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

Date: 25/07/18

WEEK-1

Name: Ranajit Roy, **Sec:** A, **Roll:** 47

Source code:

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

void sort(int *arr, int low, int high)
{
    int mid=(low+high)/2;
    if(low>=high-1)
        return;
  else{
        sort(arr, low, mid);
        sort(arr, mid, high);
        merge(arr, low, high);
  }
}

void merge(int *arr, int low, int high)
{
    int i, lp=low, rp=(low+high)/2, temp[high-low];
    for(i=0;i<high-low;i++)
  {
        if(lp==(low+high)/2)
              temp[i]=arr[rp++];
        else if(rp==high)
              temp[i]=arr[lp++];
        else if(arr[lp]>arr[rp])
              temp[i]=arr[rp++];
        else if(arr[lp]<=arr[rp])
              temp[i]=arr[lp++];
  }
  for(i=0;i<high-low;i++)
        arr[i+low]=temp[i];
}

int main()
{
  int *arr, i, n;
  printf("Enter the no. of elements: ");
  scanf("%d",&n);
  arr = (int *)malloc(n*sizeof(int));
  printf("Enter the elements: ");
  for(i=0;i<n;i++)
        scanf("%d", &arr[i]);
  sort(arr,0,n);
  printf("The sorted array is: ");
  for(i=0;i<n;i++)
        printf("%d, ", arr[i]);
  printf("\n");
  return 0;
}
```

Screen-shot:

```
rana@rana:~/Desktop/C/Week 1$ gcc merge_sort.c -w
rana@rana:~/Desktop/C/Week 1$ ./a.out
Enter the no. of elements: 7
Enter the elements: 6 5 2 3 4 1 7
The sorted array is: 1, 2, 3, 4, 5, 6, 7,
rana@rana:~/Desktop/C/Week 1$
```

Source code:

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

void qsrt(int n, int *arr)
{
  int left = -1, right = n, pivot = arr[0], i=0, temp;
  if(n<=1)
        return;
  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;
  }
  qsrt(left, arr);
  qsrt(n-right, &arr[right]);
}

int main()
{
  int *arr, i, n;
  printf("Enter the no. of elements: ");
  scanf("%d",&n);
  arr = (int *)malloc(n*sizeof(int));
  printf("Enter the elements: ");
  for(i=0;i<n;i++)
        scanf("%d", &arr[i]);
  qsrt(n, arr);
  printf("The sorted array is: ");
```

```
    for(i=0;i<n;i++)
          printf("%d, ", arr[i]);
    printf("\n");
    return 0;
}
```

Screen-shot:

```
rana@rana:~/Desktop/C/Week 1$ gcc qst.c
rana@rana:~/Desktop/C/Week 1$ ./a.out
Enter the no. of elements: 6
Enter the elements: 5 2 3 4 1 6
The sorted array is: 1, 2, 3, 4, 5, 6,
rana@rana:~/Desktop/C/Week 1$
```

Time complexity:

Source code:

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

void sort(int *arr, int low, int high)
{
  int mid=(low+high)/2, lp, rp, stack[4], top = -1;
  if(low == high-2)
  {lp = mid-1; rp = high-1;}
  else if(low == high-3)
  {lp = mid-1; rp = high-2;}
  else {lp = mid-2; rp = high-2;}
  if(low==high-1)
        return;
  else{
        sort(arr, low, mid);
        sort(arr, mid, high);
        while(lp != mid || rp!=high)
        {
              if(lp == mid)
                    stack[++top] = arr[rp++];
              else if(rp == high)
                    stack[++top] = arr[lp++];
              else if(arr[lp] < arr[rp])
                    stack[++top] = arr[lp++];
              else stack[++top] = arr[rp++];
        }
        while(top+1)
        {
              arr[(((high-low)>4)?(high-4):low) +top] = stack[top];
              top--;
        }
  }
}

int main()
{
  int *arr, i, n;
  printf("Enter the no. of elements(>1): ");
  scanf("%d",&n);
  arr = (int *)malloc(n*sizeof(int));
  printf("Enter the elements: ");
  for(i=0;i<n;i++)
        scanf("%d", &arr[i]);
  sort(arr,0,n);
  printf("The 2nd highest is: %d\n", arr[n-2]);
  printf("\n");
  return 0;
}
```

Screen-shot:

```
rana@rana:~/Desktop/C/Week 1$ gcc sechigh.c
rana@rana:~/Desktop/C/Week 1$ ./a.out
Enter the no. of elements(>1): 6
Enter the elements: 5 3 7 9 1 5
The 2nd highest is: 7

rana@rana:~/Desktop/C/Week 1$ |
```

Source code:

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

void find_sum(int n, int *arr, int x)
{
  int right=n-1, left=0;
  while(left<right)
  {
        if(arr[left]+arr[right] == x)
        {
              printf("%d, %d\n", arr[left], arr[right]);
              left++; right--;
        }
        else if(arr[left]+arr[right] > x)
              right--;
        else left++;
  }
}

int main()
{
  int n, i, x, *arr, *result=NULL;
  printf("Enter the no. of elements: ");
  scanf("%d", &n);
  arr = (int *)malloc(n*sizeof(int));
  printf("Enter the elements: ");
  for(i=0;i<n;i++)
  {
        scanf("%d", &arr[i]);
  }
  printf("Enter the sum value: ");
  scanf("%d", &x);
  find_sum(n,arr,x);
  return 0;
}
```

Screen-shot:

```
rana@rana:~/Desktop/C/Week 2$ gcc a2.c
rana@rana:~/Desktop/C/Week 2$ ./a.out
Enter the no. of elements: 8
Enter the elements: 2 3 5 6 7 9 10 13
Enter the sum value: 15
2, 13
5, 10
6, 9
rana@rana:~/Desktop/C/Week 2$
```

Source code:

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

int find_row(int n, int **arr, int x)
{
  int i;
  for(i=0;i<n;i++)
  {
        if(arr[i][0]>x)
              break;
  }
  return i-1;
}
void find(int n, int m, int **arr, int x)
{
  int res, rs = find_row(n, arr, x);
  if(rs == -1)
        printf("Not Found\n");
  else
        res = search(0, m, arr[rs], x);
        if(res == -1)
              printf("Not Found\n");
        else printf("Found in arr[%d][%d]\n", rs, res);
}
int search(int low, int high, int *arr, int x)
{
  int mid = (low+high)/2, res=-1;
  if(low==high-1)
        if(arr[low] == x)
              return low;
        else res;
  else if(arr[mid] == x)
        return mid;
  else if(arr[mid] > x)
        res = search(low, mid, arr, x);
  else if(arr[mid] < x)
        res = search(mid, high, arr, x);
  return res;
}

int main()
{
  int n, m, i, j, x, **arr, *result=NULL;
  printf("Enter N: ");
  scanf("%d", &n);
  printf("Enter M: ");
  scanf("%d", &m);
  arr = (int **)malloc(n*sizeof(int *));
  for(i=0;i<n;i++)
        arr[i] = (int *)malloc(m*sizeof(int));
  printf("Enter the matrix:\n");
  for(i=0;i<n;i++)
        for(j=0;j<m;j++)
              scanf("%d", &arr[i][j]);
  printf("Enter the value: ");
  scanf("%d", &x);
  find(n, m,arr,x);
  return 0;
}
```

Screen-shot:

```
rana@rana:~/Desktop/C/Week 2$ gcc a3.c -w
rana@rana:~/Desktop/C/Week 2$ ./a.out
Enter N: 3
Enter M: 3
Enter the matrix:
2 3 5
6 7 8
10 11 23
Enter the value: 10
Found in arr[2][0]
rana@rana:~/Desktop/C/Week 2$ |
```

Time complexity:

Source code:

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

int search1(int low, int high, int *arr, int x)
{
  int mid, res=-1;
  while(low<=high-3)
  {
        mid = (low+high)/2;
        if(arr[mid]>=x)
              high = mid+1;
        else
              low = mid+1;
  }
  if(low==high-2)
  {
        if(arr[low]==x)
              return low;
        else if(arr[low+1]==x)
              return low+1;
        else return res;
  }
  else if(low==high-1)
        if(arr[low]==x)
              return low;
        else return res;
  return res;
}

int search2(int low, int high, int *arr, int x)
{
  int mid, res=-1;
  while(low<=high-3)
  {
        mid = (low+high)/2;
        if(arr[mid]<=x)
              low = mid;
        else
              high = mid;
  }
  if(low==high-2)
  {
        if(arr[low+1]==x)
              return low+1;
        else if(arr[low]==x)
              return low;
        else return res;
  }
  else if(low==high-1)
        if(arr[low]==x)
              return low;
        else return res;
  return res;
}

int main()
{
  int n, pos1, pos2, i, x, *arr;
  printf("Enter N: ");
  scanf("%d", &n);
```

```
    arr = (int *)malloc(n*sizeof(int));
    printf("Enter the array: ");
    for(i=0;i<n;i++)
          scanf("%d", &arr[i]);
    printf("Enter the value: ");
    scanf("%d", &x);
    pos1 = search1(0, n, arr, x);
    pos2 = search2(0, n, arr, x);
    if(pos1==-1 || pos2==-1)
          printf("Error\n");
    else
          printf("No. of occurences = %d\n", (pos2-pos1)+1);
    return 0;
}
```

Screen-shot:

```
rana@rana:~/Desktop/C/Week 2$ gcc a4.c
rana@rana:~/Desktop/C/Week 2$ ./a.out
Enter N: 7
Enter the array: 2 6 6 7 7 7 9
Enter the value: 7
No. of occurences = 3
rana@rana:~/Desktop/C/Week 2$ |
```

Time complexity:

Source code:

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

float median(int n, int *arr1, int *arr2)
{
  int L1=0,H1=n, L2=0,H2=n;
  float med1, med2, max, min;
  while(L1<H1-2)
  {
        med1 = (((H1-L1)%2==1)? arr1[(L1+H1)/2] : (float)(arr1[(L1+H1-
                                                1)/2]+arr1[(L1+H1)/2])/2);
        med2 = (((H2-L2)%2==1)? arr2[(L2+H2)/2] : (float)(arr2[(L2+H2-
                                                1)/2]+arr2[(L2+H2)/2])/2);
        if(med1==med2)
              return med1;
        else if(med1<med2){
              L1 = (L1+H1-1)/2;
              H2 = ((L2+H2)/2)+1;
        }
        else{
              L2 = (L2+H2-1)/2;
              H1 = ((L1+H1)/2)+1;
        }
  }
  if(arr1[L1]>arr2[L2])
        max = (float)arr1[L1];
  else max = (float)arr2[L2];
  if(arr1[L1+1]>arr2[L2+1])
        min = (float)arr2[L2+1];
  else min = (float)arr1[L1+1];
  return (max+min)/2;
}

int main()
{
  int n, i, *arr1, *arr2;
  printf("Enter N(>1): ");
  scanf("%d", &n);
  arr1 = (int *)malloc(n*sizeof(int));
  arr2 = (int *)malloc(n*sizeof(int));
  printf("Enter the 1st sorted array: ");
  for(i=0;i<n;i++)
        scanf("%d", &arr1[i]);
  printf("Enter the 2nd sorted array: ");
  for(i=0;i<n;i++)
        scanf("%d", &arr2[i]);
  printf("The median is = %0.2f\n", median(n, arr1, arr2));
  return 0;
}
```

Screen-shot:

```
rana@rana:~/Desktop/C/Week 2$ gcc a5.c
rana@rana:~/Desktop/C/Week 2$ ./a.out
Enter N(>1): 7
Enter the 1st sorted array: 1 3 6 14 17 18 23
Enter the 2nd sorted array: 12 13 25 29 30 37 39
The median is = 17.50
rana@rana:~/Desktop/C/Week 2$ |
```

Source code:

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

int min_arr(int n, int *arr)
{
  int mid, low=0, high=n;
  while(low<high-2)
  {
        mid = (low+high)/2;
        if(arr[mid]<arr[mid-1])
              return arr[mid];
        else if(arr[mid]<arr[0])
              high = mid+1;
        else low = mid;
  }
  if(low==0 || high==n)
        return arr[0];
  if(arr[low]>arr[low+1])
        return arr[low+1];
  else return arr[low];
}

int main()
{
  int n, i, *arr;
  printf("Enter N(>1): ");
  scanf("%d", &n);
  arr = (int *)malloc(n*sizeof(int));
  printf("Enter the array(distinct element): ");
  for(i=0;i<n;i++)
        scanf("%d", &arr[i]);
  printf("The minimum element is = %d\n", min_arr(n, arr));
  return 0;
}
```

Screen-shot:

```
rana@rana:~/Desktop/C/Week 2$ gcc a6.c
rana@rana:~/Desktop/C/Week 2$ ./a.out
Enter N(>1): 9
Enter the array(distinct element): 6 7 8 9 1 2 3 4 5
The minimum element is = 1
rana@rana:~/Desktop/C/Week 2$ |
```

Time complexity:

Source code:

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

int min_arr(int n, int *arr)
{
  int mid, low=0, high=n, dif=(arr[n-1]-arr[0])/n, req;
  while(low<high-2)
  {
        mid = (low+high)/2;
        req = arr[0]+(mid*dif);
        if(arr[mid]>req || arr[mid]<req)
              high = mid+1;
        else if(arr[mid]==req)
              low = mid;
  }
  return arr[0]+(low+1)*dif;
}

int main()
{
  int n, i, *arr;
  printf("Enter N(>1): ");
  scanf("%d", &n);
  arr = (int *)malloc(n*sizeof(int));
  printf("Enter the array: ");
  for(i=0;i<n;i++)
        scanf("%d", &arr[i]);
  printf("The minimum element is = %d\n", min_arr(n, arr));
  return 0;
}
```

Screen-shot:

```
rana@rana:~/Desktop/C/Week 2$ gcc a7.c
rana@rana:~/Desktop/C/Week 2$ ./a.out
Enter N(>1): 6
Enter the array: 1 4 10 13 16 19
The minimum element is = 7
rana@rana:~/Desktop/C/Week 2$ |
```

Time complexity:

Source code:

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

int min_arr(int n, int *arr)
{
  int mid, low=0, high=n;
  while(low<high-2)
  {
        mid = (low+high)/2;
        if(arr[mid]<arr[mid-1])
              high = mid+1;
        else if(arr[mid]>arr[mid-1])
              low = mid;
  }
  if(arr[low]>arr[low+1])
        return low+1;
  else return low+2;
}

int main()
{
  int n, i, *arr;
  printf("Enter N(>1): ");
  scanf("%d", &n);
  arr = (int *)malloc(n*sizeof(int));
  printf("Enter the Bitonic sequence: ");
  for(i=0;i<n;i++)
        scanf("%d", &arr[i]);
  printf("The Bitonic point is = %d\n", min_arr(n, arr));
  return 0;
}
```

Screen-shot:

```
rana@rana:~/Desktop/C/Week 2$ gcc a8.c
rana@rana:~/Desktop/C/Week 2$ ./a.out
Enter N(>1): 8
Enter the Bitonic sequence: 2 3 4 5 3 1 -1 -3
The Bitonic point is = 4
rana@rana:~/Desktop/C/Week 2$ |
```

Time complexity: