College : Vishwakarma Institute of Technology

Name : Vedika Vikas Sontakke

Roll no : 37

PNR No : 1222006

Course : Data Structure

Assignment No 1 Problem Statement 7

- ## Quick Sort

- **Aim :**

  WAP to implement Quick Sort on 1D array of Employee structure (contains employee_name, emp_no, emp_salary), with key as emp_no. And count the number of swap performed.

- **Program :**

```c
/*
WAP to implement Quick Sort on 1D array of Employee
structure (contains employee_name, emp_no, emp_salary), with key as
emp_no. And count the number of swap performed.
*/

#include<stdio.h>
#include <string.h>
int count_swap = 0;
struct Employee
{
    char employee_name[40];
    int emp_no;
    int emp_salary;
```

```
};

void swap(struct Employee *emp_array , int x  , int y)
{
    struct Employee temp;
    temp = emp_array[x];
    emp_array[x] = emp_array[y];
    emp_array[y] = temp;

    count_swap++;
}

void quick_sort(struct Employee emp_array[] , int first , int last)
{
    int i , j , pivot;

    if(first < last){

    i = first;
    j = last;
    pivot = first;

    while(i < j)
    {
       while(emp_array[i].emp_no <= emp_array[pivot].emp_no && i<last)
         i++;

       while(emp_array[j].emp_no > emp_array[pivot].emp_no)
         j--;

       if(i < j) swap(emp_array , i , j);
    }

    swap(emp_array , j , pivot);

    // for left digits
    quick_sort(emp_array , 0 , pivot-1);
      // for right digits;
    quick_sort(emp_array , pivot+1 , last);
    }

}
int main()
{
    int size ;
```

```
    printf("enter the size of the array \n");
    scanf("%d",&size);

    struct Employee emp_array[size];

    printf("enter the employee no , employee name , employee salary :");
    for(int i=0 ; i<size ; i++)
       scanf("%d %s %d" , &emp_array[i].emp_no , &emp_array[i].employee_name ,
&emp_array[i].emp_salary);

    quick_sort(emp_array , 0 , size-1);

    printf("number of swaps are : %d" , count_swap);

    printf("\n print the employee no , employee name , employee salary  after
sorting :");
    for(int i=0 ; i<size ; i++)
       printf("\n %d %s %d ",emp_array[i].emp_no,emp_array[i].employee_name ,
emp_array[i].emp_salary);

    return 0;

}
```

Output :

```
PS C:\Users\Lenovo\Documents\vit\ds\Assignment no 1 .. problem 7> cd "c:\Users\Lenovo\Documents\vit\ds\Assignment no 1 .. pr
oblem 7\" ; if ($?) { gcc quick_sort.c -o quick_sort } ; if ($?) { .\quick_sort }
enter the size of the array
5
enter the employee no , employee name , employee salary :
2 e 4000
7 t 8000
1 w 2000
5 k 3000
3 n 1000
number of swaps are : 8
 print the employee no , employee name , employee salary  after sorting :
 1 w 2000
 2 e 4000
 3 n 1000
 5 k 3000
 7 t 8000
PS C:\Users\Lenovo\Documents\vit\ds\Assignment no 1 .. problem 7>
```

- **Merge Sort**

- **Aim :**

WAP to implement Merge Sort on 1D array of Employee structure (contains employee_name, emp_no, emp_salary), with key as emp_no. And count the number of swap performed.

- **Program :**

```
/*
WAP to implement Merge Sort on 1D array of Employee
structure (contains employee_name, emp_no, emp_salary), with key as
emp_no. And count the number of swap performed.
*/

#include<stdio.h>

void merge_sort();
void merge();
int count_swap = 0;
struct Employee
{
    char employee_name[40];
    int emp_no;
    int emp_salary;
};

int main()
{
     int size;
    printf("enter the size of the array \n");
    scanf("%d",&size);

    struct Employee emp_array[size];

    printf("enter the employee no , employee name , employee salary :");
    for(int i=0 ; i<size ; i++)
       scanf("%d %s %d" , &emp_array[i].emp_no ,
&emp_array[i].employee_name , &emp_array[i].emp_salary);

    merge_sort(emp_array , 0 , size-1);
```

```
    printf("count number of swaps %d", count_swap);
    printf("\nprint the employee no , employee name , employee
salary  after sorting :");
    for(int i=0 ; i<size ; i++)
       printf("\n %d %s %d
",emp_array[i].emp_no,emp_array[i].employee_name ,
emp_array[i].emp_salary);

    return 0;
}

void merge_sort(struct Employee emp_array[], int first , int last)
{
    int mid ;
    if(first < last)
    {
        mid = (first+last)/2;
        merge_sort(emp_array , first , mid);
        merge_sort(emp_array , mid+1 , last);
        merge(emp_array , first , mid , last);
    }
}

void merge(struct Employee emp_array[], int first , int mid , int last)
{
    struct Employee b[50];
    int i , j , k;
    i = first ;
    j = mid+1;
    k = first;

    while(i<=mid && j<=last)
    {
        if(emp_array[i].emp_no <= emp_array[j].emp_no)
        {
            b[k++] = emp_array[i++];
            count_swap++;
        }
        else {
           b[k++] = emp_array[j++];
           count_swap++;
        }
    }

    if(i>mid)
```

```
    {
        while(j<=last){
           b[k++]= emp_array[j++];
           count_swap++;
        }

    }
    else{
        while(i<=mid){
           b[k++] = emp_array[i++];
           count_swap++;
        }
    }

    for( i=first ; i<=last ; i++)
       emp_array[i] = b[i];
}
```

- **Output :**

```
PS C:\Users\Lenovo\Documents\vit\ds\Assignment no 1 .. problem 7> cd "c:\Users\Lenovo\Documents\vit\ds\Assignment no 1 .. pr
oblem 7\" ; if ($?) { gcc merge_sort.c -o merge_sort } ; if ($?) { .\merge_sort }
enter the size of the array
5
enter the employee no , employee name , employee salary :
5 rr 3000
4 tt 8000
2 ee 6000
1 kk 5000
3 ss 4000
count number of swaps 12
print the employee no , employee name , employee salary  after sorting :
 1 kk 5000
 2 ee 6000
 3 ss 4000
 4 tt 8000
 5 rr 3000
```

- **Heap Sort**

- **Aim :**

WAP to implement Heap Sort on 1D array of Employee

structure (contains employee_name, emp_no, emp_salary), with key as emp_no. And count the number of swap performed

- **Program :**

```
/*
WAP to implement Heap Sort on 1D array of Employee
structure (contains employee_name, emp_no, emp_salary), with key as
emp_no. And count the number of swap performed.
*/
#include<stdio.h>
void heap_Sort();
void heapify();
int count_swap = 0;
struct Employee
{
    char employee_name[40];
    int emp_no;
    int emp_salary;
};

int main()
{
    int size;
    printf("enter the size of the array \n");
    scanf("%d",&size);

    struct Employee emp_array[size];

    printf("enter the employee no , employee name , employee salary :");
    for(int i=0 ; i<size ; i++)
       scanf("%d %s %d" , &emp_array[i].emp_no , &emp_array[i].employee_name ,
&emp_array[i].emp_salary);

    heap_Sort(emp_array , size);
     printf("count number of swaps are %d", count_swap);
    printf("\nprint the employee no , employee name , employee salary  after
sorting :");
    for(int i=0 ; i<size ; i++)
       printf("\n %d %s %d ",emp_array[i].emp_no,emp_array[i].employee_name ,
emp_array[i].emp_salary);
```

```
    return 0;

}


void swap(struct Employee *emp_array , int x  , int y)
{
    struct Employee temp;
    temp = emp_array[x];
    emp_array[x] = emp_array[y];
    emp_array[y] = temp;

    count_swap++;
}

void heapify(struct Employee emp_array[], int size, int i)
{
    //  largest among root, left child and right child
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;
    // If left child is larger than root
    if (left < size && emp_array[left].emp_no > emp_array[largest].emp_no)
        largest = left;

    // If right child is larger than largest
    if (right < size && emp_array[right].emp_no > emp_array[largest].emp_no)
        largest = right;

    // Swap and continue heapifying if root is not largest
    if (largest != i) {
        swap(emp_array , i , largest);
        // Recursively heapify the affected sub-tree
        heapify(emp_array, size, largest);
    }
}

 void heap_Sort(struct Employee emp_array[], int size)
{
    // Build max heap
    for (int i = size / 2 - 1; i >= 0; i--)
        heapify(emp_array, size, i);

    // Heap sort
    for (int i = size - 1; i >= 0; i--) {
```

```
        swap(emp_array , 0 , i);
        // Heapify root element to get highest element at root again
        heapify(emp_array, i, 0);
    }

}
```

- **Output :**

```
PS C:\Users\Lenovo\Documents\vit\ds\Assignment no 1 .. problem 7> cd "c:\Users\Lenovo\Documents\vit\ds\Assignment no 1 .. problem 7\"
 ; if ($?) { gcc heap_sort.c -o heap_sort } ; if ($?) { .\heap_sort }
enter the size of the array
5
enter the employee no , employee name , employee salary :
10 dd 3000
50 ss 4000
40 vv 5000
30 rr 2000
20 kk 1000
count number of swaps are 9
print the employee no , employee name , employee salary  after sorting :
 10 dd 3000
 20 kk 1000
 30 rr 2000
 40 vv 5000
 50 ss 4000
PS C:\Users\Lenovo\Documents\vit\ds\Assignment no 1 .. problem 7>
```

-THANK YOU