1 1 4 2

First element of 'subarray' with lower and upper bound of: 2, 3 of the original array will be taken as pivot or array[lower_bound]:4 is the pivot

Partitioning the array(pivot:4)4 2

Swapping the pivot element: 2 with the element at 'end': 3 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 2 4

First element of 'subarray' with lower and upper bound of: 5, 11 of the original array will be taken as pivot or array[lower_bound]:7 is the pivot

Partitioning the array(pivot:7)7 6 8 9 15 13 11

Swapping the pivot element: 5 with the element at 'end': 11 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 6 7 8 9 15 13 11

First element of 'subarray' with lower and upper bound of: 7, 11 of the original array will be taken as pivot or array[lower_bound]:8 is the pivot

Partitioning the array(pivot:8)8 9 15 13 11

Swapping the pivot element: 7 with the element at 'end': 11 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 8 9 15 13 11

First element of 'subarray' with lower and upper bound of: 8, 11 of the original array will be taken as pivot or array[lower_bound]:9 is the pivot

Partitioning the array(pivot:9)9 15 13 11

Swapping the pivot element: 8 with the element at 'end': 11 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 9 15 13 11

First element of 'subarray' with lower and upper bound of: 9, 11 of the original array will be taken as pivot or array[lower_bound]:15 is the pivot

Partitioning the array(pivot:15)15 13 11

Swapping the pivot element: 9 with the element at 'end': 11 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 11 13 15

First element of 'subarray' with lower and upper bound of: 9, 10 of the original array will be taken as pivot or array[lower_bound]:11 is the pivot

Partitioning the array(pivot:11)11 13

Swapping the pivot element: 9 with the element at 'end': 10 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: 11 13

After sorting the total number of comparisons is: 39

Total number of displacements: 8

Array after sorting is:

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

Performing quick sort on a given array of characters {'S', 'B', 'I', 'M', 'H', 'Q', 'C', 'L', 'R', 'E', 'P', 'K'}

First element of 'subarray' with lower and upper bound of: 0, 11 of the original array will be taken as pivot or array[lower_bound]:S is the pivot

Partitioning the array(pivot:S)S B I M H Q C L R E P K

Swapping the pivot element: 0 with the element at 'end': 11 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: K B I M H Q C L R E P S

First element of 'subarray' with lower and upper bound of: 0, 10 of the original array will be taken as pivot or array[lower_bound]:K is the pivot

Partitioning the array(pivot:K)K B I M H Q C L R E P

Now swapping element in this subbaray at: 3 with the element at: 9

Now swapping element in this subbaray at: 5 with the element at: 6

Swapping the pivot element: 0 with the element at 'end': 10 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: C B I E H K Q L R M P

First element of 'subarray' with lower and upper bound of: 0, 4 of the original array will be taken as pivot or array[lower_bound]:C is the pivot

Partitioning the array(pivot:C)C B I E H

Swapping the pivot element: 0 with the element at 'end': 4 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: B C I E H

First element of 'subarray' with lower and upper bound of: 2, 4 of the original array will be taken as pivot or array[lower_bound]:I is the pivot

Partitioning the array(pivot:I)I E H

Swapping the pivot element: 2 with the element at 'end': 4 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: H E I

First element of 'subarray' with lower and upper bound of: 2, 3 of the original array will be taken as pivot or array[lower_bound]:H is the pivot

Partitioning the array(pivot:H)H E

Swapping the pivot element: 2 with the element at 'end': 3 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: E H

First element of 'subarray' with lower and upper bound of: 6, 10 of the original array will be taken as pivot or array[lower_bound]:Q is the pivot

Partitioning the array(pivot:Q)Q L R M P

Now swapping element in this subbaray at: 8 with the element at: 10

Swapping the pivot element: 6 with the element at 'end': 10 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: M L P Q R

First element of 'subarray' with lower and upper bound of: 6, 8 of the original array will be taken as pivot or array[lower_bound]:M is the pivot

Partitioning the array(pivot:M)M L P

Swapping the pivot element: 6 with the element at 'end': 8 index which is begining of elements bigger than pivot:

After 'partitioning' the subArray is: L M P

After sorting the total number of comparisons is: 41

Total number of displacements: 10

Array after sorting is:

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

Heap sort

Perfomrming heap sort now

Performing heap sort on a sorted array

I will use heapfy method to sort this array and make it heap before performing the sorting algorithm

Array is:

1 2 3 4 5 6 7 8 9 10

Array after heapify:

1 2 3 4 5 6 7 8 9 10

Swaping place of 10 with 5

Array after heapify:

1 2 3 4 10 6 7 8 9 5

Swaping place of 8 with 4

Array after heapify:

1 2 3 8 10 6 7 4 9 5

Swaping place of 10 with 3

Swaping place of 9 with 3

Array after heapify:

1 2 10 8 9 6 7 4 3 5

Swaping place of 10 with 2

Swaping place of 9 with 2

Swaping place of 5 with 2

Array after heapify:

1 10 9 8 5 6 7 4 3 2

Swaping place of 10 with 1

Swaping place of 9 with 1

Swaping place of 6 with 1

Array after heapify:

10 9 6 8 5 1 7 4 3 2

Our array now has a heap form:

10 9 6 8 5 1 7 4 3 2

Now array 'shrinking' and removing element by element in our max heap will be performed

10 9 6 8 5 1 7 4 3 2

Swapping 2 with 10

Swaping place of 9 with 2

Swaping place of 8 with 2

Swaping place of 7 with 2

After the swap and the heapify of subaray, array is

9 8 6 7 5 1 2 4 3 10

9 8 6 7 5 1 2 4 3 10

Swapping 3 with 9

Swaping place of 8 with 3

Swaping place of 7 with 3

Swaping place of 4 with 3

After the swap and the heapify of subaray, array is

8 7 6 4 5 1 2 3 9 10

8 7 6 4 5 1 2 3 9 10

Swapping 3 with 8

Swaping place of 7 with 3

Swaping place of 6 with 3

Swaping place of 5 with 3

After the swap and the heapify of subaray, array is

7 6 5 4 3 1 2 8 9 10

7 6 5 4 3 1 2 8 9 10

Swapping 2 with 7

Swaping place of 6 with 2

Swaping place of 5 with 2

Swaping place of 3 with 2

After the swap and the heapify of subaray, array is

6 5 3 4 2 1 7 8 9 10

6 5 3 4 2 1 7 8 9 10

Swapping 1 with 6

Swaping place of 5 with 1

Swaping place of 4 with 1

After the swap and the heapify of subaray, array is

5 4 3 1 2 6 7 8 9 10

5 4 3 1 2 6 7 8 9 10

Swapping 2 with 5

Swaping place of 4 with 2

Swaping place of 3 with 2

After the swap and the heapify of subaray, array is

4 3 2 1 5 6 7 8 9 10

4 3 2 1 5 6 7 8 9 10

Swapping 1 with 4

Swaping place of 3 with 1

Swaping place of 2 with 1

After the swap and the heapify of subaray, array is

3 2 1 4 5 6 7 8 9 10

3 2 1 4 5 6 7 8 9 10

Swapping 1 with 3

Swaping place of 2 with 1

After the swap and the heapify of subaray, array is

2 1 3 4 5 6 7 8 9 10

2 1 3 4 5 6 7 8 9 10

Swapping 1 with 2

After the swap and the heapify of subaray, array is

1 2 3 4 5 6 7 8 9 10

1 2 3 4 5 6 7 8 9 10

Swapping 1 with 1

After the swap and the heapify of subaray, array is

1 2 3 4 5 6 7 8 9 10

Total number of displacements: 39

Total number of comparisons: 90

Final array is:

1 2 3 4 5 6 7 8 9 10

Performing heap sort sort on a reversly sorted array

I will use heapfy method to sort this array and make it heap before performing the sorting algorithm

Array is:

10 9 8 7 6 5 4 3 2 1

Array after heapify:

10 9 8 7 6 5 4 3 2 1

Array after heapify:

10 9 8 7 6 5 4 3 2 1

Array after heapify:

10 9 8 7 6 5 4 3 2 1

Array after heapify:

10 9 8 7 6 5 4 3 2 1

Array after heapify:

10 9 8 7 6 5 4 3 2 1

Array after heapify:

10 9 8 7 6 5 4 3 2 1

Our array now has a heap form:

10 9 8 7 6 5 4 3 2 1

Now array 'shrinking' and removing element by element in our max heap will be performed

10 9 8 7 6 5 4 3 2 1

Swapping 1 with 10

Swaping place of 9 with 1

Swaping place of 8 with 1

Swaping place of 6 with 1

Swaping place of 2 with 1

After the swap and the heapify of subaray, array is

9 8 6 7 2 5 4 3 1 10

9 8 6 7 2 5 4 3 1 10

Swapping 1 with 9

Swaping place of 8 with 1

Swaping place of 7 with 1

Swaping place of 4 with 1

After the swap and the heapify of subaray, array is

8 7 6 4 2 5 1 3 9 10

8 7 6 4 2 5 1 3 9 10

Swapping 3 with 8

Swaping place of 7 with 3

Swaping place of 6 with 3

Swaping place of 5 with 3

After the swap and the heapify of subaray, array is

7 6 5 4 2 3 1 8 9 10

7 6 5 4 2 3 1 8 9 10

Swapping 1 with 7

Swaping place of 6 with 1

Swaping place of 5 with 1

Swaping place of 3 with 1

After the swap and the heapify of subaray, array is

6 5 3 4 2 1 7 8 9 10

6 5 3 4 2 1 7 8 9 10

Swapping 1 with 6

Swaping place of 5 with 1

Swaping place of 4 with 1

After the swap and the heapify of subaray, array is

5 4 3 1 2 6 7 8 9 10

5 4 3 1 2 6 7 8 9 10

Swapping 2 with 5

Swaping place of 4 with 2

Swaping place of 3 with 2

After the swap and the heapify of subaray, array is

4 3 2 1 5 6 7 8 9 10

4 3 2 1 5 6 7 8 9 10

Swapping 1 with 4

Swaping place of 3 with 1

Swaping place of 2 with 1

After the swap and the heapify of subaray, array is

3 2 1 4 5 6 7 8 9 10

3 2 1 4 5 6 7 8 9 10

Swapping 1 with 3

Swaping place of 2 with 1

After the swap and the heapify of subaray, array is

2 1 3 4 5 6 7 8 9 10

2 1 3 4 5 6 7 8 9 10

Swapping 1 with 2

After the swap and the heapify of subaray, array is

1 2 3 4 5 6 7 8 9 10

1 2 3 4 5 6 7 8 9 10

Swapping 1 with 1

After the swap and the heapify of subaray, array is

1 2 3 4 5 6 7 8 9 10

Total number of displacements: 30

Total number of comparisons: 72

Final array is:

1 2 3 4 5 6 7 8 9 10

Performing heap sort on a given array {5, 2, 13, 9, 1, 7, 6, 8, 1, 15, 4, 11}

I will use heapfy method to sort this array and make it heap before performing the sorting algorithm

Array is:

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

Array after heapify:

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

Swaping place of 11 with 7

Array after heapify:

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

Swaping place of 15 with 1

Array after heapify:

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

Array after heapify:

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

Swaping place of 15 with 13

Array after heapify:

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

Swaping place of 15 with 2

Swaping place of 13 with 2

Array after heapify:

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

Swaping place of 15 with 5

Swaping place of 13 with 5

Swaping place of 11 with 5

Swaping place of 7 with 5

Array after heapify:

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

Our array now has a heap form:

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

Now array 'shrinking' and removing element by element in our max heap will be performed

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

Swapping 5 with 15

Swaping place of 13 with 5

Swaping place of 11 with 5

Swaping place of 7 with 5

After the swap and the heapify of subaray, array is

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

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

Swapping 4 with 13

Swaping place of 11 with 4

Swaping place of 9 with 4

Swaping place of 8 with 4

After the swap and the heapify of subaray, array is

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

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

Swapping 1 with 11

Swaping place of 9 with 1

Swaping place of 8 with 1

Swaping place of 6 with 1

After the swap and the heapify of subaray, array is

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

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

Swapping 1 with 9

Swaping place of 8 with 1

Swaping place of 7 with 1

Swaping place of 5 with 1

After the swap and the heapify of subaray, array is

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

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

Swapping 4 with 8

Swaping place of 7 with 4

Swaping place of 6 with 4

After the swap and the heapify of subaray, array is

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

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

Swapping 1 with 7

Swaping place of 6 with 1

Swaping place of 5 with 1

Swaping place of 2 with 1

After the swap and the heapify of subaray, array is

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

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

Swapping 1 with 6

Swaping place of 5 with 1

Swaping place of 4 with 1

After the swap and the heapify of subaray, array is

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

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

Swapping 1 with 5

Swaping place of 4 with 1

Swaping place of 2 with 1

After the swap and the heapify of subaray, array is

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

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

Swapping 1 with 4

Swaping place of 2 with 1

After the swap and the heapify of subaray, array is

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

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

Swapping 1 with 2

After the swap and the heapify of subaray, array is

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

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

Swapping 1 with 1

After the swap and the heapify of subaray, array is

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

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

Swapping 1 with 1

After the swap and the heapify of subaray, array is

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

Total number of displacements: 43

Total number of comparisons: 100

Final array is:

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

Performing heap sort on a given array of characters {'S', 'B', 'I', 'M', 'H', 'Q', 'C', 'L', 'R', 'E', 'P', 'K'}

I will use heapfy method to sort this array and make it heap before performing the sorting algorithm

Array is:

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

Array after heapify:

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

Array after heapify:

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

Swaping place of R with H

Array after heapify:

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

Array after heapify:

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

Swaping place of R with I

Array after heapify:

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

Swaping place of R with B

Swaping place of Q with B

Swaping place of P with B

Array after heapify:

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

Array after heapify:

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

Our array now has a heap form:

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

Now array 'shrinking' and removing element by element in our max heap will be performed

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

Swapping K with S

Swaping place of R with K

Swaping place of Q with K

Swaping place of P with K

After the swap and the heapify of subaray, array is

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

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

Swapping B with R

Swaping place of Q with B

Swaping place of P with B

Swaping place of K with B

After the swap and the heapify of subaray, array is

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

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

Swapping E with Q

Swaping place of P with E

Swaping place of M with E

Swaping place of L with E

After the swap and the heapify of subaray, array is

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

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

Swapping H with P

Swaping place of M with H

Swaping place of L with H

After the swap and the heapify of subaray, array is

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

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

Swapping E with M

Swaping place of L with E

Swaping place of K with E

Swaping place of I with E

After the swap and the heapify of subaray, array is

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

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

Swapping C with L

Swaping place of K with C

Swaping place of I with C

Swaping place of E with C

After the swap and the heapify of subaray, array is

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

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

Swapping B with K

Swaping place of I with B

Swaping place of H with B

After the swap and the heapify of subaray, array is

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

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

Swapping C with I

Swaping place of H with C

Swaping place of E with C

After the swap and the heapify of subaray, array is

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

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

Swapping B with H

Swaping place of E with B

Swaping place of C with B

After the swap and the heapify of subaray, array is

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

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

Swapping B with E

Swaping place of C with B

After the swap and the heapify of subaray, array is

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

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

Swapping B with C

After the swap and the heapify of subaray, array is

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

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

Swapping B with B

After the swap and the heapify of subaray, array is

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

Total number of displacements: 41

Total number of comparisons: 96

Final array is:

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