ASSIGNMENT - 05

NAME: M. Prasad

REGr NO: 192311252

COURSE CODE: CSA 0389

COURSE NAME: DATA STRUCTURE

DATE:- 21-AUG-2024

the algorithm for insertion sort and sort the
wing sequence.

3, 1, 4, 1, 5 9, 2, 6, 5

) Explain the procedure for merge sort and perform the merge sort for following inputs. Also show the result for each step of iteration 64, 9, 216, 512, 27, 729, 0, 1, 343, 125

d- Algorithm for insertion:-

1) Begin with second element in list
2) compare the current element to previous element
3) shift all larger elements one position to right
4) Insert the current elements into its correct position.
5. Repeat steps 2-4 for each element in list until the entire list is sorted.

sorting sequence:-

sequence : 3, 1, 4, 1, 5, 9, 2, 6, 5

3	1	4	1	5	9	2	6	5

compare 3 & 1, 3>1
swap 3, 1

1	3	4	1	5	9	2	6	5

compare 4 & 1, 4>1
swap 4, 1

1	3	1	4	5	9	2	6	5

compare 3 & 1, 3>1
swap 3, 1

1	1	3	4	5	9	2	6	5

compare 9 & 2, 9>2
swap 9, 2

1	1	3	4	5	2	9	6	5

compare 5 & 2
swap 5, 2

1	1	3	4	2	5	9	6	5

compare 4 & 2, 4 > 2
swap 4, 2

1	1	3	2	4	5	9	6	5

compare 3 & 2, 3 > 2
swap 3, 2

1	1	2	3	4	5	9	6	5

compare 9 & 6, 9 > 6
swap 9, 6

1	1	2	3	4	5	6	9	5

compare 9 & 5, 9 > 5
swap 9, 5

1	1	2	3	4	5	5	6	9

sorted

Sorted sequence: 1, ~~8~~ 1, 2, 3, 4, 5, 5, 6, 9

Merge sort procedure:

-> split the list into halves until each sublist has one element

-> combine the sublists to produce new sorted sublists until there is one sorted list.

Merge sort with 64, 8, 216, 512, 27, 729, 0, 1, 343, 125

1) Initial split:
- [64, 8, 216, 512, 27] and [729, 0, 1, 343, 125]

2 Further split:
- [64, 8] and [216, 512, 27]
  [729, 0] and [1, 343, 125]

the concept map of partitioning in quick sort, to write an algorithm for it, which is as llows & develop a program considering the steps
step 1:- choose the highest index value has pivot
step 2:- take two variables to point left and right Of the list excluding pivot
step 3:- left points to low index using elements your own.

Algorithm:-
* select the element at highest index as pivot
* set 'left' to low index and 'right' to high index-1
* Move 'left' rightwards and 'right' leftwards until 'left' is greater than or equal to 'right' swapping elements as the needed.
* swap the pivot with the element at the 'left' pointer position.
* Return the index of the pivot element.

Program:-
```
#include<stdio.h>
int main() {
    int arr[] = {64, 8, 216, 512, 27, 729, 0, 1, 343, 125};
    int n = size of (arr) size of (arr[0]);
```

3. Further split:

- [64] and [8]
- [216] and [512, 27]
- [729] and [0]
- [1] and [343, 125]

4. Merge:

- Merge [64] and 8 -> [8, 64]
- Merge [512, 27] -> [27, 512]
- Merge [216] and [27, 512] -> [27, 216, 512]
- Merge [0] and [729] -> [0, 729]
- Merge [125, 343] -> [125, 343]
- Merge [1] and [125, 343] -> [1, 125, 343]

5. Final Merge:-

- Merge [8, 64] and [27, 216, 512]
  -> [8, 27, 64, 216, 512]
- Merge [0, 729] and [1, 125, 343]
  -> [0, 1, 125, 343, 729]
- Merge [8, 27, 64, 216, 512] and [0, 1, 125, 343, 729]

-> [0, 1, 8, 27, 64, 125, 216, 343, 512, 729]

sorted list: 0, 1, 8, 27, 64, 125, 216, 343, 512, 729

```
int low =0 , high =n-1;
le ( low < high) {
    int pivot = arr [high];
     int left = low;
    int right = high-1;
while (left <= right) {
    while (left <= right && arr [left] < pivot) {
        left ++;
    }
 while (right >= low && arr [right] > pivot) {
        right --;
    }
 if (left < right) {
       int temp = arr [left];
        arr [left] = arr [right];
        arr [right] = temp;
          left ++;
 right --;
    }
 }
   int temp = arr [left];
      arr [left] = arr [high];
       arr [high] = temp;
       high = left - 1;
if (high < low) {
        low = left+1;
        high = n-1;
    }
 }
printf("sorted array:")
for (int i=0; i<n; i++){
        printf("%d", arr[i]);
   }
      printf("\n");
    return 0;
   }
```

OUTPUT:-

```
sorted array: 0, 1, 8, 27, 64, 125, 216, 343, 512, 729.
```