

Institute of Engineering & Management
Department of Computer Science & Engineering
Design & Analysis of Algorithm Lab for 3rd year 5th semester 2018
Code: CS 591

Date: 01/08/18

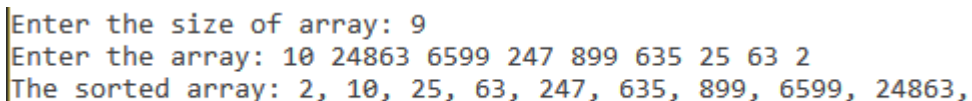
WEEK-2

Source code:

```
#include <stdio.h>

void radixsort(int arr[], int n){
    int max=arr[0], i, j, temp[n];
    for(i=1;i<n;i++)
        if(max<arr[i])
            max = arr[i];
    for(i=1;max/i>0;i*=10){
        int count[10]={0};
        for(j=0;j<n;j++)
            count[(arr[j]/i)%10]++;
        for(j=1;j<10;j++)
            count[j]+=count[j-1];
        for(j=n-1;j>=0;j--){
            temp[count[(arr[j]/i)%10]-1]=arr[j];
            count[(arr[j]/i)%10]--;
        }
        for(j=0;j<n;j++)
            arr[j]=temp[j];
    }
}

int main(){
    int n, i;
    printf("Enter the size of array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter the array: ");
    for(i=0;i<n;i++)
        scanf("%d",&arr[i]);
    radixsort(arr, n);
    printf("The sorted array: ");
    for(i=0;i<n;i++)
        printf("%d, ",arr[i]);
    printf("\n");
    return 0;
}
```

Screen-Shot:

```
Enter the size of array: 9
Enter the array: 10 24863 6599 247 899 635 25 63 2
The sorted array: 2, 10, 25, 63, 247, 635, 899, 6599, 24863,
```

Time Complexity:

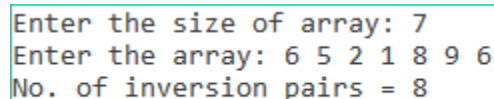
Source code:

```
#include <stdio.h>
#include <stdlib.h>

int inver_count(int *arr, int low, int high){
    int mid=(low+high)/2, i, lp=low, rp=(low+high)/2, count_inver=0,
        temp[high-low];

    if(low>=high-1)
        return 0;
    else{
        count_inver+=inver_count(arr, low, mid);
        count_inver+=inver_count(arr, mid, high);
        for(i=0;i<high-low;i++){
            if(lp==mid)
                temp[i]=arr[rp++];
            else if(rp==high)
                temp[i]=arr[lp++];
            else if(arr[lp]>arr[rp]){
                temp[i]=arr[rp++];
                count_inver+=mid-lp;
            }
            else if(arr[lp]<=arr[rp])
                temp[i]=arr[lp++];
        }
        for(i=0;i<high-low;i++)
            arr[i+low]=temp[i];
        return count_inver;
    }
}

int main(){
    int n, i, *arr;
    printf("Enter the size of array: ");
    scanf("%d", &n);
    arr = (int *)malloc(n*sizeof(int));
    printf("Enter the array: ");
    for(i=0;i<n;i++)
        scanf("%d", &arr[i]);
    printf("No. of inversion pairs = %d\n", inver_count(arr, 0, n));
    free(arr);
    return 0;
}
```

Screen-Shot:A screenshot of a terminal window showing the execution of the program. The user enters '7' for the size of the array, then '6 5 2 1 8 9 6' for the array elements. The program outputs 'No. of inversion pairs = 8'.

```
Enter the size of array: 7
Enter the array: 6 5 2 1 8 9 6
No. of inversion pairs = 8
```

Time Complexity:

Source code:

```
#include <stdio.h>

int kth(int *arr1, int *arr2, int *end1, int *end2, int k){
    if (arr1 == end1)
        return arr2[k];
    if (arr2 == end2)
        return arr1[k];
    int mid1 = (end1 - arr1) / 2;
    int mid2 = (end2 - arr2) / 2;
    if (mid1 + mid2 < k){
        if (arr1[mid1] > arr2[mid2])
            return kth(arr1, arr2 + mid2 + 1, end1, end2, k-mid2-1);
        else return kth(arr1 + mid1 + 1, arr2, end1, end2, k-mid1-1);
    }
    else{
        if (arr1[mid1] > arr2[mid2])
            return kth(arr1, arr2, arr1 + mid1, end2, k);
        else return kth(arr1, arr2, end1, arr2 + mid2, k);
    }
}

int main(){
    int n, m, k, i;
    printf("Enter the size of 1st & 2nd array: ");
    scanf("%d%d", &n, &m);
    int arr1[n], arr2[m];
    printf("Enter the 1st array: ");
    for(i=0;i<n;i++)
        scanf("%d",&arr1[i]);
    printf("Enter the 2nd array: ");
    for(i=0;i<m;i++)
        scanf("%d",&arr2[i]);
    printf("Enter the postion: ");
    scanf("%d", &k);
    printf("Element at %d position is %d\n", kth(arr1, arr2, arr1+n,
arr2+m, k-1));
    return 0;
}
```

Screen-Shot:

```
Enter the size of 1st & 2nd array: 5 5
Enter the 1st array: 1 2 4 7 9
Enter the 2nd array: 3 5 6 7 9
Enter the postion: 8
Element at 8 position is 7
```

Time Complexity:

Source code:

```
#include <stdio.h>

int kth(int n, int *arr, int pos)
{
    int left = -1, right = n, pivot = arr[0], i=0, temp, res=0;
    if(n<=1)
        return arr[0];
    while(left!=right-1)
    {
        if(i%2 == 0)
        {
            if(pivot>=arr[left+1])
                left++;
            else{
                temp = arr[left+1];
                arr[left+1] = arr[right-1];
                arr[right-1] = temp;
                right--;
            }
        }
        else{
            if(pivot<=arr[right-1])
                right--;
            else{
                temp = arr[left+1];
                arr[left+1] = arr[right-1];
                arr[right-1] = temp;
                left++;
            }
        }
        i++;
    }
    if(left!=-1)
    {
        arr[0] = arr[left];
        arr[left] = pivot;
    }
    if(left>pos)
        res=kth(left, arr, pos);
    else if(left<pos)
        res=kth(n-right, &arr[right], pos-left-1);
    else return arr[left];
    return res;
}

int main()
{
    int n, i;
    printf("Enter the size(>1) of array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter the array: ");
    for(i=0;i<n;i++)
        scanf("%d",&arr[i]);
    printf("The neighbor elements are %d & %d\n", kth(n, arr, ((n/2)-1)),
        kth(n, arr, ((n+1)/2)));
    return 0;
}
```

Screen-Shot:

```
Enter the size(>1) of array: 10  
Enter the array: 9 0 3 1 4 5 2 6 7 8  
The neighbor elements are 4 & 5
```

Time Complexity: