CHANDIGARH UNIVERSITYUNIVERSITYINSTITUTEOFENGINEERING

DEPARTMENTOFCOMPUTERSCIENCEANDENGINEERING

StudentName: Aniket Kumar **UID:**20BCS5306

Branch:BE-CSE **Semester:**5th

SubjectName: CompetitiveCoding **Section/Group:**20BCS_WM−703/B

LAB INDEX

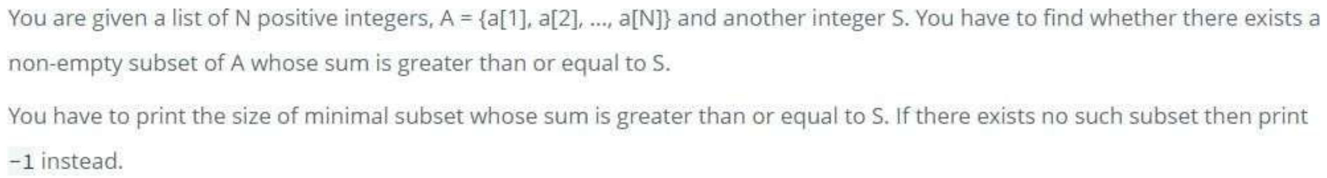
Sr. No	Program	Date	Evaluation				Sign
			LW (12)	VV (8)	FW (10)	Total (30)	
1.	Todemonstratetheconceptofarrays						
2.	Todemonstratetheconceptofstackand queues						
3.	Todemonstrate theconcept ofLinkedlist						
4.	Todemonstratetheconcept ofSearchingandsorting						
5.	Todemonstrate theconcept ofGraphs						
6.	Todemonstrate theconcept ofTrees						
7.	Todemonstrate theconcept ofStrings						
8.	Todemonstratetheconceptof Dynamicprogramming						
9.	TodemonstratetheconceptofBacktracking						
10.	To demonstrate theconcept ofGreedyand Branchand Bound						

EXPERIMENTNUMBER–9

9.1 AIMOFTHEEXPERIMENT–

Backtracking

TASKTOBEDONE/LOGISTICSUSED--



You are given a list of N positive integers, A = {a[1], a[2], ..., a[N]} and another integer S. You have to find whether there exists a non-empty subset of A whose sum is greater than or equal to S.

You have to print the size of minimal subset whose sum is greater than or equal to S. If there exists no such subset then print -1 instead.

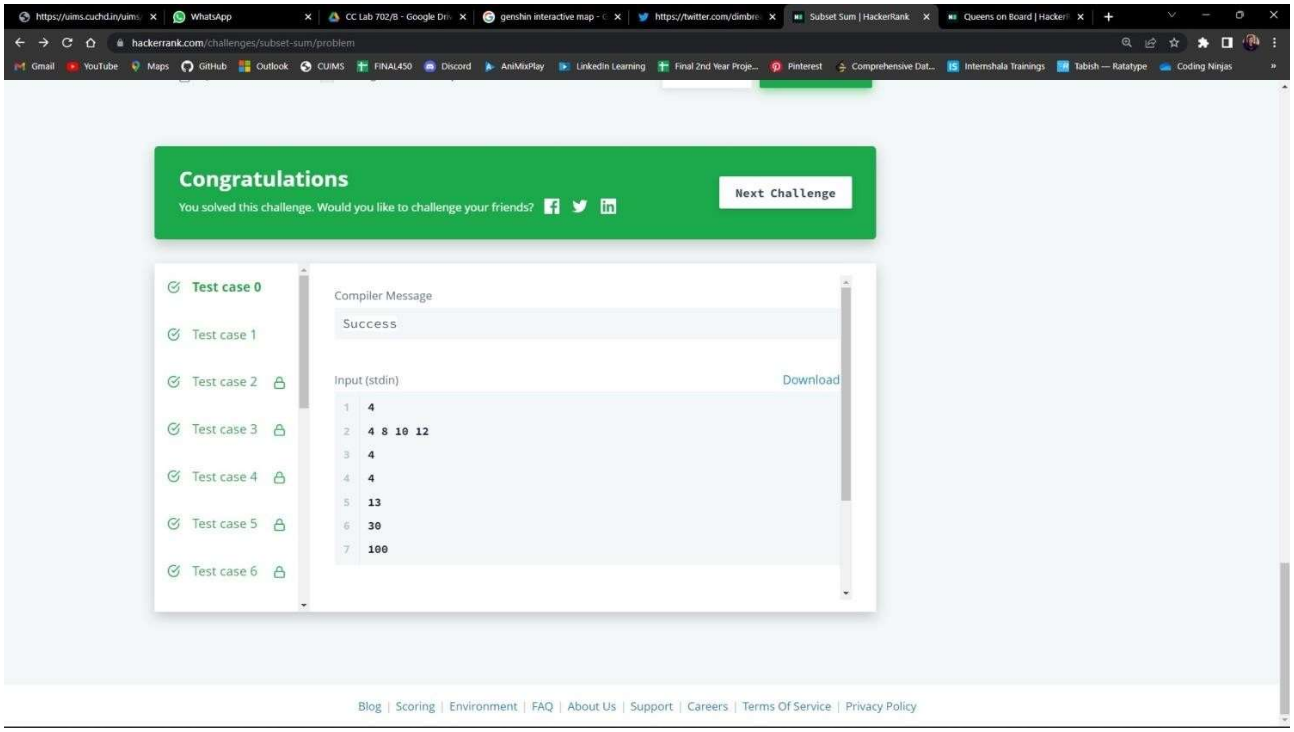
https://www.hackerrank.com/challenges/subset-sum/problem

CODE:(INC++)

```
importData.List(sortOn,intercalate)imp
ortData.Ord (Down(Down))
importData.Array(listArray,bounds,(!))from

Listxs= listArray(1, lengthxs)xs

--arrisnon-
decreasinghelparrlow
(beg,end)
  | beg ==end=beg
  | val >= low=help arrlow (beg, med)
  | otherwise = help arr low (med+1,
  end)wheremed=(beg+end)`div`2
        val = arr !

medsearch arr low
```

```
  | low<= arr!beg=Just beg
  |low>arr! end= Nothing
  | otherwise = Just $ help arr low (beg,
  end)where bnds=boundsarr
        beg=fstbndsend
        =sndbnds

ansNothing=-
1ans(Justx)=x

main=do
  ls<-fmaplinesgetContents
  let as = map read $ words $ ls !! 1 ::
      [Int]ss=mapread$drop3ls::[Int]
      arr=fromList$scanl1 (+) $ sortOnDown as
  putStrLn$ intercalate "\n"$map (show .ans .search arr)ss
```

OUTPUT:



Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

✓ Test case 0

✓ Test case 1

✓ Test case 2

✓ Test case 3

✓ Test case 4

✓ Test case 5

✓ Test case 6

Compiler Message

```
Success
```

Input (stdin)   Download

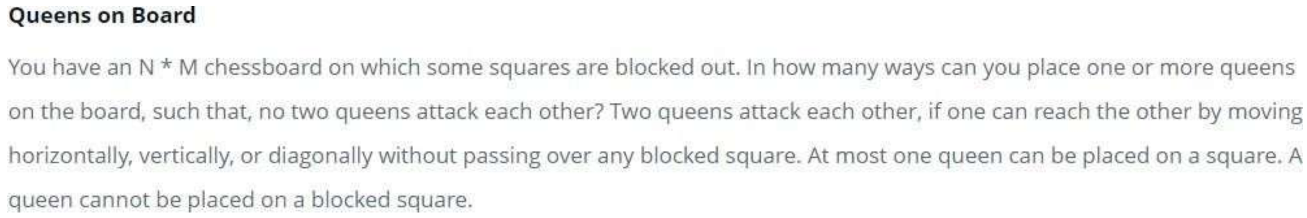
```
4
4 8 10 12
4
4
13
30
100
```

Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy

9.2 AIMOFTHEEXPERIMENT–

Backtracking

TASKTOBEDONE/LOGISTICSUSED–



Queens on Board

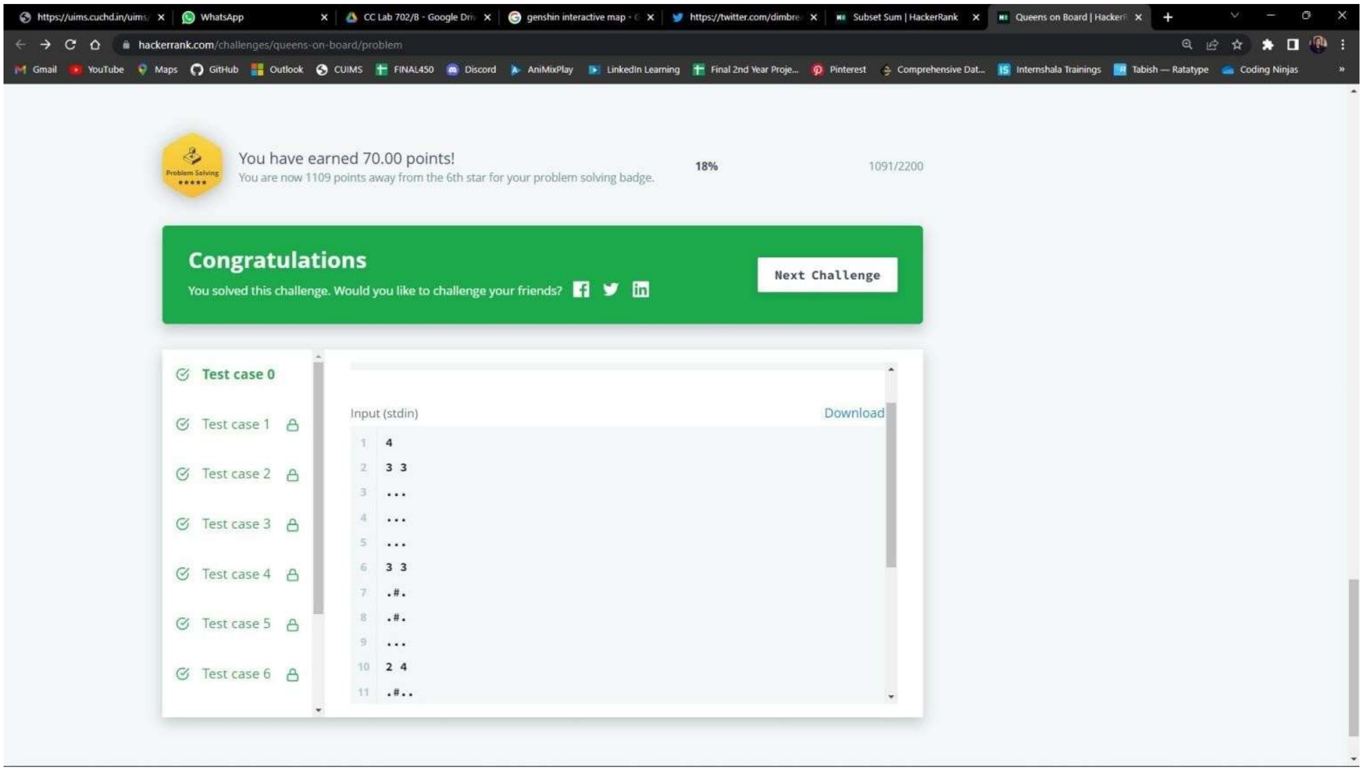
You have an N * M chessboard on which some squares are blocked out. In how many ways can you place one or more queens on the board, such that, no two queens attack each other? Two queens attack each other, if one can reach the other by moving horizontally, vertically, or diagonally without passing over any blocked square. At most one queen can be placed on a square. A queen cannot be placed on a blocked square.

https://www.hackerrank.com/challenges/queens-on-board/problem

CODE:(INC++)

```
#include
<vector>#include
<string>#include
<algorithm>#include
<iostream>#include<unord
ered_map>#include<casser
t>

usingnamespacestd;st

ructSolution2
{
    typedefbasic_string<unsignedchar>Board;
    typedef__Board::value_type           Row;
    long long solve(const vector<string>&
        B){if (B.empty() || B[0].empty())
            return0;

        for(size_ti =0;i<B.size();++i){
            __Rowrow=0;
            for(size_t j= 0; j <B[i].size();++j){
                if('.'==B[i][j])
                    row|=(1<<j);
            }
            row =
            ~row;board.push_back(row);

              __Boardp;
            genPlacements(row, p,
            B[i].size());placements.push_back(p
            );
        }
        bmask = (1 << B[0].size()) -
        1;returnhelp(0,0,0,0);
    }
private:
    staticvoidgenPlacements(Row
        block,Board&ret,intM){for(inti=0;i<M;++i){

            __Rowp1=1<< i;
            if (0 != (p1 &
                block))continue;
            ret.push_back(p1);

            for (intj=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){
                    Rowp3=p2|(1<<k);if(0!=(p3&
                    block))
                        continue;
                    Rowm3= (1<< k)-(1 <<(j +1));
                    if(0==(m3&block))
                        continue;   //there is not enough blocks between 3
                    Qsret.push_back(p3);
                }
            }
        }
    }
    RowcalcMask(Rowmask,Rowblocks){
        Rowb =mask & blocks;
        mask&=~b;
        return(mask& bmask);
    }
    static int hash(size_t      __Rowlmask,Row dmask,Rowrmask){
        row,intr=row;
        r<<=8;
        r+=lmask;r
        <<= 8;
        r+=dmask;r
        <<= 8;
        r+=rmask;re
        turnr;
    }
    long long help(size_t row,Row lmask,if  __Rowdmask,Row rmask){
        (row >= board.size())
            return0;
        constinth =hash(row,lmask,dmask, rmask);
        unordered_map<int, long long>::const_iterator wh =
        save.find(h);if(wh!=save.end())
            returnwh->second;
        constRow blocks=board[row];
        constRow mask = lmask | dmask | rmask | blocks;long
        longret=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){constRow p=ps[i];
                if(0!=(mask&p))conti
                    nue;
                ++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]=r
        et%1000000007);
    }
    Boardboard;
    vector<Board>
    placements;unordered_map<int,longlo
    ng>save;
    Rowbmask;
};

typedefSolution2Solution;

intmain()
{

    int
    t;cin>>t
    ;
    while(t--){
        intn,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;
    }
    return0;
}
```

OUTPUT:



You have earned 70.00 points!

You are now 1109 points away from the 6th star for your problem solving badge.

18%

1091/2200

Congratulations

You solved this challenge. Would you like to challenge your friends?

- ✓ Test case 0
- ✓ Test case 1
- ✓ Test case 2
- ✓ Test case 3
- ✓ Test case 4
- ✓ Test case 5
- ✓ Test case 6

Input (stdin)

```
4
3 3
...
...
...
3 3
.#.
.#.
...
2 4
.#..
```

LearningOutcomes(WhatIhavelearnt):

- Itwillprovidethemodestexperiencethatallowsstudentstodevelopandimprovetheire xperimentalskillsand developability to analyzedata.
- Ability to demonstrate the practical skill on measurements and instrumentationtechniques of some Physics experiments. Students will develop the ability to useappropriatephysical conceptstoobtainquantitativesolutionstoproblemsinphysics.
- Studentswilldemonstratebasicexperimentalskillsbysettinguplaboratoryequipment safely and efficiently, plan and carry out experimental procedures, andreportverballyandin written languagetheresultsoftheexperiment.
- Studentswilldevelopskillsbythepracticeofsettingupandconductinganexperiment with dueregardsto minimizing measurementerror.