1. **Spiral traversal of matrix**

**vector<int> spiralOrder(vector<vector<int>>& matrix) {**

**int left=0, top=0;**

**int right=matrix[0].size()-1,bottom=matrix.size()-1;**

**vector<int> result;**

**while(left<=right && top<=bottom)**

**{**

**for(int i=left;i<=right;i++)**

**result.push\_back(matrix[top][i]);**

**top++;**

**for(int i=top;i<=bottom;i++)**

**result.push\_back(matrix[i][right]);**

**right--;**

**if(top<=bottom)**

**for(int i=right;i>=left;i--)**

**result.push\_back(matrix[bottom][i]);**

**bottom--;**

**if(left<=right)**

**for(int i=bottom;i>=top;i--)**

**result.push\_back(matrix[i][left]);**

**left++;**

**}**

**return result;**

**}**

1. **Search in 2D Matrix**

**2->Search in 2D matrix**

**bool searchMatrix(vector<vector<int>>& matrix, int target) {**

**int left = 0;**

**int right = (matrix.size()\*matrix[0].size())-1;**

**int c = matrix[0].size();**

**while(left<=right)**

**{**

**int mid = (left+right)/2;**

**if(matrix[mid/c][mid%c] == target)**

**return true;**

**else if(matrix[mid/c][mid%c] > target)**

**right = mid-1;**

**else**

**left = mid+1;**

**}**

**return false;**

**}**

1. **Median in a row wise sorted array**

**int binaryMedian(int m[][MAX], int r ,int c)**

**{**

**int min = INT\_MAX, max = INT\_MIN;**

**for (int i=0; i<r; i++)**

**{**

**// Finding the minimum element**

**if (m[i][0] < min)**

**min = m[i][0];**

**// Finding the maximum element**

**if (m[i][c-1] > max)**

**max = m[i][c-1];**

**}**

**int desired = (r \* c + 1) / 2;**

**while (min < max)**

**{**

**int mid = min + (max - min) / 2;**

**int place = 0;**

**// Find count of elements smaller than mid**

**for (int i = 0; i < r; ++i)**

**place += upper\_bound(m[i], m[i]+c, mid) - m[i];**

**if (place < desired)**

**min = mid + 1;**

**else**

**max = mid;**

**}**

**return min;**

**}**

1. **Row with max 1s**

**4.1-> Binary search on each row**

**int first(vector<int> arr,int l,int r)**

**{**

**while(l<=r)**

**{**

**int mid = (l+r)/2;**

**if((mid==0 or arr[mid-1]==0) and (arr[mid]==1))**

**return mid;**

**else if(arr[mid]==0)**

**l=mid+1;**

**else**

**r=mid-1;**

**}**

**return -1;**

**}**

**int rowWithMax1s(vector<vector<int> > arr, int r, int c) {**

**// code here**

**int index=0,res=-1,maxsum=INT\_MIN;**

**for(int i=0;i<r;i++)**

**{**

**index = first(arr[i],0,c-1);**

**if(index!=-1&&c-index > maxsum)**

**{**

**res = i;**

**maxsum = c-index;**

**}**

**}**

**return res;**

**}**

**4.2->**

**// The main function that returns index of row with maximum number of 1s.**

**int rowWithMax1s(bool mat[R][C])**

**{**

**// Initialize first row as row with max 1s**

**int max\_row\_index = 0;**

**// The function first() returns index of first 1 in row 0.**

**// Use this index to initialize the index of leftmost 1 seen so far**

**int j = first(mat[0], 0, C-1);**

**if (j == -1) // if 1 is not present in first row**

**j = C - 1;**

**for (int i = 1; i < R; i++)**

**{**

**// Move left until a 0 is found**

**while (j >= 0 && mat[i][j] == 1)**

**{**

**j = j-1; // Update the index of leftmost 1 seen so far**

**max\_row\_index = i; // Update max\_row\_index**

**}**

**}**

**return max\_row\_index;**

**}**