



Day-4

Name:Rajan Mishar

UID:22BCS103621

Branch:BE-CSE

Sec:615-IOT

1.Recursive Insertion Sort

You are tasked with completing the `insertsort()` function to implement the Insertion Sort algorithm. The function should take an array as input and return the array sorted in ascending order.

Examples

Example 1:

● Input:

`arr[] = [4, 1, 3, 9, 7]` ●

Output:

`[1, 3, 4, 7, 9]`

● Explanation:

The sorted array is `[1, 3, 4, 7, 9]`.

```
#include <iostream>
```

```
#include <vector> using
```

```
namespace std;
```

```
void insertSortRecursive(vector<int>& arr, int n) {
```

```
    if (n <= 1) {        return;
```

```
    }
```



```
insertSortRecursive(arr, n - 1);

int last = arr[n - 1];

int j = n - 2;

while (j >= 0 && arr[j] > last) {
arr[j + 1] = arr[j];    j--; }
arr[j + 1] = last;
} int main() {    vector<int> arr = {4,
1, 3, 9, 7};
insertSortRecursive(arr, arr.size());
for (int num : arr) {    cout << num
<< " ";
}
cout << endl;

return 0;
}
```

2.Row with Maximum 1s

You are given a 2D binary array `arr[][]` consisting of only 1s and 0s. Each row of the array is sorted in non-decreasing order. Your task is to determine and return the index of the row that contains the maximum number of 1s. If no such row exists, return -1.

Notes:



- The array follows 0-based indexing.
- If multiple rows have the same number of 1s, return the index of the first such row.

Examples:

Input:

```
arr[][] = [[0, 1, 1, 1],  
[0, 0, 1, 1],  
[1, 1, 1, 1], [0,  
0, 0, 0]]
```

Output: 2

Explanation: Row 2 contains the maximum number of 1s (4 1s). Hence, the output is 2.

Answer

```
#include <iostream>  
  
#include <vector> using  
namespace std;  
  
int rowWithMax1s(vector<vector<int>>& arr, int n, int m) {  
    int maxRow = -1;    int maxCount = 0;    for (int i = 0; i  
< n; i++) {        int count = 0;        for (int j = 0; j < m; j++)  
{            if (arr[i][j] == 1) {                count++;  
            }  
        }  
    }
```



```
        if (count > maxCount) {  
maxCount = count;          maxRow =  
i;  
  
        }  
    }  
  
    return maxRow;  
}  
int main() {    vector<vector<int>> arr  
= {{0, 1, 1, 1},  
        {0, 0, 1, 1},  
        {1, 1, 1, 1},    {0, 0, 0, 0}};    int n = 4, m = 4;    cout << "Row with  
Maximum 1s: " << rowWithMax1s(arr, n, m) << endl;  
    return 0;  
}
```

3.Search a 2D Matrix ||

Write an efficient algorithm that searches for a value target in an m x n integer matrix matrix. This matrix has the following properties:

- Integers in each row are sorted in ascending from left to right.
- Integers in each column are sorted in ascending from top to bottom.

Example 1:



Input: matrix = [[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]], target = 5

Output: true

Answer

```
#include <iostream>

#include <vector> using
namespace std;

bool searchMatrix(vector<vector<int>>& matrix, int target) {
    int m = matrix.size();    int n = matrix[0].size();    int i =
    0, j = n - 1;    while (i < m && j >= 0) {        if
    (matrix[i][j] == target) {            return true;
        } else if (matrix[i][j] < target) {
            i++;        } else {            j--;
        }    }    return false; } int main() {
    vector<vector<int>> matrix = {{1, 4, 7, 11, 15},
                                {2, 5, 8, 12, 19},
                                {3, 6, 9, 16, 22},
                                {10, 13, 14, 17, 24},
                                {18, 21, 23, 26, 30}};    int target = 5;

    if (searchMatrix(matrix, target)) {
        cout << "Target found!" << endl;
```



```
    } else {        cout << "Target not  
found!" << endl;  
  
    }  
return 0;  
}
```

4. Find a Peak Element

A peak element in a 2D grid is an element that is strictly greater than all of its adjacent neighbors to the left, right, top, and bottom.

Given a 0-indexed $m \times n$ matrix `mat` where no two adjacent cells are equal, find any peak element `mat[i][j]` and return the length 2 array `[i,j]`.

You may assume that the entire matrix is surrounded by an outer perimeter with the value -1 in each cell.

You must write an algorithm that runs in $O(m \log(n))$ or $O(n \log(m))$ time.

Example 1:

Input: `mat = [[1,4],[3,2]]` Output:

`[0,1]`

Explanation: Both 3 and 4 are peak elements so `[1,0]` and `[0,1]` are both acceptable answers.

Answer

```
#include <iostream>
```

```
#include <vector> using
```

```
namespace std;
```



```
bool isPeakElement(const vector<vector<int>>& mat, int i, int j) {  
    int m = mat.size();    int n = mat[0].size();    if (i >  
0 && mat[i][j] <= mat[i-1][j]) return false;    if (i < m -  
1 && mat[i][j] <= mat[i+1][j]) return false;    if (j > 0  
&& mat[i][j] <= mat[i][j-1]) return false;    if (j < n - 1  
&& mat[i][j] <= mat[i][j+1]) return false;    return  
true;  
}  
  
vector<int> findPeakElement(const vector<vector<int>>& mat) {  
    int m = mat.size();    int n = mat[0].size();    for (int i = 0; i <  
m; i++) {        for (int j = 0; j < n; j++) {            if  
(isPeakElement(mat, i, j)) {                return {i, j};  
            }  
        }  
    }  
    return {-1, -1};  
}  
  
int main() {    vector<vector<int>> mat =  
{{10, 20, 15},  
    {21, 30, 14},  
    {7, 16, 32}};  
  
    vector<int> peak = findPeakElement(mat);    cout << "Peak Element found at: ["  
<< peak[0] << ", " << peak[1] << "]" << endl;    return 0;
```



}

5. Median of a Row-Wise Sorted Matrix

You are given a matrix mat of size $m \times n$ where both the number of rows (m) and columns (n) are odd. The matrix is row-wise sorted, i.e., each row is sorted in non-decreasing order. Your task is to find the median of the matrix.

Definition of Median:

The median of a matrix is the middle value when all the elements of the matrix are arranged in a sorted order. Since $m \times n$ is odd, there is always a single middle element.

Example1:

Input: $mat =$

[[1, 3, 5],
[2, 6, 9], [3,
6, 9]]

Output: 5

Explanation: Flattening the matrix and sorting gives: {1, 2, 3, 3, 5, 6, 6, 9, 9}. The median is the middle element: 5. Answer

```
#include <iostream>
```

```
#include <vector> #include
```

```
<algorithm> using
```

```
namespace std; int
```

```
findMedian(vector<vector
```




```
<int>>& mat) {  
    vector<int> flattened;  
    for (auto& row : mat) {  
        flattened.insert(flattened.en  
d(), row.begin(),  
row.end());  
    }    sort(flattened.begin(),  
flattened.end());    return  
flattened[flattened.size() / 2];  
} int main() {    vector<vector<int>>  
mat = {{1, 3, 5},  
        {2, 6, 9},  
        {3, 6, 9}};  
  
    cout << "Median: " << findMedian(mat) << endl;  
  
    return 0;  
}
```