24csu350

Assingment:-01

Butterfly Pattern

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

void butterfly(int n) {

for(int i = 1; i <= n; i++) {

for(int j = 1; j <= i; j++) cout << "\*";

for(int j = 1; j <= 2\*(n-i); j++) cout << " ";

for(int j = 1; j <= i; j++) cout << "\*";

cout << endl;

}

for(int i = n; i >= 1; i--) {

for(int j = 1; j <= i; j++) cout << "\*";

for(int j = 1; j <= 2\*(n-i); j++) cout << " ";

for(int j = 1; j <= i; j++) cout << "\*";

cout << endl;

}

}

// Reverse array

void reverseArray(vector<int>& arr) {

reverse(arr.begin(), arr.end());

}

// Maximum and Minimum

pair<int,int> findMaxMin(const vector<int>& arr) {

return { \*min\_element(arr.begin(), arr.end()), \*max\_element(arr.begin(), arr.end()) };

}

// Sum and Average

pair<int,double> sumAndAverage(const vector<int>& arr) {

int sum = 0;

for(int num : arr) sum += num;

return {sum, (double)sum / arr.size()};

}

// Sort of array

void sortArray(vector<int>& arr) {

sort(arr.begin(), arr.end());

}

// linear search

int linearSearch(const vector<int>& arr, int target) {

for(int i = 0; i < arr.size(); i++) if(arr[i] == target) return i;

return -1;

}

//Remove duplicate

vector<int> removeDuplicates(vector<int> arr) {

sort(arr.begin(), arr.end());

arr.erase(unique(arr.begin(), arr.end()), arr.end());

return arr;

}

// longest consecutive

subsequence

int longestConsecutive(vector<int>& nums) {

sort(nums.begin(), nums.end());

nums.erase(unique(nums.begin(), nums.end()), nums.end());

int longest = 1, current = 1;

for(int i = 1; i < nums.size(); i++) {

if(nums[i] == nums[i-1] + 1) current++;

else current = 1;

longest = max(longest, current);

}

return longest;

}

// Matrix Transpose

void transpose(vector<vector<int>>& mat) {

int n = mat.size(), m = mat[0].size();

vector<vector<int>> result(m, vector<int>(n));

for(int i = 0; i < n; i++)

for(int j = 0; j < m; j++)

result[j][i] = mat[i][j];

mat = result;

}

// Multiplication of matrix

vector<vector<int>> multiply(vector<vector<int>>& A, vector<vector<int>>& B) {

int n = A.size(), m = B[0].size(), p = B.size();

vector<vector<int>> result(n, vector<int>(m, 0));

for(int i = 0; i < n; i++)

for(int j = 0; j < m; j++)

for(int k = 0; k < p; k++)

result[i][j] += A[i][k] \* B[k][j];

return result;

}

// Diagonal sum

pair<int,int> diagonalSums(const vector<vector<int>>& mat) {

int n = mat.size();

int primary = 0, secondary = 0;

for(int i = 0; i < n; i++) {

primary += mat[i][i];

secondary += mat[i][n-i-1];

}

return {primary, secondary};

}

// row with max sum

int rowWithMaxSum(const vector<vector<int>>& mat) {

int maxSum = -1, rowIndex = -1;

for(int i = 0; i < mat.size(); i++) {

int sum = 0;

for(int val : mat[i]) sum += val;

if(sum > maxSum) {

maxSum = sum;

rowIndex = i;

}

}

return rowIndex;

}

// 13.search in sorted Matrix

Bool search Matrix(const vector<vector<int>>& mat, int target) {

int i = 0, j = mat[0].size() - 1;

while(i < mat.size() && j >= 0) {

if(mat[i][j] == target) return true;

else if(mat[i][j] > target) j--;

else i++;

}

return false;

}

// Boundary elements

vector<int> boundaryElements(const vector<vector<int>>& mat) {

vector<int> result;

int n = mat.size(), m = mat[0].size();

for(int i = 0; i < m; i++) result.push\_back(mat[0][i]);

for(int i = 1; i < n-1; i++) result.push\_back(mat[i][m-1]);

if(n > 1)

for(int i = m-1; i >= 0; i--) result.push\_back(mat[n-1][i]);

for(int i = n-2; i >= 1; i--) result.push\_back(mat[i][0]);

return result;

}

// Saddle point

int saddlePoint(const vector<vector<int>>& mat) {

int n = mat.size(), m = mat[0].size();

for(int i = 0; i < n; i++) {

int minRow = mat[i][0], colIndex = 0;

for(int j = 1; j < m; j++) {

if(mat[i][j] < minRow) {

minRow = mat[i][j];

colIndex = j;

}

}

bool isMax = true;

for(int k = 0; k < n; k++) {

if(mat[k][colIndex] > minRow) {

isMax = false;

break;

}

}

if(isMax) return minRow;

}

return -1; // no saddle point

}