**PROBING PEAK IN ARRAYS**

**LAB # 05**



**Data Structures & Algorithms**

Submitted by: **Shah Raza**

Registration No: **18PWCSE1658**

Class Section: **B**

“On my honor, as a student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to: **Dr. Khurram Shehzad Khattak**

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Lab Objectives:**

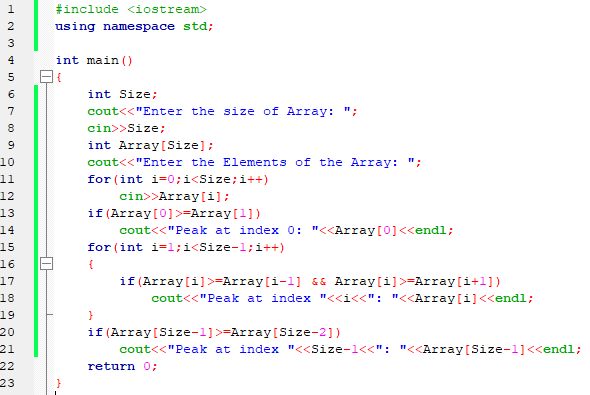
Objectives of this lab are to learn about some techniques and algorithms to probe peak in:

* One Dimensional Array
* Two Dimensional Array

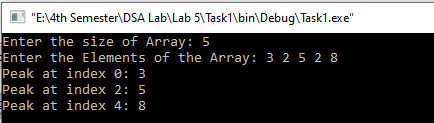
**Task # 1:**

Probe peak element in one dimensional array and analyze its worst, best and average case complexity.

**Code:**

****

**Output:**

****

**Complexity:**

* **Best case:**

For linear peak finding in 1D array the best case complexity is O[1] since in this algorithm if the peak is at the first or last index then we can find it easily.

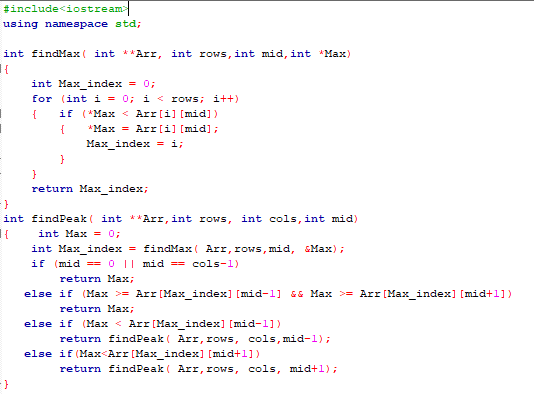
* **Worst case:**

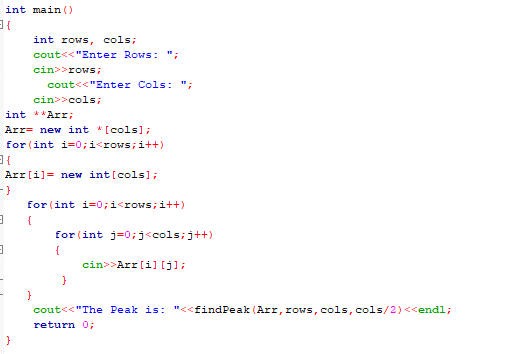
For linear peak finding in 1D array the worst case complexity is O[n ] since in this algorithm if the peak is not at first or last index then we traverse through the whole Array.

**Task # 2:**

Probe peak element in two dimensional array and analyze its worst, best and average case complexity.

**Code:**

****

****

**Pseudo-Code/Explanation:**

* Ask the user to enter rows and columns of the Array
* Ask the user to enter elements of the Array
* Call the function to find Peak

Peak Finding function (Array,Rows, Columns, Midpoint)

{

Put max=0

Call another function to find the maximum index.

Put maxindex = Find maximum index function(Array,Rows,Midpoint,max)

if mid is equal to 0 or mid is equal to cols-1

then return max

else if max is greater than or equal to array[maxindex][mid-1] and max is greater than or equal to array[maxindex][mid+1])

then return max

else if max is less than array[maxindex][mid-1])

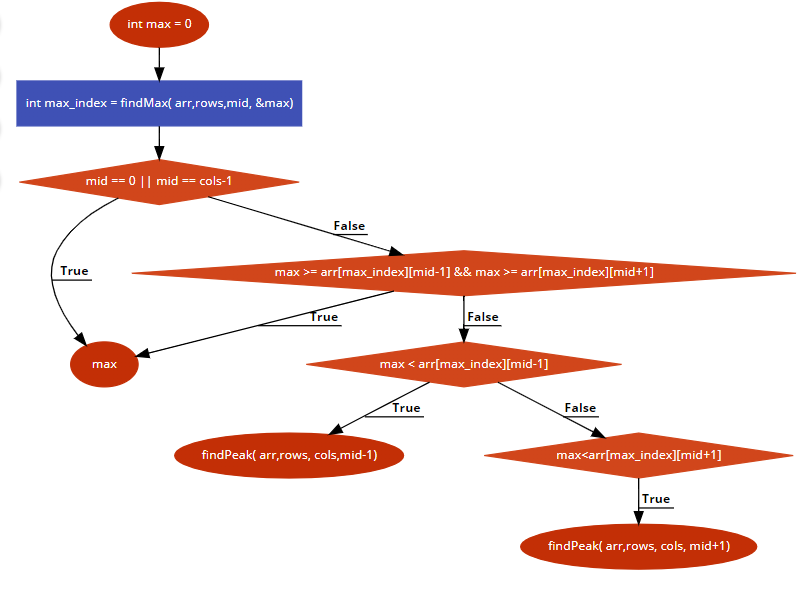
then recursively call findPeak(array,rows, cols,mid-1)

else if max is less than arr[maxindex][mid+1])

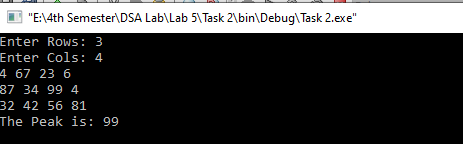
then recursively call findPeak( array,rows, cols, mid+1)

}

**Flow Chart:**

****

**Output:**

****

**Complexity:**

* **Best case/Worst case:**

For binary peak finding in 2D array the best case and worst case complexity is O[n log n] since in this algorithm the function will check if the peak is at the right or left side of the center and call itself recursively to reach the peak.