Data Structures and Algorithms –CSE2011

FALL 2021-2021 – Slot – L57+L58

Lab Exercises

Date: 24-09-2021

1. Implementation of selection sort algorithm for sorting N numbers in ascending and descending order.
2. Implementation of Bubble sort algorithm for sorting N numbers in ascending and descending order.
3. Implementation of insertion sort algorithm for sorting N numbers in ascending and descending order
4. Implementation of Merge sort algorithm for sorting N numbers in ascending and descending order
5. Implementation of Quick sort algorithm for sorting N numbers in ascending and descending order

array_boiler.h

(common to all)

```c
#include<stdio.h>
#include<stdlib.h>
int *arr,n;

void take_input()
{
    printf("\nEnter the size of the array: ");
    scanf("%d",&n);
    arr=(int*)malloc(sizeof(int)*n);
    printf("Enter the array: ");
    for(int i=0;i<n;i++)
    scanf("%d",arr+i);
}

void disp()
{
    printf("\nArray:");
    for(int i=0;i<n;i++)
    printf(" %d",arr[i]);
}

void swap(int p1,int p2)
{
    int t=arr[p1];
    arr[p1]=arr[p2];
    arr[p2]=t;
}
```

Q1)

Aim: To develop C programming for sorting N elements using Selection sort algorithms.

Algorithm:

1. The function selection_sort takes an integer as input and sorts the array arr[] from the index 0 to p by using recursive call.
2. The function selection_sort first checks if the value of p is less than 0. then the function returns.
3. Then it initiates the recursive call for p-1
4. Then it finds the p+1th greatest/smallest element of the array with the help of for loop.
5. Then it swaps the pth index index with the element containing p+1th greatest/smallest element depending upon we are doing ascending/descending sorting.
6. And when all the recursive call is over we have a sorted arry

Code Implemented In C:

```
#include "array_boiler.h"

void selection_sort(int p)
{
   if(p<0)
   return;
   selection_sort(p-1);
   int m=p;
   for(int j=p+1;j<n;j++)
   if(arr[m]>arr[j])//for desending order we will write arr[m]<arr[j]
   m=j;
   swap(m,p);
}

main()
{
   take_input();
   selection_sort(n-1);
   disp();
}
```

Output Obtained:

```
Enter the size of the array: 10
Enter the array: 4 5 3 103 3 124 -4 3 1 10

Array: -4 1 3 3 3 4 5 10 103 124
```

Results: The code has been developed and tested successfully for selection sort algorithm.

Q2)

Aim: To develop C programming for sorting N elements using Selection Bubble sort algorithms.

Algorithm:

1. The function bubble_sort takes 2 integer as input and sorts the array arr[] from the index 0 to p by using recursive call.
2. The function bubble_sort first checks if the value of j is less than 0, then decrements i and assigns then ith index from the last to j.
3. Then it initiates the recursive call for p-1
4. If the element at jth index is grater/less than the element at j+th index it swaps those elements
5. At each recursive call of i we find the greatest/smallest element of the array and puts it to its position ,depending upon we are doing ascending/descending sorting.
6. And when all the recursive call is over we have a sorted array

Code Implemented In C:

```
#include "array_boiler.h"

void bubble_sort(int i,int j)
{
   if(j<0)
   {
      i--;
      j=n-i-1;
   }
   if(i<0)
   return;
   bubble_sort(i,j-1);
   if (arr[j] > arr[j+1])//for desending order we will write arr[j]<arr[j+1]
         swap(j,j+1);
}

main()
{
   take_input();
   bubble_sort(n-1,n-1);
   disp();
}
```

Output Obtained:

```
Enter the size of the array: 10
Enter the array: 45 43 -23 -4 0 243 -342 4 56 3

Array: -342 -23 -4 0 3 4 43 45 56 243
```

Results: The code has been developed and tested successfully for Bubble sort algorithm.

Q3)

Aim: To develop C programming for sorting N elements using insertion sort algorithms.

Algorithm:

1. The function insertion_sort takes an integer as input and sorts the array arr[] from the index 0 to i by using recursive call.
2. The function insertion_sort first checks if the value of i is 0 then it returns.
3. With each recursive call we sort the array till ith index
4. By using the while loop we find position of element at ith index in our sorted array.
5. In the while loop we keep on shifting all the elements to the right until the right position is found.
6. Then we place the element at that index

Code Implemented In C:

```
#include "array_boiler.h"


void insertion_sort(int i)
{
   if(i==0)
   return;
   int p, j;
   insertion_sort(i-1);
   p = arr[i];
   j = i - 1;

   while (j >= 0 && arr[j] > p)
   {
      arr[j+1] = arr[j];
      j=j-1;
   }
      arr[j + 1] = p;
}

main()
{
   take_input();
   insertion_sort(n-1);
   disp();
}
```

Output Obtained:

```
Enter the size of the array: 10
Enter the array: 45 43 -23 -4 0 243 -342 4 56 3

Array: -342 -23 -4 0 3 4 43 45 56 243
```

Results: The code has been developed and tested successfully for insertion sort algorithm.

Q4)

Aim: To develop C programming for sorting N elements using merge sort algorithms.

Algorithm:

1. We follow divide and conqour technique
2. First we divide the array into two parts and then merge those divided arrays.
3. The mergeSort function calls itself recursively.
4. Fist it sorts the first half of the array and then the second half of the array
5. After that we combine both the array using merge function
6. In the merge function we make the copy of both parts of the array into another array
7. Then we check each subsequent present index in the array to place in the main array.
8. At last if the elements are not placed from the first array then we place it in the array.
9. Otherwise if the loop has been terminated due to the first array then we place the remaining part of the second array.

Code Implemented In C:

```
#include "array_boiler.h"

void merge(int l, int m, int r)
{
  int l1=m-l+1,l2=r-m;
  int left[l1],right[l2];
  for(int i=0;i<l1;i++)
  left[i]=arr[i+l];
  for(int i=0;i<l2;i++)
  right[i]=arr[i+m+1];
  int a1=0,a2=0,f=l;

  while(a1<=m-l && a2<r-m)
  if(left[a1]<right[a2])
  arr[f++]=left[a1++];
  else
  arr[f++]=right[a2++];
  while(a1<l1)
  arr[f++]=left[a1++];
  while(a2<l2)
  arr[f++]=right[a2++];
}
void mergeSort(int l, int r)
{
```

```
    if (l <r)
    {
        int m = l + (r - l) / 2;
        mergeSort(l, m);
        mergeSort(m + 1, r);
        merge(l, m, r);
    }
}
main()
{
    take_input();
    mergeSort(0,n-1);
    disp();
}
```

Output Obtained:

```
Enter the size of the array: 10
Enter the array: 45 43 -23 -4 0 243 -342 4 56 3

Array: -342 -23 -4 0 3 4 43 45 56 243
```

Results: The code has been developed and tested successfully for merge sort algorithm.

Q5)

Aim: To develop C programming for sorting N elements using Quick Bubble sort algorithms.

Algorithm:

1. The function quick_sort takes 2 integer as input and sorts the array arr[] from the index l to n by using recursive call.
2. The function quick_sort first checks if the value of n is less than equal to l, if so it returns.
3. Then it initiates j=l.
4. It takes the n^{th} element as pivot element and shifts all the elements greater/smaller than the number at the beginning of the array using an iterative loop.
5. Then it shifts the pivot element at the position of j so that all the elements greater than it should be after it.
6. Then it calls the quick_sort function to sort the elements after and before it.

Code Implemented In C:

```
#include "array_boiler.h"



void quick_sort(int l,int n)
{
   if(n<=l)
   return;

   int j=l;

   for(int i=l;i<n;i++)
   if(arr[i]<arr[n])
   swap(i,j++);

   swap(j,n);


   quick_sort(l,j-1);
   quick_sort(j+1,n);
}

main()
{
   take_input();
   quick_sort(0,n-1);
   disp();
}
```

Output Obtained:

```
Enter the size of the array: 7
Enter the array: 90 80 70 50 40 30 10

Array: 10 30 40 50 70 80 90
```

```
Enter the size of the array: 5
Enter the array: 787 686 4 6 7

Array: 4 6 7 686 787
```

Results: The code has been developed and tested successfully for quick sort algorithm.

CLICK HERE FOR GITHUB LINK