HomeWork #6 Question 1

In this question There is 4 array and 4 sorting algorithm, so there are 4 shell sort, 4 merge sort, 4 heap sort and 4 quick sort.

SHELL SORT

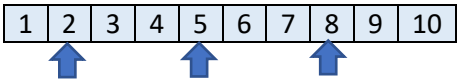
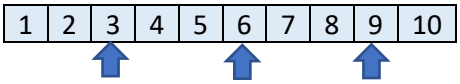
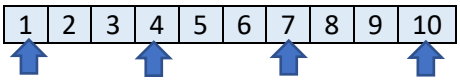
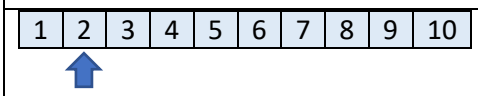
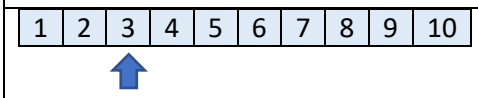
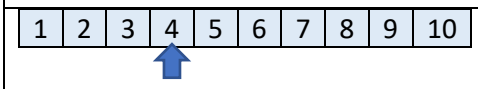
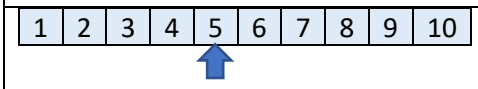
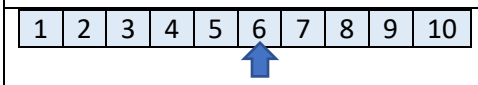
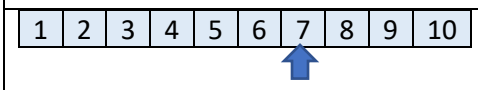
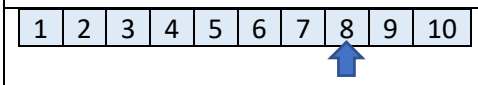
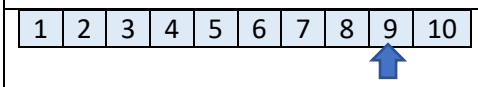
a) A is an ordered integer array with 10 elements from small to large

So I choose A array like this to Show Shell steps ;

1	2	3	4	5	6	7	8	9	10

I choose my first gap value 7 and divede it 2.2 like in lecture slayts.

ARRAY	Action	Comparison	Displacement
1 2 3 4 5 6 7 8 9 10	Because of my gap is 7, it compare index 0 and index 7 but didn't Exchange them because index 0 and index 7 sorted among themselves	1	0
1 2 3 4 5 6 7 8 9 10	Now, we compare index 1 and index 8 because our gap value still 7.But didn't Exchange them beucase index 1 and index 8 sorted among themselves	1	0
1 2 3 4 5 6 7 8 9 10	Now, we compare index 2 and index 9 because our gap value still 7.But didn't Exchange them beucase index 2 and index 9 sorted among themselves	1	0
1 2 3 4 5 6 7 8 9 10	I reach the end of array som y gap value diveded by 2.2,now my gap value is 3 so firstly I check index 0 and index 3.	1	0
1 2 3 4 5 6 7 8 9 10	Now go to the end of array 1 by 1 anc check some indexes according to gap value.Compara index 1 and index 4	1	0
1 2 3 4 5 6 7 8 9 10	Compare index 2 and index 5	1	0
1 2 3 4 5 6 7 8 9 10	Now compara 3 index because of our gap value 3 and we are in index 6,it just	1	0

	compare index 6 with index 3		
	İt just compare index 4 and index 7 and return because index 7 and index 4 already sorted.	1	0
	İt just compare index 5 and index 8 then return because index 8 greater than index 5 so dont need compare it with other indexes	1	0
	İt just compare index 9 and index 6 then return because index 9 is greater than index 6 so dont need compare it with other indexes	1	0
	Now we reach end of array so we divede our gap to 2.2. Our new gap is 1(3/2.2) so we check elements like insertion sort	1	0
	İt just compare index 2 with index 1 and stop because index 2 is greater than index 1	1	0
	İt just compare index 3 with index 2 and stop because index 3 is greater than index 2	1	0
	İt just compare index 4 with index 3 and stop because index 4 is greater than index 3	1	0
	İt just compare index 5 with index 4 and stop because index 5 is greater than index 4	1	0
	İt just compare index 6 with index 5 and stop because index 6 is greater than index 5	1	0
	İt just compare index 7 with index 6 and stop because index 7is greater than index 6	1	0
	İt just compare index 8 with index 7 and stop because index 8 is greater than index 7	1	0

1 2 3 4 5 6 7 8 9 10	İt just compare index 9 with index 8 and stop because index 9 is greater than index 8.Gap is 1 and we reach end of array our sort finished.	1	0

At The end total Displacement is 0 because array is already sorted and there are 19 comparisons

b) B is an ordered integer array with 10 elements from large to small

So I choose B array like this to Show Shell steps ;

10	9	8	7	6	5	4	3	2	1

I choose my first gap value 7 and divede it 2.2 like in lecture slayts.

ARRAY	Action	Comparison	Displacement
10 9 8 7 6 5 4 3 2 1	My first gap value 7 so I compare index 0 and index 7 and Exchange them because index 7 smaller then index 0	1	1
3 9 8 7 6 5 4 10 2 1	Now compare index 1 and index 8 and Exchange them because index 8 smaller than index 1	1	1
3 2 8 7 6 5 4 10 9 1	Compare index 2 and index 9 and Exchange them because index 9 smaller than index 2	1	1
3 2 1 7 6 5 4 10 9 8	We reach at the end of array now divede our gap 2.2 and start again our new gap is 3(7/2.2). Compare index 0 and index 3 but didn't Exchange them because index 3 bigger	1	0
3 2 1 7 6 5 4 10 9 8	Compare 2 and 6 but didnt exhange 6 because is bigger	1	0
3 2 1 7 6 5 4 10 9 8	Compare 1 and 5 but didnt Exchange because 5 is bigger	1	0

3 2 1 7 6 5 4 10 9 8 3 2 1 4 6 5 7 10 9 8	Compare 4 and 7 and ex change them because 4 is smaller.After that compare 4 and 3 but didnt Exchange because 4 bigger	2	1
3 2 1 4 6 5 7 10 9 8	Compare 10 with 6 but didn't change because 10 bigger	1	0
3 2 1 4 6 5 7 10 9 8	Compare 9 and 5 but didnt change because 9 bigger.	1	0
3 2 1 4 6 5 7 10 9 8	Compare 8 and 7 but didnt change because 8 bigger	1	0
3 2 1 4 6 5 7 10 9 8	We reach end of the array now divede gap 2.2.Our new gap is 1.Compare 2 and 3, 2 is smaller so Exchange them	1	1
2 3 1 4 6 5 7 10 9 8 2 3 3 4 6 5 7 10 9 8 1 2 3 4 6 5 7 10 9 8	Compare 1 and 3, 1 is smaller Exchange 1 and 3. Then compare 1 and 2, 1 still smaller than Exchange 1 and 2	2	2
1 2 3 4 6 5 7 10 9 8	Compare 4 and 3, 4 already bigger so don't do anything	1	0
1 2 3 4 6 5 7 10 9 8	Compare 6 and 4, 6 already bigger so don't Exchange and return	1	0
1 2 3 4 6 5 7 10 9 8 1 2 3 4 6 6 7 10 9 8 1 2 3 4 5 6 7 10 9 8	Compare 5 and 6, 5 is smaller than 6 then Exchange them.Then compare 5 and 4, 5 is bigger than 4 than dont Exchange them and return	2	1
1 2 3 4 5 6 7 10 9 8	Compare 7 and 6, 7 already bigger so didn't exchange	1	0

1 2 3 4 5 6 7 10 9 8	Compare 10 and 7, 10 already bigger so didn't exchange	1	0
1 2 3 4 5 6 7 10 9 8 1 2 3 4 5 6 7 10 10 8 1 2 3 4 5 6 7 9 10 8	Compare 9 and 10, 9 smaller so exhange them, Then compare 9 and 7 , 9 already bigger than 7 so didnt exchange	2	1
1 2 3 4 5 6 7 9 10 8 1 2 3 4 5 6 7 9 10 10 1 2 3 4 5 6 7 9 9 10 1 2 3 4 5 6 7 8 9 10	Compare 8 and 10, 8 smaller so Exchange them, After that compare 8 and 9, 8 smaller so Exchange them, After than compare 8 and 7, 8 bigger so didnt change them and finish sorting becaus our gap 1 and we reach end of the array	3	2

At The end total Displacement is 11 and there are 25 comparisons

c) C = {5, 2, 13, 9, 1, 7, 6, 8, 1, 15, 4, 11}

Lets first gap is 7.

ARRAY	Action	Comparison	Displacement
5 2 13 9 1 7 6 8 1 15 4 11	Compare 8 and 5, 8 is bigger than 5 so dont exchange	1	0
5 2 13 9 1 7 6 8 1 15 4 11 5 1 13 9 1 7 6 8 2 15 4 11	Compare 1 and 2, 1 smaller than 2 so exchange them	1	1
5 1 13 9 1 7 6 8 2 15 4 11	Compare 15 and 13, 15 bigger than 13 so don't exchange them	1	0
5 1 13 9 1 7 6 8 2 15 4 11 5 1 13 4 1 7 6 8 2 15 9 11	Compare 4 and 9, 4 smaller than 9 so exchange them	1	1

5 1 13 4 1 7 6 8 2 15 9 11	Compare 11 and 1, 11 is bigger so don't exchange them. We reach end of array so divide gap 2.2(7/2.2=3) and again begin sort array	1	0
5 1 13 4 1 7 6 8 2 15 9 11 4 1 13 5 1 7 6 8 2 15 9 11	Compare 4 and 5, 4 smaller so exchange them	1	1
4 1 13 5 1 7 6 8 2 15 9 11	Compare 1 and 1, they are equal so don't exchange	1	0
4 1 13 5 1 7 6 8 2 15 9 11 4 1 7 5 1 13 6 8 2 15 9 11	Compare 7 and 13, 7 smaller so exchange them	1	1
4 1 7 5 1 13 6 8 2 15 9 11	Compare 6 and 5, 6 bigger than 5 so dont exchange them	1	0
4 1 7 5 1 13 6 8 2 15 9 11	Compare 8 and 1, 8 bigger than 1 so don't exchange them	1	0
4 1 7 5 1 13 6 8 2 15 9 11 4 1 7 5 1 13 6 8 13 15 9 11 4 1 2 5 1 7 6 8 13 15 9 11	Compare 2 and 13, 2 smaller than 13 exchange them.After that Compare 2 and 7, 2 smaller then 7 so excange them too	2	2
4 1 2 5 1 7 6 8 13 15 9 11	Compare 15 and 6, 15 is bigger than 6 so dont exchange them	1	0
4 1 2 5 1 7 6 8 13 15 9 11	Compare 9 and 8, 9 bigger than	1	0

	8 so don't exchange them		
4 1 2 5 1 7 6 8 13 15 9 11 4 1 2 5 1 7 6 8 13 15 9 13 4 1 2 5 1 7 6 8 11 15 9 13	Compare 11 and 13 , 11 smaller than 13 then exchange them, After that compare 11 and 7, 11 bigger so stop. Lastly we reach end of array so divede gap 2.2 and begin sorting again	2	1
4 1 2 5 1 7 6 8 11 15 9 13 1 4 2 5 1 7 6 8 11 15 9 13	Our gap is 1 now. Compare 1 and 4, 1 is smaller so exchange them	1	1
1 4 2 5 1 7 6 8 11 15 9 13 1 4 4 5 1 7 6 8 11 15 9 13 1 2 4 5 1 7 6 8 11 15 9 13	Compare 2 and 4, 2 smaller than 4 then exchange them,After that compare 2 and 1, 2 bigger so stop	2	1
1 2 4 5 1 7 6 8 11 15 9 13	Compare 5 and 4, 5 bigger than 4 so dont exchange	1	0
1 2 4 5 1 7 6 8 11 15 9 13 1 2 4 5 5 7 6 8 11 15 9 13 1 2 4 1 5 7 6 8 11 15 9 13 1 2 4 4 5 7 6 8 11 15 9 13 1 2 1 4 5 7 6 8 11 15 9 13 1 2 2 4 5 7 6 8 11 15 9 13	Compare 1 and 5, 1 smaller so exchange them. After that compare 1 and 4, 1 is smaller so exchange them,After that compare 1 and 2, 1 smaller so exchange them, Finallay compare 1 and 1, they are equal so dont exchange	4	3

1 1 2 4 5 7 6 8 11 15 9 13			
1 1 2 4 5 7 6 8 11 15 9 13	Compare 7 and 5, 7 bigger so dont exchange	1	0
1 1 2 4 5 7 6 8 11 15 9 13 1 1 2 4 5 7 7 8 11 15 9 13 1 1 2 4 5 6 7 8 11 15 9 13	Compare 6 and 7, 6 smaller so exchange them. After that compare 6 and 5, 6 bigger so dont exchange them	2	1
1 1 2 4 5 6 7 8 11 15 9 13	Compare 8 and 7, 8 bigger so dont exchange	1	0
1 1 2 4 5 6 7 8 11 15 9 13	Compare 11 and 8, 11 bigger so dont exchange	1	0
1 1 2 4 5 6 7 8 11 15 9 13	Compare 15 and 11, 15 bigger so dont exchange	1	0
1 1 2 4 5 6 7 8 11 15 9 13 1 1 2 4 5 6 7 8 11 15 15 13 1 1 2 4 5 6 7 8 11 11 15 13 1 1 2 4 5 6 7 8 9 11 15 13	Compare 9 and 15, 9 smaller so exchange them, After that compare 9 and 11, 9 smaller so exchange them, then compare 9 and 8, 9 bigger so dont exchange them and stop	3	2
1 1 2 4 5 6 7 8 9 11 15 13 1 1 2 4 5 6 7 8 9 11 15 15 1 1 2 4 5 6 7 8 9 11 13 15	Compare 13 and 15, 13 smaller so exchange them, then compare 13 and 11, 13 bigger so dont exchange them and stop	2	1

At The end total Displacement is 16 and there are 35 comparisons

d-) D = {‘S’, ‘B’, ‘I’, ‘M’, ‘H’, ‘Q’, ‘C’, ‘L’, ‘R’, ‘E’, ‘P’, ‘K’}

I choose my first gap value 7 and divede it 2.2 like in lecture slayts.

ARRAY	Action	Comparison	Displacement
S B I M H Q C L R E P K	Compare L and S, L smalller than S so exchange them	1	1
L B I M H Q C S R E P K	Compare R and B, R bigger than B so dont exchange	1	0
L B I M H Q C S R E P K	Compare E and I, E smaller than I,so exchange them	1	1
L B E M H Q C S R I P K	Compare P and M, P bigger than M so dont exchange them	1	0
L B E M H Q C S R I P K	Compare K and H, K bigger than H so dont change them. Now we reach end of array so divide new gap 2.2 Gap=3(7/2.2)	1	0
L B E M H Q C S R I P K	New gap=3 Compare M and L,M bigger so dont exchange	1	0
L B E M H Q C S R I P K	Comprae H and B H bigger than B so dont exchange	1	0
L B E M H Q C S R I P K	Compare Q and E Q bigger than E so dont change them	1	0
L B E M H Q C S R I P K L B E M H Q M S R I P K C B E L H Q M S R I P K	Compare C and M, C smaller than M so exchange them,after that compare C and L, C smaller so exchange them	2	2
C B E L H Q M S R I P K	Compare S and H, S bigger so dont exchange	1	0
C B E L H Q M S R I P K	Compare R and Q, R bigger than Q so dont exchange them	1	0

C B E L H Q M S R I P K C B E L H Q M S R M P K C B E L H Q L S R M P K C B E I H Q L S R M P K	Compare I and M, I smaller so exchange them. After that compare I and L, I smaller so exchange them, lastly compare I and C, I bigger so stop	3	2
C B E I H Q L S R M P K C B E I H Q L S R M S K C B E I H Q L P R M S K	Compare P and S, P smaller than S so exchange them,After that compare P and H, P bigger so stop	2	1
C B E I H Q L P R M S K C B E I H Q L P R M S R C B E I H Q L P Q M S R C B E I H K L P Q M S R	Compare K and R, K smaller so exchange them, After that compare K and Q, K smaller so exchange them,After that compare K and E, K bigger so stop	3	2
C B E I H K L P Q M S R B C E I H K L P Q M S R	Gap is now 1 (3/2.2) Compare B and C, B smaller then exchange them	1	1
B C E I H K L P Q M S R	Compare E and C,E bigger so dont exchange	1	0
B C E I H K L P Q M S R	Compare I and E, I bigger so dont exchange	1	0
B C E I H K L P Q M S R B C E I I K L P Q M S R B C E H I K L P Q M S R	Compare H and I, H smaller so exchange them , After that compare H and E, H bigger so stop exchanging	2	1
B C E H I K L P Q M S R	Compare K and I, K bigger so dont exchange	1	0
B C E H I K L P Q M S R	Compare L and K L bigger so dont exchange	1	0

B C E H I K L P Q M S R	Compare P and L, P bigger so dont exchange	1	0
B C E H I K L P Q M S R	Compare Q and P, Q bigger than P so dont exchange	1	0
B C E H I K L P Q M S R B C E H I K L P Q Q S R B C E H I K L P P Q S R B C E H I K L M P Q S R	Compare M and Q, M smaller so exchange them, after that compare M and P, M smaller so exchange them,After that compare M and L, M bigger so stop	3	2
B C E H I K L M P Q S R	Compare S and Q, S bigger so dont exchange them	1	0
B C E H I K L M P Q S R B C E H I K L M P Q S S B C E H I K L M P Q R S	Compare R and S, R is smaller than S, so exchange them, After that compare R and Q, R bigger so stop	2	1

At The end total Displacement is 14 and there are 35 comparisons

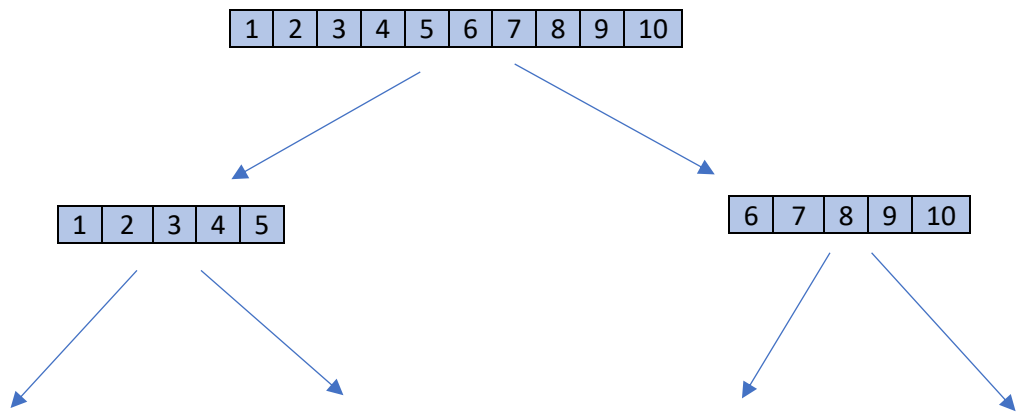
MERGE SORT

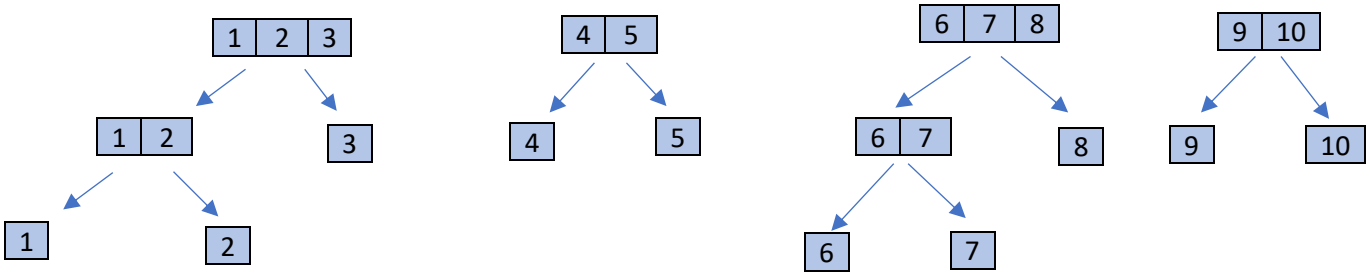
a) A is an ordered integer array with 10 elements from small to large

So I choose A array like this to Show Merge Sort steps ;

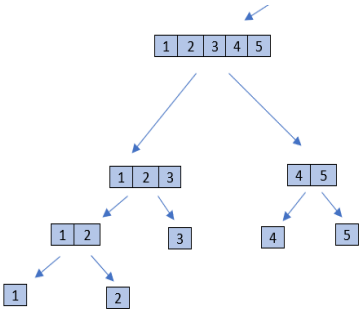
1	2	3	4	5	6	7	8	9	10

Merge Tree:

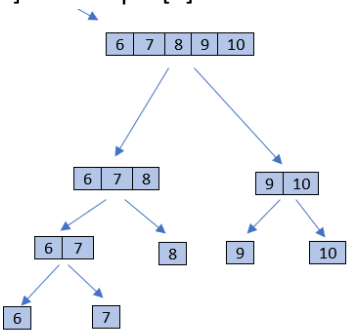




Tree steps:



- Left array is [1], right array is [2], compare 1 and 2 then copy 1 to output[0] because its smaller than 2, then copy 2 to output[1].
- Now left array is [1,2], right array is [3], compare 1 and 3 and copy 1 to output[0] because 1 smaller than 3, then compare 2 and 3 and copy 2 to output[1] because 2 smaller than 3,lastly just copy 3 into output[2] without a comparision.
- Now lets build right tree [4,5], here left tree [4] and right tree [5] , compare 4 and 5, here 4 smaller than 5 so copy 4 to output[0] and copy 5 to output[1].
- Then our left tree is [1,2,3] and right tree [4,5],
- now compare first elements 1 and 4 then copy 1 to output[0] because its smaller than 4, then compare 2 and 4 and copy 2 to output[1] because 2 smaller than 4,
- compare 3 and 4 and copy 3 to output[2] because 3 smaller than 4
- Then just copy 4 and 5 to output[3] and output[4].



-

- Now sort [6,7,8,9,10] array
- Firstly, sort [6] and [7], compare 6 and 7, 6 smaller than 7 so output[0] is 6 and output[1] is 7
- Now left array[6,7] right array [8], compare 6 and 8, 6 smaller than 8 so output[0] is 6

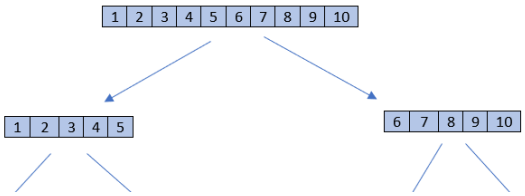
Compare 7 and 8, 7 smaller than 8 so copy 7 to output[1] and copy 8 to output[2].

- Now sort left [9] and right [10] array, compare 9 and 10, 9 smaller than 10 so copy 9 to output[0] and copy 10 to output[1].
- Now our left array is [6,7,8] and right array is [9,10] lets merge them, compare 6 and 9, 6 smaller than 9 so copy 6 to output[6]

Compare 7 and 9, 7 smaller than 9 so copy 7 to output[1],

Compare 8 and 9, 8 smaller than 9 so copy 8 to output[2],

Left array is finished then just copy right array to output so copy 9 to output[3] and 10 to output[4].



Now sort last recursive left and right array and return, after this last merge our array became sorted.

-Left array smallest is 1 and right array smallest element is 6, lets compare 1 and 6, 1 is smaller than 6 so copy 1 to output[0],

Compare 2 and 6, copy 2 to output[1]

Compare 3 and 6, copy 3 to output[2]

Compare 4 and 6,copy 4 to output[3]

Compare 5 and 6, copy 5 to output[4]

One of the our arrays finished so just copy other array elements to output array.

Copy 6 to output[5], copy 7 to output[6], copy 8 to output[7], copy 9 to output[8], copy[10] to output[9].

Return output array its out sorted array.

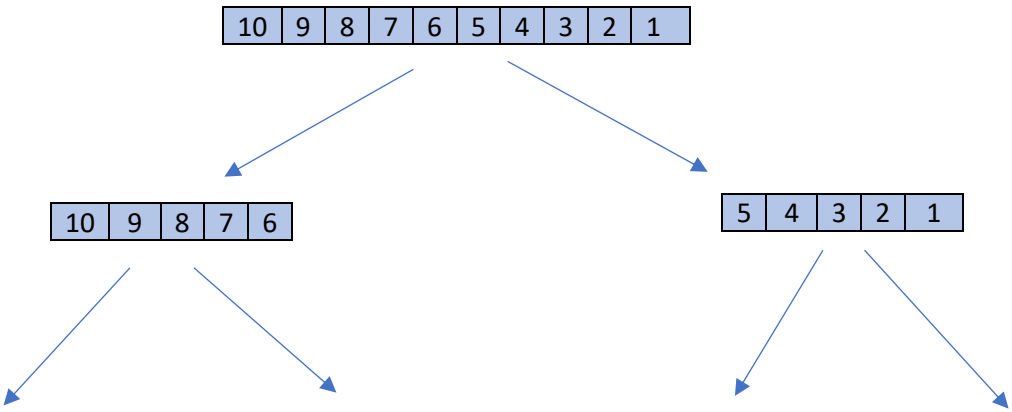
Total compare = 19 , Total copy = 34

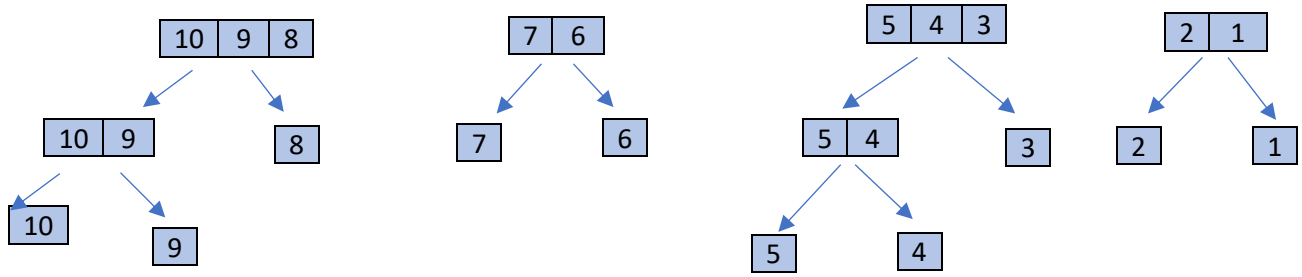
b) B is an ordered integer array with 10 elements from large to small

So I choose B array like this to Show Merge Sort steps ;

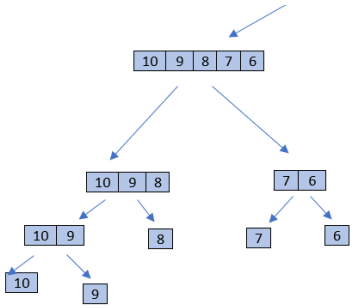
10	9	8	7	6	5	4	3	2	1

Merge Tree:





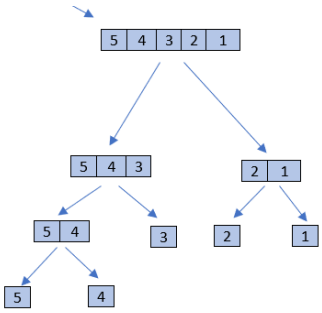
Tree steps:



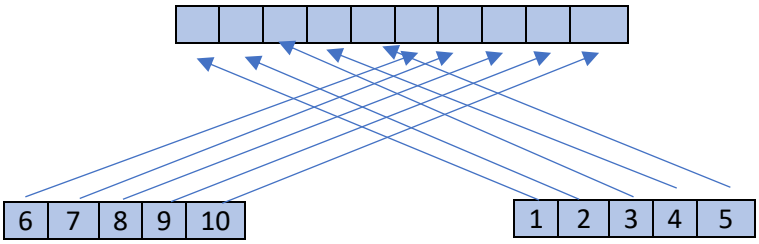
Let first sort this branch

- Left array [10] and right array[9], compare 10 and 9, copy 9 to output[0] because its smaller than 10. After that copy 10 to output[1].
- Now left array become [9,10] and right array [8], left array smallest element is 9 right smallest element 8 so compare 9 and 8, 8 is smaller so copy 8 to output[0],then copy 9 to output[1] and 10 to output[2], so array became [8,9,10]
- Lest sort right [7,6] array, in these arrays left array [7] and right array is [6], compare 6 and 7, 6 smaller so copy output[0] to 6, then copy 7 output[1], after that our output array became[6,7].
- Now our left array [8,9,10] and right array [6,7], left array smallest element 8 and right array smallest element 6, compare 6 and 8, 8 smaller so copy it to output[0]
  Compare 7 to 8, 7 smaller so copy it to output[1]
  One of the array finish so copy all other array elements to output so
  Output[2]=8,output[3]=9, output[4]=10.
  İn the end output array become [6,7,8,9,10].

Lest sort other branch



- Left array [5] and right array [4], compare 5 and 4, 5 is smaller then copy 4 to output[0] and copy 5 to output[1].
- Now left array become[4,5] and right array [3], smallest element of left array is 4 and smallest element of right array is 3 so compare 3 and 4, 3 smaller copy it to output[0],then one of the array finish, just copy all elements of the other array to output so, output[1]=4 and output[2]=5.Now left output become [3,4,5].
- Lets sort right [2,1] array, compare 2 and 1, 1 is smaller so copy 1 to output[0] and copy 2 to output[1].
- Now our left array [3,4,5] and right array[1,2]
- Left smallest array element 3 and right smallest array element 1, compare 1 and 3, 1 smaller so copy it to output[0],
  Compare 2 and 3, 2 smaller copy it to output[1],
  Right array finish so just copy left array element to output array so,
  output[2]=3, output[3]=4, output[4]=5.
- In the end, my output array become [1,2,3,4,5].

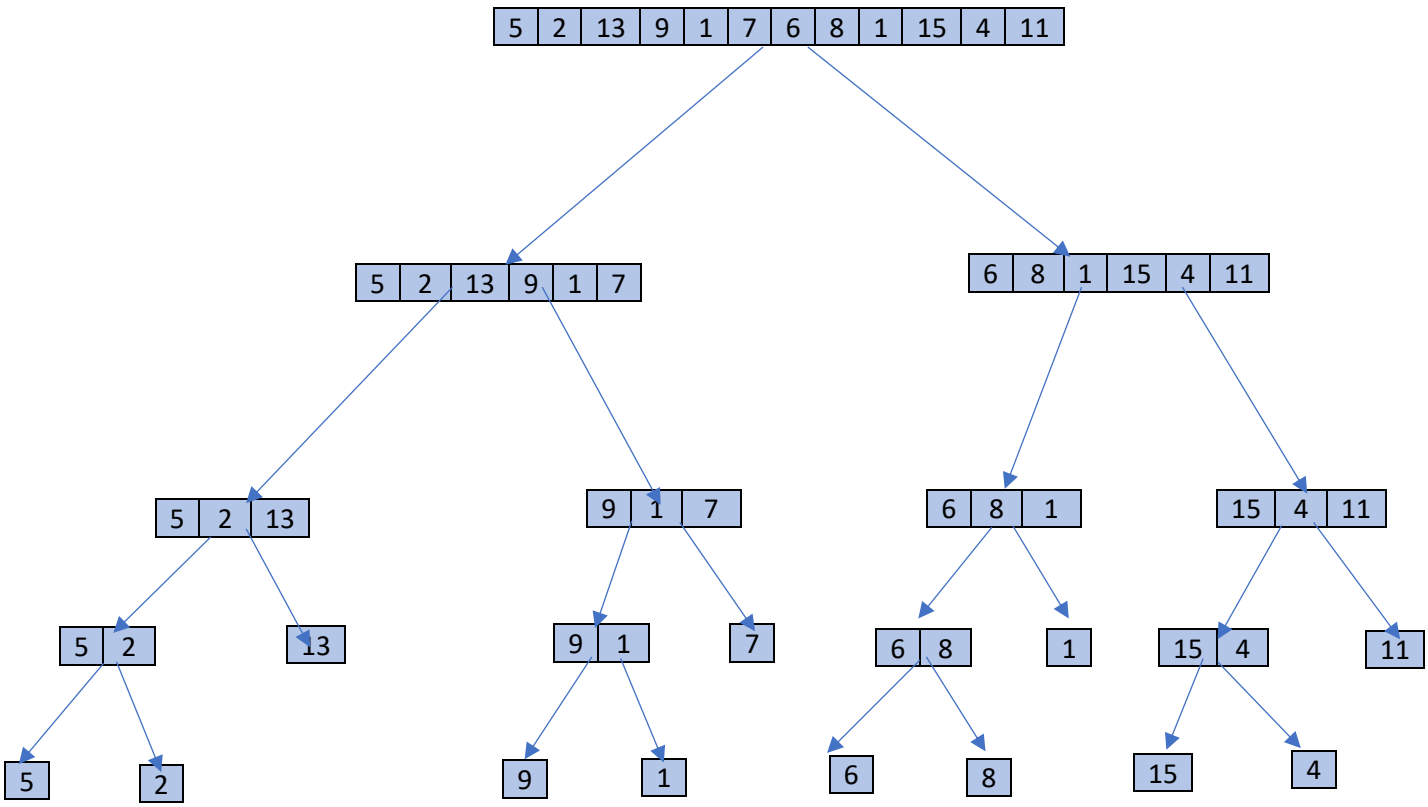


- Lastly we sort left array and right array now we must compare its elements and sort our output array and return sorted array.
- Left array smallest element 6 and right array smallest is 1, so compare 1 and 6, 1 smaller so copy 1 to output[0], compare 2 and 6, copy 2 to output[1], compare 3 and 6 copy 3 to output[2], compare 4 and 6, copy 4 to output[3], compare 5 and 6, copy 5 to output[4].
- Our right array finished so just copy all left array elements to output so, output[5]=6
- Output[6]=7, output[7]=8, output[8]=9, output[9]=10.
- In the end our output array become [1,2,3,4,5,6,7,8,9,10] and return it.

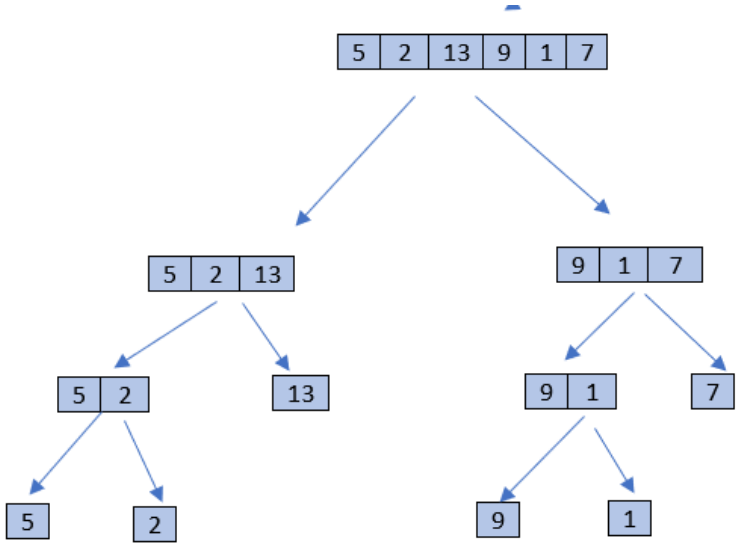
Total compare = 15, Total copy = 34

c) C = {5, 2, 13, 9, 1, 7, 6, 8, 1, 15, 4, 11}

Merge Tree:



Tree Steps:



-Let start with left branch of tree. First we have left array [5] and right array[2]. Compare 2 and 5, 2 smaller so copy 2 to output[0] and copy 5 to output[1]. Now our output array is [2,5].

-Now out left array [5,2] and right array [13], left array smallest element 2 and right array smallest element 13, so compare 2 and 13, 2 smaller then 13 then copy 2 to output[0], then compare 5 and 13, 5 smaller so copy 5 to output[1] and copy 13 output[2],

-Now our left array become [2,5,13], now lets merge sort right branch.

-in right branch highest level we have [9] as left array and [1] as right array, Compare 1 and 9, 1 smaller than 9 so copy 1 to output[0] then compy 9 to output[1]. Now our array become [1,9].

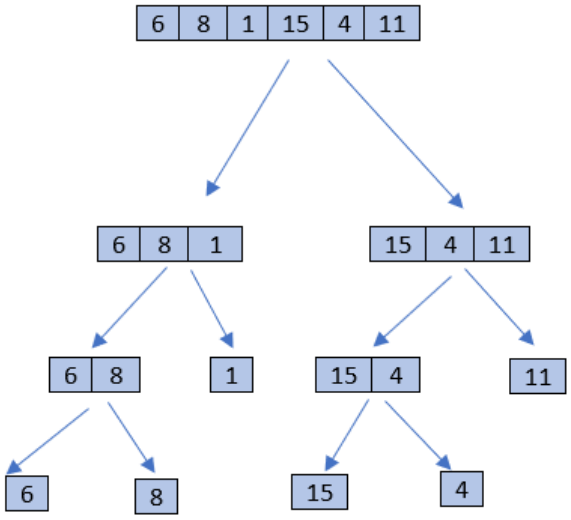
-Now left array is [1,9] and right array is [7], smallest element of left array is 1 and smallest element of right array is 7, compare 1 and 7, 1 smaller so copy it to output[0], then compare 7 and 9, 7 smaller so copy it to output[1], lastly copy 9 to output[2].

-Now our left array is [2,5,13] and right array is [1,7,9]. Left array smallest element is 2 and right array smallest element 1, compare 1 and 2, 1 smaller so copy it to output[0].

Compare next smallest element of left array (2) with 7, 2 smaller then copy it to output[1].

Compare 5 and 7, 5 smaller so copy it to output[2], Compare 13 and 7, 7 smaller copy it to output[3]

Compare 13 and 9, 9 smaller copy it to output[4], then copy 13 to output[5]. Now our left branch array [1,2,5,7,9,13].



-Now, lets merge sort right branch of tree.

-firstly highest level left side we have array [6] as left array and array [8] as right array. Compare 6 and 8, 6 smaller so copy 6 to output[0], then copy 8 to output[1].

-now our left array is [6,8] and right array is [1], left array smallest element is 6 and right array smallest element is 1 so compare 1 and 6, 1 smaller so copy it to output[0], copy 6 to output[1] and copy 8 to output[2]. Out array become [1,6,8].

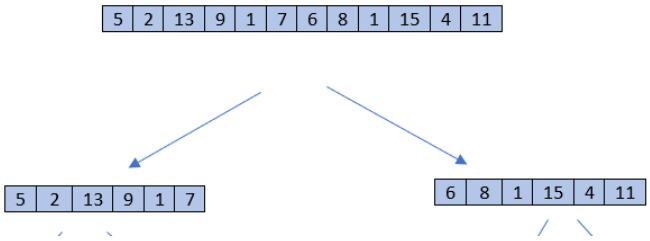
-Now focus right branch, in the higest level of right branch we have array [15] as the left array and array [4] as the right array, compare 4 and 15, 4 is smaller so copy it to output[0] then copy 15 to output[1]. Output become[4,15].

-now, left array [4,15] right array [11], comapre 4 and 11, 4 is smaller so copy it to output[0] , then comapre 15 and 11, 11 smaller so copy it to output[1], lastly copy 15 to output[2].

-Lastly out left array was [1,6,8] now we found our right array as [4,11,15]

Smallest element of the left array is 1 and smallest element of right array is 4, so compare 1 and 4 then copy 1 to output[0], compare 6 and 4, 4 smaller copy it to output[1], compare 6 and 11,6 smaller copy it to output[2], compare 8 and 11, 8 smaller copy it to output[3], copy 11 to output[4] and lastly copy 15 to output[5].

Now our right output array become [1,4,6,8,11,15].



Lastly we have [1,2,5,7,9,13] as left array and [1,4,6,8,11,15] as right array. Now merge them and find our last output array and return it. This will be sorted result array.

- Compare 1 and 1, they are equal so copy right array 1 to output[0]
- Compare 1 and 4, 1 smaller so copy it to output[1]
- Compare 2 and 4, 2 smaller so copy it to output[2]
- Compare 5 and 4,4 smaller so copy it to output[3]
- Compare 5 and 6, 5 smaller so copy it to output[4]

- Compare 7 and 6 ,6 smaller so copy it to output[5]
- Compare 7 and 8, 7 smaller so copy it to output[6]
- Compare 9 and 8,8 smaller so copy it to output[7]
- Compare 9 and 11, 9 is smaller,copy it to output[8]
- Compare 13 and 11, 11 is smaller copy it to output[9]
- Compare 13 and 15, 13 smaller copy it to output[10]
- Copy 15 to output[11]

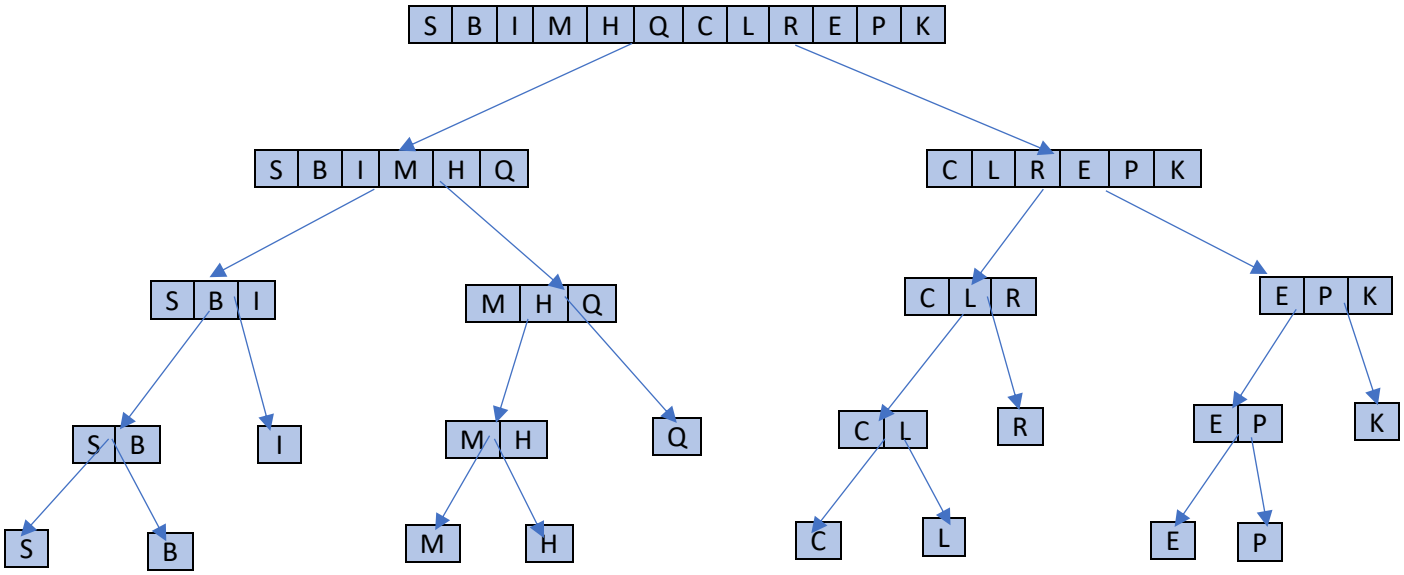
In the end our output array become [1,1,2,4,5,6,7,8,9,11,13,15]

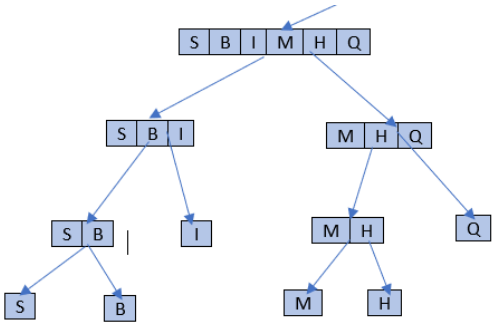
Total compare = 31, Total copy = 44

d-) D = {'S', 'B', 'I', 'M', 'H', 'Q', 'C', 'L', 'R', 'E', 'P', 'K'}

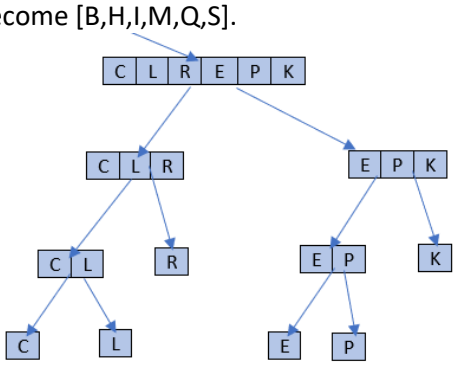
myArray:

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



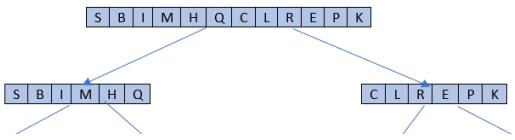


Lets begin with left tree branch.
-Highest level of left tree we have [S] as left array and [B] as right array. Compare S and B
B is smaller so copy B to output[0] and copy S to output[1]. Output [B,S]
-Now in left we have [B,S] and right [I], compare B and I, B smaller so copy B to output[0], now compare S and I, I smaller so copy it to output[1], lastly just copy S to output[2].
Now our left array become [B,I,S]
-Now lets find right array. In right side highest levet we have [M] and [H], compare M and H
H is smaller so copy it to output[0], then copy M to output[1].now left array [H,M] and right arrah [Q]. Compare H and Q, H smaller so copy it to output[0], then compare M and Q, M smaller copy it to output[1], lastly jsut copy Q to otuput[2].
-Now our left array [B,I,S] and right array is [H,M,Q]. Compare B and H, B smaller copy it to output[0], compare I and H, H smaller copy it to output[1], compare I and M, I smaller copy it to output[2], compare S and M, M smaller copy it to output[3], compare S and Q, Q smaller copy it to output[4], copy S to output[5].
- In the end our left array become [B,H,I,M,Q,S].



Now lest find right array.
-First start left side array [C] and array [L], compare C and L, C smaller so copy C to output[0],and copy L to output[1].
now left array is [C,L] and right array is [R], compare C and R, C smaller than R so copy C to output[0], Compare L and R, L smaller so copy it to output[1], copy R to output[2].
-Now our left array become[C,L,R] now find right array, In the right branch there is left array [E] and right array [P], Compare E and P, E smaller so copy it to output[0] and copy P to output[1]. Now left array become [E,P] and right array [K], Compare E and K, E smaller so copy it to output[0], then compare P and K, K smaller copy iy yo output[1], then copy P to output[2]. Now our right array become[E,K,P].
-Our left array was [C,L,R] and our right array is [E,K,P], first
Compare C and E, C smaller so copy it to output[0],
Compare L and E, E smaller so copy it to output[1],
Compare L and K, K smaller so copy it to output[2],
Compare L and P, L smaller so copy it to output[3],
Compare R and P, P smaller so copy it to output[4],
Copy R to output[5],

Now we found right array as [C,E,K,L,P,R]



-Now our left array is [B,H,I,M,Q,S] and right array is [C,E,K,L,P,R]. Lets merge sort them and output will be out sorted array result.
-Compare B and C, B smaller so copy it to output[0],
-Compare H and C, C smaller so copy it to output[1],
-Compare H and E, E smaller so copy it to output[2],
-Compare H and K, H smaller so copy it to output[3],
-Compare I and K, I smaller so copy it to output[4],
-Compare M and K, K smaller so copy it to output[5],
-Compare M and L, L smaller so copy it to output[6],
-Compare M and P, M is smaller so copy it to output[7],
-Compare Q and P, P is smaller so copy it to ouput[8],
-Compare Q and R, Q is smaller so copy it to output[9],
-Compare S and R, R is smaller so copy it to output[10],
-Copy S to output[11]

-Finally we sort out array result is output array [B,C,E,H,I,K,L,M,P,Q,R,S]
Total Compare=33
Total Displacement=44

Heap Sort

a) A is an ordered integer array with 10 elements from small to large

So I choose A array like this to Show Shell steps ;

1	2	3	4	5	6	7	8	9	10

For heap sort we must build a heap with this array then sort it.

Build heap:

Heap	Action
1 2 3 4 5 6 7 8 9 10 2 1 3 4 5 6 7 8 9 10	Lets say index 0 our heap parent then start with index 1, add 2 to the heap,2's parent is 1 so according to max heap priority exchange 1 and 2
2 1 3 4 5 6 7 8 9 10 3 1 2 4 5 6 7 8 9 10	3 parent is 2 compare 2 and 3, because of 3 bigger than 2 then exchange them

3 1 2 4 5 6 7 8 9 10 3 4 2 1 5 6 7 8 9 10 4 3 2 1 5 6 7 8 9 10	4 parent is 1, compare 1 and 4, 4 bigger than 1 so exchange them.After that 4 parent is 3, compare 3 and 4, because of 4 greater than 3 exchange them
4 3 2 1 5 6 7 8 9 10 4 5 2 1 3 6 7 8 9 10 5 4 2 1 3 6 7 8 9 10	5 parent is 3 so compare 3 and 5, because of 5 greater than 3 exchange them. Then 5 parent is 4, compare 5 and 4 and because of 5 greater then 4 exchange them.
5 4 2 1 3 6 7 8 9 10 5 4 6 1 3 2 7 8 9 10 6 4 5 1 3 2 7 8 9 10	6 parent is 2, compare 6 and 2, 6 bigger than 2 so exchange them,after that 6 new parent is 5, compare 6 and 5, still 6 greater so exchange them too.
6 4 5 1 3 2 7 8 9 10 6 4 7 1 3 2 5 8 9 10 7 4 6 1 3 2 5 8 9 10	7 parent is 5, compare 7 and 5, 7 bigger than 5 so exchange them,After that exchange 7 new parent is 6, compare 6 and 7, because of 7 greater than 6 exchange them too.
7 4 6 1 3 2 5 8 9 10 7 4 6 8 3 2 5 1 9 10 7 8 6 4 3 2 5 1 9 10 8 7 6 4 3 2 5 1 9 10	8 parent is 1, comapre 8 and 1, because of 8 greater than 1 exchange them, After that 8 new parent is 4, compare 4 and 8, beacuse of 8 greater then 4 exchange them too, Finally compare 8 with its new parent 7, 8 still greater then 7 so exchange them too.
8 7 6 4 3 2 5 1 9 10 8 7 6 9 3 2 5 1 4 10 8 9 6 7 3 2 5 1 4 10 9 8 6 7 3 2 5 1 4 10	Compare 9 with its parent 4, 9 bigger than 4 then exchange them, after that compare 9 with its new parent 7, 9 bigger than 7 so exchange them,finally compare 9 with 8, 9 bigger then 8 so exchange them and stop.

9 8 6 7 3 2 5 1 4 10 9 8 6 7 10 2 5 1 4 3 9 10 6 7 8 2 5 1 4 3 10 9 6 7 8 2 5 1 4 3	Compare 10 with its parent 3,10 bigger than 3 so exchange them,after than compare 10 with its new parent 8,10 bigger than 8 so exchange them,finally compare 10 and its new parent 9,still 10 bigger so exchange them and stop

Now we build our heap

10	9	6	7	8	2	5	1	4	3

We must ShrinkHeap now and result will be our sorted array.

Heap	Action
10 9 6 7 8 2 5 1 4 3 3 9 6 7 8 2 5 1 4 10 9 3 6 7 8 2 5 1 4 10 9 8 6 7 3 2 5 1 4 10	Remove first element and add last element and move last element(3) to first element,After that we must reorginaze our array for max heap priority, -Compare 3 with its max child(9), 3 smaller so exchange -Compare 3 with its max child(8),3 smaller so exchange them
4 8 6 7 3 2 5 1 9 10 8 4 6 7 3 2 5 1 9 10 8 7 6 4 3 2 5 1 9 10	Remove first element and add last element and move last element(4) to first element,After that we must reorginaze our array for max heap priority, -Compare 4 with its max child(8),4 smaller so exchange them. -Compare 4 with its max child(7),4 smaller so exchange them. -Compare 4 with its max child(1), 4 bigger so dont exchange
1 7 6 4 3 2 5 8 9 10 7 1 6 4 3 2 5 8 9 10 7 4 6 1 3 2 5 8 9 10	Remove first element and add last element and move last element(1) to first element,After that we must reorginaze our array for max heap priority, -Compare 1 with its max child(7),1 smaller so exchange them -Compare 1 with its max child(4),1 smaller then 4 exchange them.
5 4 6 1 3 2 7 8 9 10	Remove first element and add last element and move last element(5) to first element,After that we must

6 4 5 1 3 2 7 8 9 10	reorginaze our array for max heap priority, -Compare 5 with its max child(6),5 smaller so exchange the. -Compare 5 with 2, 5 bigger so dont exchange.
2 4 5 1 3 6 7 8 9 10 5 4 2 1 3 6 7 8 9 10	Remove first element and add last element and move last element(2) to first element,After that we must reorginaze our array for max heap priority, -Compare 2 with its max child(5),2 smaller so exchange them.
3 4 2 1 5 6 7 8 9 10 4 3 2 1 5 6 7 8 9 10	Remove first element and add last element and move last element(3) to first element,After that we must reorginaze our array for max heap priority, -compare 3 with its max child(4),3 smaller then 3 so exchange them. -Compare 3 with its max child(1), 3 bigger so dont exchange.
1 3 2 4 5 6 7 8 9 10 3 1 2 4 5 6 7 8 9 10	Remove first element and add last element and move last element(1) to first element,After that we must reorginaze our array for max heap priority, -Compare 1 with its max child(3),1 smaller so exchange them.
2 1 3 4 5 6 7 8 9 10	Remove first element and add last element and move last element(2) to first element,After that we must reorginaze our array for max heap priority, -compare 2 with its max child(1), 2 bigger so dont exchange them.
1 2 3 4 5 6 7 8 9 10	Just remove first element and add it too.
1 2 3 4 5 6 7 8 9 10	Result

Total Compare=30
Total Displacement=35

B) B is an ordered integer array with 10 elements from large to small

So I choose B array like this to Show Shell steps ;

10	9	8	7	6	5	4	3	2	1

For heap sort we must build a heap with this array then sort it.

First build heap:

Heap	Action
10 9 8 7 6 5 4 3 2 1	Lets say index 0 our heap parent then start with index 1, add 9 to the heap,9's parent is 10,so 9 is smaller than 10 dont exchange
10 9 8 7 6 5 4 3 2 1	Compare 8 with its parent 10, 8 smaller than 10 so dont exchange
10 9 8 7 6 5 4 3 2 1	Compare7 with its parent 9, 7 smaller than 9 so dont exchange them
10 9 8 7 6 5 4 3 2 1	Compare 6 with its parent 9,6 smaller than 9 so dont exchange them.
10 9 8 7 6 5 4 3 2 1	Compare 5 with its parent 8, 5 smaller than 8 so dont exchange.
10 9 8 7 6 5 4 3 2 1	Compare 4 with its parent 8,4 smaller so dont exchange them.
10 9 8 7 6 5 4 3 2 1	Compare 3 with its parent 7,3 smaller so dont exchange them.
10 9 8 7 6 5 4 3 2 1	Compare 2 with its parent 7, 2 smaller so dont exchange them.
10 9 8 7 6 5 4 3 2 1	Compare 1 with its parent 6, its smaller then 6 so dont exchange.

Now we must Shrink heap then result will be our sorted array.

Heap	Action
1 9 8 7 6 5 4 3 2 10 9 1 8 7 6 5 4 3 2 10 9 7 8 1 6 5 4 3 2 10 9 7 8 3 6 5 4 1 2 10	Remove first element and add last element and move last element(1) to first element,After that we must reorginaze our array for max heap priority, -Compare 1 to its max child(9),1 smaller so exchange them -Compare 1 to its max child(7),1 smaller so exchange them -Compare 1 to its max child(3),1 smaller so exchange them
2 7 8 3 6 5 4 1 9 10 8 7 2 3 6 5 4 1 9 10	Remove first element and add last element and move last element(2) to first element,After that we must reorginaze our array for max heap priority,

8 7 5 3 6 2 4 1 9 10	-Compare 2 to its max child(8),2 smaller so exchange them. -compare 2 with its max child(5),2 smaller so exchange them.
1 7 5 3 6 2 4 8 9 10 7 1 5 3 6 2 4 8 9 10 7 6 5 3 1 2 4 8 9 10	Remove first element and add last element and move last element(1) to first element,After that we must reorginaze our array for max heap priority, -Compare 1 with its max child(7),1 smaller so exchange them. -Compare 1 with its max child(6),1 smaller so exchange them.
4 6 5 3 1 2 7 8 9 10 6 4 5 3 1 2 7 8 9 10	Remove first element and add last element and move last element(4) to first element,After that we must reorginaze our array for max heap priority, -Compare 4 with its max child(6),4 smaller so exchange them. -Compare 4 with its max child(3),4 bigger so dont exchange
2 4 5 3 1 6 7 8 9 10 5 4 2 3 1 6 7 8 9 10	Remove first element and add last element and move last element(2) to first element,After that we must reorginaze our array for max heap priority, -Compare 2 with its max child(5),2 smaller so exchange them.
1 4 2 3 5 6 7 8 9 10 4 1 2 3 5 6 7 8 9 10 4 3 2 1 5 6 7 8 9 10	Remove first element and add last element and move last element(1) to first element,After that we must reorginaze our array for max heap priority, -Compare 1 with its max child(4),1 smaller so exchange them. -Compare 1 wit its max child(3),1 smaller so exchange them.
1 3 2 4 5 6 7 8 9 10 3 1 2 4 5 6 7 8 9 10	Remove first element and add last element and move last element(1) to first element,After that we must reorginaze our array for max heap priority, -Compare 1 with its max child(3),1 smaller so exchange them
2 1 3 4 5 6 7 8 9 10	Remove first element and add last element and move last element(2) to first element,After that we must

	reorginaze our array for max heap priority, Compare 2 with its max child(1),2 is bigger so dont change them.
1 2 3 4 5 6 7 8 9 10	Add last element too then its done.
1 2 3 4 5 6 7 8 9 10	

Total Comparision=23
Total Displacement=21

C) C = {5, 2, 13, 9, 1, 7, 6, 8, 1, 15, 4, 11}

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

First build heap:

Heap	Action
5 2 13 9 1 7 6 8 1 15 4 11	Compare 2 and 5, 2 smaller so dont exchange
13 2 5 9 1 7 6 8 1 15 4 11	Compare 13 with its parent(5),13 is greater so exchange them
13 2 5 9 1 7 6 8 1 15 4 11 13 9 5 2 1 7 6 8 1 15 4 11	Compare 9 with its parent(2), 9 is greater so exchange them,then compare 9 with 13,9 smaller so dont exchange
13 9 5 2 1 7 6 8 1 15 4 11	Compare 1 with its parent(9),1 smaller so dont exchange
13 9 5 2 1 7 6 8 1 15 4 11 13 9 7 2 1 5 6 8 1 15 4 11	Compare 7 with its parent(5),7 greater so exchange them,thencompare 7 with 13, 7 smaller so dont exhange
13 9 7 2 1 5 6 8 1 15 4 11	Compare 6 with its parent(7), 6 smaller so dont exchange
13 9 7 2 1 5 6 8 1 15 4 11 13 9 7 8 1 5 6 2 1 15 4 11	Compare 8 with its parent(2),8 greater so exchange them,then compare 8 with 9, 8 smaller so dont exchange
13 9 7 8 1 5 6 2 1 15 4 11	Compare 1 with its parent(8),1 smaller then 8 so dont exchange

13 9 7 8 1 5 6 2 1 15 4 11 13 9 7 8 15 5 6 2 1 1 4 11 13 9 15 8 7 5 6 2 1 1 4 11 15 9 13 8 7 5 6 2 1 1 4 11	Compare 15 with its parent(1),15 greater so exchange them,compare 15 with 7,15 greater so exchange them,compare 15 with 13,15 greater so exchange
15 9 13 8 7 5 6 2 1 1 4 11	Compare 4 with 7,4 smaller so dont exchange
15 9 13 8 7 5 6 2 1 1 4 11 15 9 13 8 7 11 6 2 1 1 4 5	Compare 11 with 5,11 greater so exchange them,compare 11 with 13,11 smaller so dont excahnge

We build heap

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

Now we must shrink this and result will be sorted array

Heap	Action
5 9 13 8 7 11 6 2 1 1 4 15 13 9 5 8 7 11 6 2 1 1 4 15 13 9 11 8 7 5 6 2 1 1 4 15	Compare 5 with its max child(13), 5 smaller so exchange them,compare 5 with its max child(11) exchange them
4 9 11 8 7 5 6 2 1 1 13 15 11 9 4 8 7 5 6 2 1 1 13 15 11 9 6 8 7 5 4 2 1 1 13 15	Compare 4 with its max child(11),4 smaller so exchange them,Compare 4 with its max child(6),4 smaller so exchange them.
1 9 6 8 7 5 4 2 1 11 13 15 9 1 6 8 7 5 4 2 1 11 13 15	Compare 1 with its max child(9),1 smaller so exchange them,Compare 1 with its max child(8),1 smaller so exchange them.Compare 1 with its max child(2),1 smaller so exchange them

9 8 6 1 7 5 4 2 1 11 13 15 9 8 6 2 7 5 4 1 1 11 13 15	
1 8 6 2 7 5 4 1 9 11 13 15 8 1 6 2 7 5 4 1 9 11 13 15 8 7 6 2 1 5 4 1 9 11 13 15	Compare 1 with its max child(8),1 smaller exchange them,Compare 1 with its max child(7),1 smaller so exchange them
1 7 6 2 1 5 4 8 9 11 13 15 7 1 6 2 1 5 4 8 9 11 13 15 7 2 6 1 1 5 4 8 9 11 13 15	Compare 1 with its max child(7),1 smaller so exchange them,compare 1 with its max child(2),1 smaller so exchange them.
4 2 6 1 1 5 7 8 9 11 13 15 6 2 4 1 1 5 7 8 9 11 13 15 6 2 5 1 1 4 7 8 9 11 13 15	Compare 4 with its max child(6),4 smaller so exchange them,compare 4 with its max child(5),4 smaller so exchange them
4 2 5 1 1 6 7 8 9 11 13 15 5 2 4 1 1 6 7 8 9 11 13 15	Compare 4 with its max child(5),4 smaller so exchange them
1 2 4 1 5 6 7 8 9 11 13 15 4 2 1 1 5 6 7 8 9 11 13 15	Compare 1 with its max child(4),1 smaller so exchange them
1 2 1 4 5 6 7 8 9 11 13 15	Compare 1 with its max child(2),1 smaller then 2 so exchange them.

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

After this our array sorted

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

Total Comparision=33
Total Displacement=34

d-) D = {'S', 'B', 'I', 'M', 'H', 'Q', 'C', 'L', 'R', 'E', 'P', 'K'}

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

Heap	Action
S B I M H Q C L R E P K	Compare B with its parent(S),B smaller so dont exchange
S B I M H Q C L R E P K	Compare I with its parent(S),I smaller than S so dont change
S B I M H Q C L R E P K S M I B H Q C L R E P K	Compare M with its parent(B),M bigger than B so exchange them,Compare M with its parent(S), M smaller than S so dont exchange
S M I B H Q C L R E P K	Compare H with its parent(M),H smaller then M so dont exchange
S M I B H Q C L R E P K S M Q B H I C L R E P K	Compare Q with its parent(I),Q bigger than I so exchange them,Compare Q with its parent(S),Q smaller than S so dont exchange
S M Q B H I C L R E P K	Compare C with its parent(Q),C smaller than Q so dont exchange

S M Q B H I C L R E P K S M Q L H I C B R E P K	Compare L with its parent(B),L bigger than B so exchange them.Compare L with its parent(M), L smaller than M so dont exchange
S M Q L H I C B R E P K S M Q R H I C B L E P K S R Q M H I C B L E P K	Compare R with its parent(L), R bigger so exchange them,Compare R with its parent(M),R bigger so exchange them,Compare R with its parent(S),R smaller so dont exchange
S R Q M H I C B L E P K	Compare E with its parent(H),E smaller then H so dont exchange them.
S R Q M H I C B L E P K S R Q M P I C B L E H K	Compare P with its parent(H),P bigger so exchange them,Compare P with its parent(R),P smaller so dont exchange
S R Q M P I C B L E H K S R Q M P K C B L E H I	Compare K with its parent(I),K bigger so exchange them,compare K with its parent(Q), K smaller so dont exchange.

This is our heap now we must shrink it

Heap	Action
I R Q M P K C B L E H S R I Q M P K C B L E H S R P Q M I K C B L E H S	Compare I with its max child(R),I smaller so exchange them,Compare I with its max child(P),I smaller so exchange them, Compare I with its max child(H), I bigger so dont exchange.
H P Q M I K C B L E R S	Compare H with its max child(Q),H smaller so exchange them,Compare H with its max child(K),H smaller than K so exchange them

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

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

Compare E with its max child(P),E smaller so exchange them,Compare E with its max child(M),E smaller so exchange them,comapre E with its max child(L),E smaller so exchange them.

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

Compare E with its max child(M),E smaller so exchange them,compare E with its maz child(L),E smaller so exchange them,comapre E with B,E bigger so dont exchange them.

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

Compare B with its maz child(L),B smaller so exchange them, Compare B with its max child(I),B smaller so exchange them

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

Compare C with its max child(K),C smaller so exchange them,Compare C with its max child(H),C smaller so exchange them.

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

Compare C with its max child(I),C smaller so exchance them,Compare C with its max child(E),C smaller so exchange.

I E H C B K L M P Q R S	
B E H C I K L M P Q R S H E B C I K L M P Q R S	Compare B with its max child(H),B smaller so exchange them
C E B H I K L M P Q R S E C B H I K L M P Q R S	Compare C with its max child(E),C smaller so exchange them.
B C E H I K L M P Q R S C B E H I K L M P Q R S	Compare B with its max child(C),B smaller so exchange
B C E H I K L M P Q R S	

The result array is B C E H I K L M P Q R S

Total Comparisions=38
Total Displacements=35

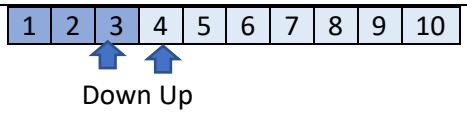
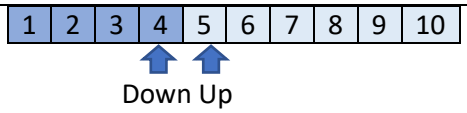
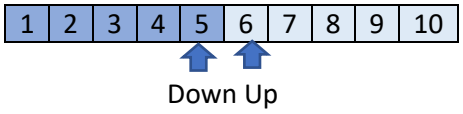
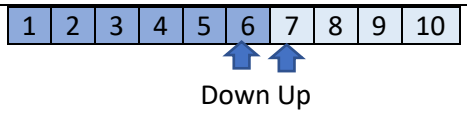
Quick Sort

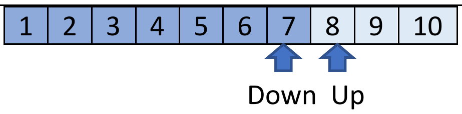
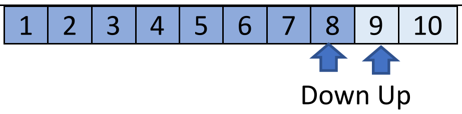
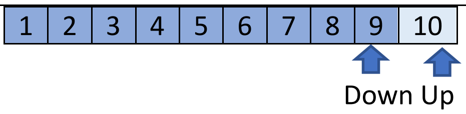
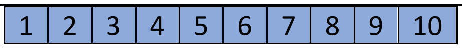
a)A is an ordered integer array with 10 elements from small to large

So I choose A array like this to Show Quick Sort steps;

1	2	3	4	5	6	7	8	9	10

Array	Actions
1 2 3 4 5 6 7 8 9 10 Down Up	Select the first element as Pivot element. Move the smaller element to the left side of the pivot and move the greater elements to the rigt side of the pivot.Then start to search first element that greater than pivot element on the left side(Up) and start to search first element that smaller than pivot element on the right side(Down).Then exchange Down and Up elements and repeat, If down past up so stop and exchange pivot with down element.
1 2 3 4 5 6 7 8 9 10 Down Up	Select the first element as Pivot element. Move the smaller element to the left side of the pivot and move the greater elements to the rigt side of the pivot.Then start to search first element that greater than pivot element on the left side(Up) and start to search first element that smaller than pivot element

	on the right side(Down).Then exchange Down and Up elements and repeat, If down past up so stop and exchange pivot with down element.
	Select the first element as Pivot element. Move the smaller element to the left side of the pivot and move the greater elements to the rigt side of the pivot.Then start to search first element that greater than pivot element on the left side(Up) and start to search first element that smaller than pivot element on the right side(Down).Then exchange Down and Up elements and repeat, If down past up so stop and exchange pivot with down element.
	Select the first element as Pivot element. Move the smaller element to the left side of the pivot and move the greater elements to the rigt side of the pivot.Then start to search first element that greater than pivot element on the left side(Up) and start to search first element that smaller than pivot element on the right side(Down).Then exchange Down and Up elements and repeat, If down past up so stop and exchange pivot with down element.
	Select the first element as Pivot element. Move the smaller element to the left side of the pivot and move the greater elements to the rigt side of the pivot.Then start to search first element that greater than pivot element on the left side(Up) and start to search first element that smaller than pivot element on the right side(Down).Then exchange Down and Up elements and repeat, If down past up so stop and exchange pivot with down element.
	Select the first element as Pivot element. Move the smaller element to the left side of the pivot and move the greater elements to the rigt side of the pivot.Then start to search first element that greater than pivot element on the left side(Up) and start to search first element that smaller than pivot element on the right side(Down).Then exchange

	Down and Up elements and repeat, If down past up so stop and exchange pivot with down element.
	Select the first element as Pivot element. Move the smaller element to the left side of the pivot and move the greater elements to the rigt side of the pivot.Then start to search first element that greater than pivot element on the left side(Up) and start to search first element that smaller than pivot element on the right side(Down).Then exchange Down and Up elements and repeat, If down past up so stop and exchange pivot with down element.
	Select the first element as Pivot element. Move the smaller element to the left side of the pivot and move the greater elements to the rigt side of the pivot.Then start to search first element that greater than pivot element on the left side(Up) and start to search first element that smaller than pivot element on the right side(Down).Then exchange Down and Up elements and repeat, If down past up so stop and exchange pivot with down element.
	Select the first element as Pivot element. Move the smaller element to the left side of the pivot and move the greater elements to the rigt side of the pivot.Then start to search first element that greater than pivot element on the left side(Up) and start to search first element that smaller than pivot element on the right side(Down).Then exchange Down and Up elements and repeat, If down past up so stop and exchange pivot with down element.
	Select the first element as Pivot element. Move the smaller element to the left side of the pivot and move the greater elements to the rigt side of the pivot.Then start to search first element that greater than pivot element on the left side(Up) and start to search first element that smaller than pivot element on the right side(Down).Then exchange Down and Up elements and repeat, If

	down past up so stop and exchange pivot with down element.

In this array there is no left pivot element because we choose first element in this array as pivot so there is no small element then this pivot element.So always pivot element will be right place no need to change its position.

Total Comparisions=72
Total Displacements=9

b) B is an ordered integer array with 10 elements from large to small

So I choose B array like this to Show Quick Sort steps ;

10	9	8	7	6	5	4	3	2	1

Array	Actions
10 9 8 7 6 5 4 3 2 1 Down 1 9 8 7 6 5 4 3 2 10	Select first element Pivot(10), Start Up from left and start Down from right,find greater then pivot with up and find smaller than down.then exchange them.If Down index pass Up index then exchange down index with pivot element Here Down will be 1 but Up compare all element with pivot element. But all element smaller than up so up goes to the end of the array and pass up,so we exchange down with pivot and call same steps with pivot left and right sides.
1 9 8 7 6 5 4 3 2 10 1 9 8 7 6 5 4 3 2 10	Select first element as pivot element(1),There is no smaller then pivot element in array so Down index pass Up index. Call quick sort fort he right side of the array
1 9 8 7 6 5 4 3 2 10 Down 1 2 8 7 6 5 4 3 9 10	Select first element pivot,There is no Up index because there is no bigger element then 9 so its pass Down index,so just exchange pivot with Down index. Call quick sort for left side of array
1 2 8 7 6 5 4 3 9 10	Select first element(2) as pivot.Up will be 8 but there is no down because smaller element is pivot so Down index become 2, that means pivot element in correct possition dont need to exchange. Call quick sort for right side of the array
1 2 8 7 6 5 4 3 9 10 Down	Select first element(8) as pivot. There is no Up index because there is no bigger element then 8 so its pass Down

1 2 3 7 6 5 4 8 9 10	index,so just exchange pivot with Down index. Call quick sort for left side of the array
1 2 3 7 6 5 4 8 9 10 Up	Select first element as pivot(3). Up will be 7 but there is no down because smaller element is pivot so Down index become 3, that means pivot element in correct possition dont need to exchange. Call quick sort for right side of the array
1 2 3 7 6 5 4 8 9 10 Down 1 2 3 4 6 5 7 8 9 10	Select first element as pivot(7). There is no Up index because there is no bigger element then 7 so its pass Down index,so just exchange pivot with Down index. Call quick sort for left side of the array
1 2 3 4 6 5 7 8 9 10 Up	Select first element as pivot(4), Up will be 6 but there is no down because smaller element is pivot so Down index become 4, that means pivot element in correct possition dont need to exchange. Call quick sort for right side of the array
1 2 3 4 6 5 7 8 9 10 Down 1 2 3 4 5 6 7 8 9 10	Select first element as pivot(6), There is no Up index because there is no bigger element then 6 so its pass Down index,so just exchange pivot with Down index. Call quick sort for left side of the array
1 2 3 4 5 6 7 8 9 10	There is only one element left and this is pivot so its done

1	2	3	4	5	6	7	8	9	10

It is finished result is the sorted array.
Total Comparisions=67
Total Displacements=9

c-) C = {5, 2, 13, 9, 1, 7, 6, 8, 1, 15, 4, 11}

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

Array	Actions
5 2 13 9 1 7 6 8 1 15 4 11 1 2 4 1 5 7 6 8 9 15 13 11 Down Up	Select first element as pivot(5) Up 13 and Down 4, Exchange them, Up 9, down 1, exchage them, Up 7,down 1 but down pas sup so exchange with pivot and down. Call quick sort for left and right array

1 2 4 1 5 7 6 8 9 15 13 11 1 2 4 1 5 7 6 8 9 15 13 11	Select first element as pivot(1). There is no down its in correct position
1 2 4 1 5 7 6 8 9 15 13 11 Up Down 1 1 2 4 5 7 6 8 9 15 13 11 1 1 2 4 5 7 6 8 9 15 13 11	For left array choose first element as pivot(2), Up 4 and down 1 so exchange them, Down 1 but its pass Up so exchange Down with pivot And there is only one element(4), so its already sorted
1 1 2 4 5 7 6 8 9 15 13 11 1 1 2 4 5 6 7 8 9 15 13 11	Now lets back table 1 right array,Firstly choose first element as pivot element(7), Up 8 and down 6 but down pass up so exchange down with pivot element.Pivot in correct place Call Quick sort for left side and right side but, left side only 1 element so its already in correct position so call for right side.
1 1 2 4 5 6 7 8 9 15 13 11 1 1 2 4 5 6 7 8 9 15 13 11 1 1 2 4 5 6 7 8 9 15 13 11	Select first elemenet as pivot(8), Up 9 but down is pivot element so dont exchange anything pivot element in right place Casll quick sort for right side of array
1 1 2 4 5 6 7 8 9 15 13 11 1 1 2 4 5 6 7 8 9 15 13 11	Select first element as pivot(9), Up 15 but down is pivot element so dont exchange anything pivot in correct place Call quick side for ride side of array
1 1 2 4 5 6 7 8 9 15 13 11 1 1 2 4 5 6 7 8 9 11 13 15	Select first element as Pivot element(15), Down element is 11 but up pass Down so exchange pivot with down element, Call Quick sort for right side of the array.
1 1 2 4 5 6 7 8 9 11 13 15 1 1 2 4 5 6 7 8 9 11 13 15	Select first element as pivot(11), Up 13 but down is pivot so dont exchange pivot in corretc place, After that there is only one element left so its already sorted
1 1 2 4 5 6 7 8 9 11 13 15	

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

Total Comparision=59
Total Displacements=11

d-) D = {‘S’, ‘B’, ‘I’, ‘M’, ‘H’, ‘Q’, ‘C’, ‘L’, ‘R’, ‘E’, ‘P’, ‘K’}

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

Array	Actions
S B I M H Q C L R E P K K B I M H Q C L R E P S	Select first element as pivot(S), Down is K but there is no Up because there is no bigger element than S in array,So exchage K with S. Call Quick Sort for left array
K B I M H Q C L R E P S Up Down K B I E H Q C L R M P S Up Down K B I E H C Q L R M P S C B I E H K Q L R M P S	Select first element as pivot(K), Up M down E, so exchange them,Then Up is Q and Down is C, so exchange them,Then Up is Q and Down is C,So Down pass Up,Exchange Down and pivot Call Quick sort for left and right array
C B I E H K Q L R M P S Down Up B C I E H K Q L R M P S	Select first elemen as pivot(C), Up is I and down is B ,Down pass Up so exchange pivot element with Down Call Quick sort for left and right array Of element C, But in left side there is only one olement so that means its already sorted.Then call quick sort for right hand
B C I E H K Q L R M P S Down B C H E I K Q L R M P S	Select first element as pivot(I),Down is H but there is no Up element so exchange pivot with Down, Call Quick sort for left side
B C H E I K Q L R M P S B C E H I K Q L R M P S B C E H I K Q L R M P S	Select first element as pivot(H), Down is E and there is no Up so exchange E with pivot After that there is only one element in left side E, so it means its sorted already
B C E H I K Q L R M P S Up Down B C E H I K Q L P M R S Down Up	Get back right side of the Secont table array. Select First element as pivot(Q),Up R and Down P, Exchange Up and Down, Then Up R and Down M, but Down pass Up so exchange M with pivot.

B C E H I K M L P Q R S	Now call quick sort for right side and left side but right side only has one element so its dont need call right side quick sort its already sorted
B C E H I K M L P Q R S Down Up B C E H I K L M P Q R S	Select first element as Pivot(M),Up is P and Down is L, Because of Down pass Up,Exchange Pivot and Down. Call Quick sort for left and right array. But left and right array has 1 element so they are already sorted
B C E H I K L M P Q R S	

Sorted Array:

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

Total Comparisions=58
Total Displacements=10

Muhammed Yasir Fidan
161044056