ASSIGNMENT- 5

Name: Bhuvaneshwari. S

Reg no: 192311375

Course Code: CSA0339

Course name: Data structure for stack overflow

Faculty name: Dr. Ashok Kumar

Date of Submission: 21/08/24

i) Write the algorithm for Insertion sort and sort the following
sequence: 3,1,4,1,5,9,2,6,5

Algorithm:

1. Start with the second element (index i) of the array
   → The first element is considered sorted.
2. For each element from the second to the last.
   Key: The element of key in the sorted portion.
   Compare: The key with elements in the sorted portion from left to right.
   shift: Elements of the sorted portion to the right if they are greater than the key
   Insert: The key into its correct position until the entire array is sorted.
3. Repeat the above step for each element until the entire array is sorted

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

(i) | 3 | 1 | 4 | 1 | 5 | 9 | 2 | 6 | 5 | → First two elements are compared (3 & 1)
                                              3 > 1, swap 1, 3

(ii) | 1 | 3 | 4 | 1 | 5 | 9 | 2 | 6 | 5 | → Compare next two elements
                                              4 > 1, swap 1, 4   (4,1)

(iii)| 1 | 3 | 1 | 4 | 5 | 9 | 2 | 6 | 5 | ⇒ compare next two elements
                                              3 > 1, swap 3, 1   (3,1)

(iv) | 1 | 1 | 3 | 4 | 5 | 9 | 2 | 6 | 5 | → 6 > 2, Swap 9,2

(v) | 1 | 1 | 3 | 4 | 5 | 2 | 9 | 6 | 5 | → 5 > 2 , swap

(vi) | 1 | 1 | 3 | 4 | 2 | 5 | 9 | 6 | 5 | → 4 > 2 , swap

(vii) | 1 | 1 | 3 | 2 | 4 | 5 | 9 | 6 | 5 | → 3 > 2, swap

(viii) | 1 | 1 | 2 | 3 | 4 | 5 | 9 | 6 | 5 | → compare the least element and shift it's place towards left (6,5)

(ix) | 1 | 1 | 2 | 3 | 4 | 5 | 9 | 5 | 6 | → compare next element and (9,5) 9 > 5 , swap

(x) | 1 | 1 | 2 | 3 | 4 | 5 | 5 | 9 | 6 | → compare the next element (9,6) 9 > 6 swap

| 1 | 1 | 2 | 3 | 4 | 5 | 5 | 6 | 9 | ⇒ sorted Array

(ii) Explain the procedure for merge sort and perform the merge sort for the following input. Also, show the input for each step of iteration.

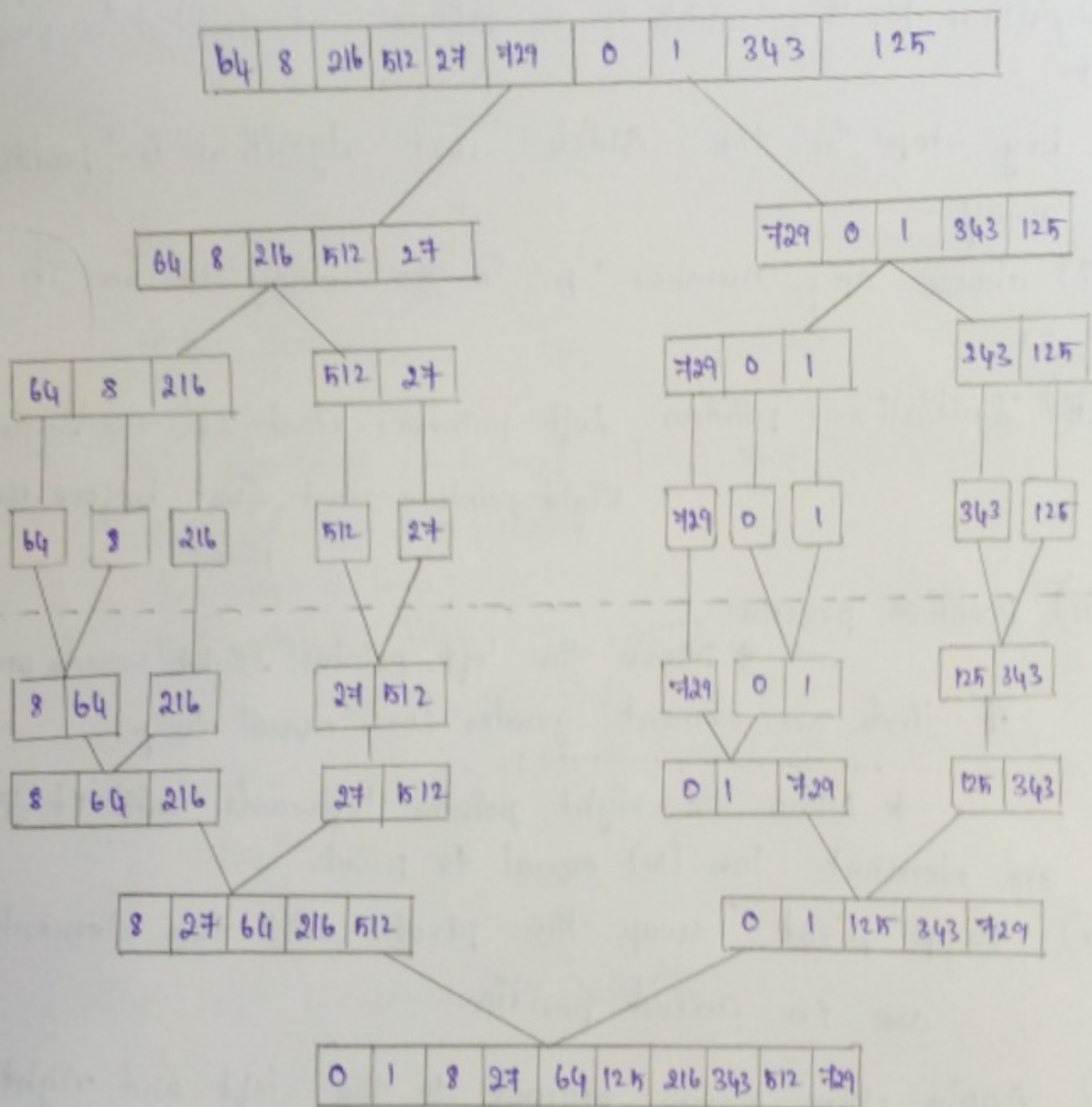64, 8, 216, 512, 27, 729, 0, 1, 343, 125

Algorithm:

Divide:
* If the array has more than one element, split into two halves.
* Continue recursively splitting each half until you have subarrays that are trivially sorted

Conquer:
* Recursively sort each other smaller subarrays. Since arrays with one element are already sorted, this focuses on the merging process.

64	8	216	512	27	729	0	1	343	125

64	8	216	512	27

729	0	1	343	125

64	8	216

512	27

729	0	1

343	125

64		8		216

512		27

729		0		1

343		125

8	64	216

27	512

729	0	1

125	343

8	64	216

27	512

0	1	729

125	343

8	27	64	216	512

0	1	125	343	729

0	1	8	27	64	125	216	343	512	729

SORTED ARRAY

Draw the concept map of partitioning in quick sort, try to write an algorithm for it, which is as follows and develop a program for it.

A key step in the Quick sort algorithm is 'partitioning' the array.

(i) choose any number 'p' in the array to use its as 'pivot'

(ii) Initialize pointers: Left pointer: start at the begining

Right pointer: start just before the pivot

(iii) Partition process:
* Move the left pointer right wards until it finds an element greater (or) equal to p.

* Move the right pointer leftwards until it finds an element less (or) equal to pivot.

(iv) Swap pivot: swap the pivot until the elements are in correct position.

(v) Apply the same process to the left and right sub-arrays.

Algorithm:
1. choose element at index 'high' as pivot
2. Initialize − left = low
   − right = high − 1
3. while left <= right.
   a. Increment left while A[left] < pivot
   b. decrement right while A[right] > pivot
   c. If left <= right;
      swap A[left] and A[right]
4. swap pivot
5. Return left

5

Program:

```c
# include < stdio. h>
void swap (int* a , int * b) {
    int temp = *a ;
        *a = *b;
        *b = temp;
    }
    int partition (int arr[] , int low, int high) {
        int pivot = arr[high];
        int left = low;
        int right = high-1;
        while  (left <= right) {
            while (left <= right && arr[left]< pivot) {
                left ++;
            }
while (left <= right && arr[right] > pivot){
    right --;
    }
    if (left <= right) {
    swap (& arr[left], & arr[right]);
    left ++;
    right --;
    }
    }
    swap (& arr[left] , & arr[high]);
    return left;
    }
void quick_sort (int arr[]. int low, int high) {
    if (low < high) {
        int pivot_index = partition (arr, low, high);
```

6

```c
        quick - sort (arr, low, pivot index -1);
        quick - sort (arr, pivot - index +1, high);
    }
}
void print - array (int arr[ ], int size){
    for (int i=0; i < size; i++){
        printf ("%d ", arr[i]);
    }
    printf (" \n");
}

int main ( )
{
    int arr[ ] = {10, 7, 8, 9, 1, 5};
    int size = sizeof (arr)/ sizeof (arr[0]);
    printf ("original array: ")};
    print - array (arr, size);
    quick - sort (arr, 0, size-1);
    printf ("sorted array"),
    print - array (arr, size);
    return 0;
}
```