**S.Tejeswar – 22AD104 – AI & DS**

**(12-11-2024)**

1. String Anagram:

C++ Program:

class Solution {

public:

// Function is to check whether two strings are anagram of each other or not.

bool areAnagrams(string& s1, string& s2) {

// Your code here

int len1 = s1.size();

int len2 = s2.size();

if(len1 != len2){

return false;

}

unordered\_map<char, int> freqCount;

for(int i = 0; i < len1; i++){

freqCount[s1[i]] += 1;

}

for(int i = 0; i < len2; i++){

freqCount[s2[i]] -= 1;

}

for(auto& pair: freqCount){

if(pair.second != 0){

return false;

}

}

return true;

}

};

Time Complexity : O(NlogN)

Space Complexity: O(N)

1. Row with Max ones

C++ Program:

class Solution {

public:

int rowWithMax1s(vector<vector<int> > &arr) {

// code here

int row = arr.size();

int columns = arr[0].size();

int r = 0;

int c = columns - 1;

int maxOnesRowIndex = -1;

while(r < row && c >= 0){

if(arr[r][c] == 1){

maxOnesRowIndex = r;

c -= 1;

}

else{

r += 1;

}

}

return maxOnesRowsIndex;

}

};

Time Complexity: O(N + M)

Space complexity: O(1)

1. Longest Consequtive Subsequence

C++ Program:

class Solution {

public:

int rowWithMax1s(vector<vector<int> > &arr) {

// code here

int row = arr.size();

int columns = arr[0].size();

int r = 0;

int c = columns - 1;

int maxOnesRowIndex = -1;

while(r < row && c >= 0){

if(arr[r][c] == 1){

maxOnesRowIndex = r;

c -= 1;

}

else{

r += 1;

}

}

return maxOnesRowsIndex;

}

};

Time Complexity: O(N)

Space complexity: O(1)