

**UNIVERSITY INSTITUTE OF ENGINEERING**

**Department of Computer Science & Engineering**

**Subject Name:** Competitive coding-I

**Subject Code:** 20CSP-314

|  |  |
| --- | --- |
| **Submitted By:** Kanishk Soni **Submitted To: Mr. Syed Abdul Basit** | |
| **Subject Name** | CC LAB |
| **Subject Code** | 20CSP-314 |
| **Branch** | Computer Science |
| **Semester** | 5th |

**Practical Evaluation Sheet**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sr.No** | **Program** | **Date** | **Evaluation** | | | | **Sign** |
| **Conduct**  **(12)** | **Viva(8)** | **Worksheet(10)** | **Total (30)** |
| 1 | Concept of Arrays |  |  |  |  |  |  |
| 2 | Concept of Stack and Queues |  |  |  |  |  |  |
| 3 | Concept of Linked List |  |  |  |  |  |  |
| 4 | Concept of Searching and Sorting |  |  |  |  |  |  |
| 5 | Concept of Graph |  |  |  |  |  |  |
| 6 | Concept of Trees |  |  |  |  |  |  |
| 7 | Concept of String |  |  |  |  |  |  |
| 8 | Concept of Dynamic Programming |  |  |  |  |  |  |
| 9 | Concept of Backtracking |  |  |  |  |  |  |
| 10 | Concept of Greedy and Branch and Bound |  |  |  |  |  |  |

**Experiment 9**

**Student Name: Kanishk Soni UID: 20BCS9398**

**Branch: CSE Section/Group:707\_WM\_B**

**Semester: 5th Date of Performance: 07/11/2022**

**Subject Name: Competitive Coding - I Subject Code: 20CSP-314**

**Question1: Queens-on-Board**

**Code:**

#include <vector>

#include <string>

#include <algorithm>

#include <iostream>

#include <unordered\_map>

#include <cassert>

using namespace std;

struct Solution2 {

typedef basic\_string<unsigned char> \_\_Board;

typedef \_\_Board::value\_type \_\_Row;

long long solve(const vector<string> & B){

if (B.empty() || B[0].empty())

return 0;

for (size\_t i = 0; i < B.size(); ++i) {

\_\_Row row = 0;

for (size\_t j = 0; j < B[i].size(); ++j) {

if ('.' == B[i][j])

row |= (1 << j);

}

row = ~row;

board.push\_back(row);

\_\_Board p;

genPlacements(row, p, B[i].size());

placements.push\_back(p);

}

bmask = (1 << B[0].size()) - 1;

return help(0, 0, 0, 0);

}

private:

static void genPlacements(\_\_Row block, \_\_Board & ret, int M){

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

\_\_Row p1 = 1 << i;

if (0 != (p1 & block))

continue;

ret.push\_back(p1);

for (int j = i + 2; j < M; ++j){

\_\_Row p2 = p1 | (1 << j);

if (0 != (p2 & block))

continue;

\_\_Row m2 = (1 << j) - (1 << (i + 1));

if (0 == (m2 & block))

continue;

ret.push\_back(p2);

for (int k = j + 2; k < M; ++k){

\_\_Row p3 = p2 | (1 << k);

if (0 != (p3 & block))

continue;

\_\_Row m3 = (1 << k) - (1 << (j + 1));

if (0 == (m3 & block))

continue;

ret.push\_back(p3);

}

}

}

}

\_\_Row calcMask(\_\_Row mask, \_\_Row blocks){

\_\_Row b = mask & blocks;

mask &= ~b;

return (mask & bmask);

}

static int hash(size\_t row, \_\_Row lmask, \_\_Row dmask, \_\_Row rmask){

int r = row;

r <<= 8;

r += lmask;

r <<= 8;

r += dmask;

r <<= 8;

r += rmask;

return r;

}

long long help(size\_t row, \_\_Row lmask, \_\_Row dmask, \_\_Row rmask){

if (row >= board.size())

return 0;

const int h = hash(row, lmask, dmask, rmask);

unordered\_map<int, long long>::const\_iterator wh = save.find(h);

if (wh != save.end())

return wh->second;

const \_\_Row blocks = board[row];

const \_\_Row mask = lmask | dmask | rmask | blocks;

long long ret = 0;

lmask = calcMask(lmask, blocks);

dmask = calcMask(dmask, blocks);

rmask = calcMask(rmask, blocks);

if (\_\_Row(-1) != mask){

const \_\_Board & ps = placements[row];

for (size\_t i = 0; i < ps.size(); ++i){

const \_\_Row p = ps[i];

if (0 != (mask & p))

continue;

++ret;

ret += help(row + 1, (lmask | p) << 1, dmask | p, (rmask | p) >> 1);

}

}

ret += help(row + 1, lmask << 1, dmask, rmask >> 1);

return (save[h] = ret % 1000000007);

}

\_\_Board board;

vector<\_\_Board> placements;

unordered\_map<int, long long> save;

\_\_Row bmask;

};

typedef Solution2 Solution;

int main(){

int t;

cin >> t;

while (t--){

int n, m;

cin >> n >> m;

vector<string> b;

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

string line;

cin >> line;

b.push\_back(line);

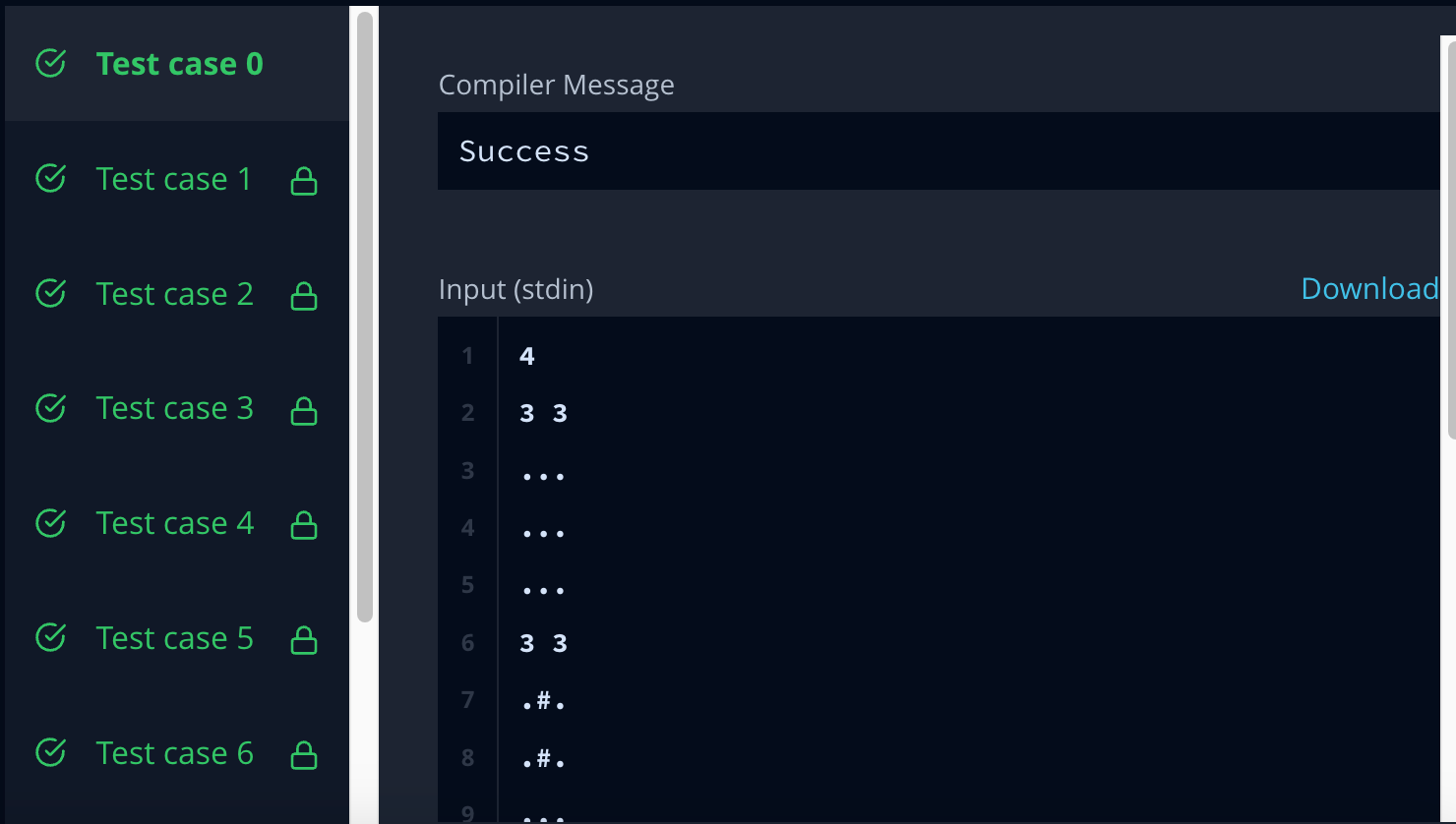
}

cout << Solution().solve(b) << endl;

}

return 0;

}



**Question 2:** **Subset-sum**

**Code:**

import Data.List (sortOn, intercalate) import Data.Ord (Down(Down))

import Data.Array (listArray, bounds, (!)) fromList xs = listArray (1, length xs) xs

-- arr is non-decreasing help arr low (beg, end)

| beg == end = beg

| val >= low = help arr low (beg, med)

| otherwise = help arr low (med+1, end) where med = (beg + end) `div` 2

val = arr ! med search arr low

| low <= arr ! beg = Just beg

| low > arr ! end = Nothing

| otherwise = Just $ help arr low (beg, end) where bnds = bounds arr

beg = fst bnds end = snd bnds

ans Nothing = -1 ans (Just x) = x

main = do

ls <- fmap lines getContents

let as = map read $ words $ ls !! 1 :: [Int] ss = map read $ drop 3 ls :: [Int]

arr = fromList $ scanl1 (+) $ sortOn Down as

putStrLn $ intercalate "\n" $ map (show . ans . search arr) ss

