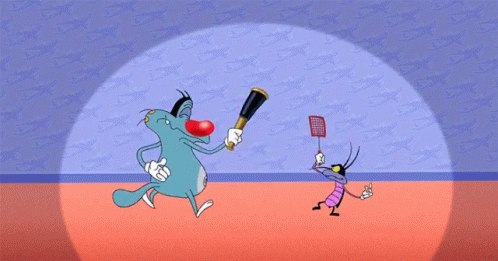
Challenge #8 [Amazon] - Oggy and his cockroaches

In the previous challenge, you successfully got out of the tunnel. As you reached the other side, you see the Oggy having a duel with one of the cockroaches. Oggy also happens to be in-charge of the cruise ship (a cat in-charge of a cruise ship? Well, if Suppandi can solve a math problem, then anything is possible).



To get onto the ship, you get in line to buy a ticket. The line is very long, so you buy the special pass. The special pass allows you to move ahead in the line.

Using this special pass, if you are at the ith position in the line, you can move forward by x steps where 1 <= x <= A[i]. Then, when you get to a new position, you can again move by the same logic.

For example, let's say line is like this: 4 1 2 2 3 3 1 2 4 4:

1. At the first position, you can move forward by 1, 2, 3 or 4 stepssince A[0] is 4
2. At the 2nd position, you can move forward by 1 step only (since A[1] is 1)
3. At the 3rd position you man move forward by 1 or 2 steps (since A[2] is 2)
4. And so on...

In order to reach the end of the line, the minimum number of steps you require are 3. A[0] -> A[4]; A[4] -> A[7]; A[7] -> A[9]

Given the various line positions, find the minimum number of steps to reach the end line.

**Input format**

* The first line contains T, the number of inputs
* The following lines contain N followed by space-separated numbers A1, A2, A3.... AN

**Output format**

* Print minimum number of steps for each test case

**Constraints**

* 1 <= A[i] <= 1000
* 1 <= N <= 1000

**Example Input**

|  |  |
| --- | --- |
|  | 3 |
|  | 3 7 8 7 |
|  | 2 2 1 |
|  | 10 4 1 2 2 3 3 1 2 4 4 |

**Example Output**

|  |  |
| --- | --- |
|  | 1 |
|  | 1 |
|  | 3 |

**Explanation**

1. We can reach A[2] directly from A[0], because we are allowed 1,2,.... 7 steps and we just need 2 steps
2. We can reach A[1] in just 1 step
3. Explained in the example above

Ans-Code

#include <bits/stdc++.h>

using namespace std;

int getMinimum(int a[] ,int n)

{

int min = a[0];

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

{

if(a[i] == INT\_MAX)

break;

if(a[i] < min)

min = a[i];

}

return min;

}

int main()

{

int T,n;

cin>>T;

for(int t = 1 ; t <= T; t++)

{

cin>>n;

int a[n];

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

{ cin>>a[i];}

int p[n][n];

//start processing

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

{

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

{

p[i][j] = 0;

}

}

for(int i = 0; i <= n-2; i++)

{

p[i][i+1] = 1;

}

for(int diff = 2; diff < n; diff++)

{

for(int i = n - diff - 1 ; i >= 0; i--)

{

int j = i + diff;

if(a[i] + i >= j)

p[i][j]++;

else

{

int m = 0;

int myarray[a[i]];

for(int w = 0; w < a[i]; w++)

myarray[w] = INT\_MAX;

for(int k = i + 1; k <= a[i] + i; k++)

{

if(k < n)

{

myarray[m] = p[i][k] + p[k][j];

m++;

}

}

p[i][j] = getMinimum(myarray,a[i]);

}

}

}

cout<<p[0][n-1];

cout<<"\n";

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

{

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

{

cout<<p[i][j]<<" ";

}

cout<<endl;

}

}

return 0;

}

**Question: 7**

Challenge #7 [HackWithInfy] - Dora's Batteries

Shinchan is overjoyed that you were able to clear the maze. Good job! His other friends haven't been so lucky. While you are on Party Island**TM**it seems everyone is enjoying themselves. Noddy doesn't look like he's heading back any time soon. You, on the other hand, want to get home to start practicing for the upcoming ***Edyst Web Development Course***.

You meet **Dora, the explorer**, who says that there is a way back home using the ship. You've always wanted to experience a cruise - so you agree to follow her.

To get to the ship though, you need to cross a tunnel. The tunnel is very long and we will need flashlights to see in the dark.



Dora has a pack of N batteries.

* Given a battery of certain rating Ai, the battery's power is the number of set bits in Ai
* When two batteries Ai and Aj combine, their combined power becomes  Ai || Aj, where || is the bitwise OR operator

Dora had carefully packed the batteries such that the power of the entire set of batteries is just enough to help us cross the tunnel. However, now we need to remove some batteries, without hurting the overall power of the set of batteries, because we don't have enough space in the backpack.

Given a set of N battery ratings, choose the minimum number of batteries min such that the power of the min batteries is the same as power of N batteries.

**Constraints:**

* 1 <= N <= 105
* 1 <= Ai <= 106

**Input format**

* The first line contains T, the number of test cases. Each of the following T lines contain:
* N followed by N numbers A1, A2, A3...... An

**Output format**

* Print min batteries required for each test case

**Example Input**

|  |  |
| --- | --- |
|  | 3 |
|  | 4 3 5 7 13 |
|  | 4 2 4 8 16 |
|  | 4 5 3 1 7 |

**Output**

|  |  |
| --- | --- |
|  | 2 |
|  | 4 |
|  | 1 |

**Explanation:**

**Case 1:**we have powers: 3: 011, 5: 101, 7: 111, 13: 1101. Combining 13 and 7 will give us 15 : 1111, which is the same as the bitwise OR of entire batteries. Thus answer is 2 - only 2 batteries needed

**Case 2:**we need all 4 numbers to reach combined powers of  11110

**Case 3:** we only need 7 ( 111) to reach the same combined power as the entire batteries

**Ans:**

#include<bits/stdc++.h>

using namespace std;

int main(){

int T,n,TotalPower;

cin>>T;

for(int test = 1; test <= T; ++test)

{

cin>>n;

int a[n];//array declaration

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

cin>>a[i];

//proceed

TotalPower = 0;

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

TotalPower = TotalPower | a[i];

sort(a, a+n, greater<int>()); //sort in deaccending order

//minimum battery required

int minPower = 0,prevPower = 0;

int count = 0;

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

{

minPower = minPower | a[i];

count++;

if(prevPower == minPower)

{

count--;

}

if(minPower == TotalPower)

break;

prevPower = minPower;

}

cout<<count<<endl;

}

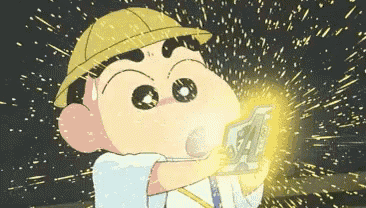
return 0;

}

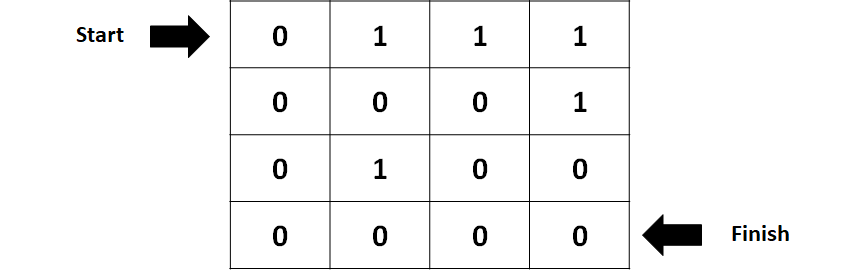
Challenge #6 [EPAM] - Shinchan's Party Games

With your help, Noddy has been able to transport the both of you to the grand party on Party Island**TM**

Over there, the games have begun. One such game is the **maze game**, which Shinchan has set up.



The maze looks like this:



In the maze, you start at the top left 1,1, and make your way down to n,n. At each step, you are only allowed 3 moves:

* You can move to the right (east)
* You can move down (south)
* If you are on the left diagonal (row number and column number are same), then you can move down and right (south-east) in one move

However, you cannot move to the block if it is an obstacle. You can only move if there is a path. 1 denotes an obstacle and 0 the path.

You have to print the minimum number of steps to reach the destination (bottom-right) from the starting position (top-left).

In the above diagram, the minimum number of steps required is 3 steps:

|  |  |
| --- | --- |
|  | Step 1: (1,1) -> (2,2) |
|  | Step 2: (2,2) -> (3,3) |
|  | Step 3: (3,3) -> (4,4) |

**Input format**

* The first line contains T the number of test cases
* Each test case contains n and n, the number of rows and columns (it is always a square maze)
* This is followed by n space separated numbers in n lines

**Output format**

* For each test case, print the minimum number of steps required to reach the destination

Note: the start and destination will always be marked as 0 in the map

**Example Input**

|  |  |
| --- | --- |
|  | 2 |
|  | 3 3 |
|  | 0 1 1 |
|  | 1 0 1 |
|  | 0 0 0 |
|  | 4 4 |
|  | 0 0 0 1 |
|  | 0 0 1 0 |
|  | 0 0 1 0 |
|  | 1 0 0 0 |

**Example Output**

|  |  |
| --- | --- |
|  | 2 |
|  | 5 |

**Explanation**

* In first case, move along the diagonal and you will reach the destination in 2 steps
* The steps are: (1,1) -> (2,2); (2,2) -> (3,2); (3,2) -> (4,2); (4,2) -> (4,3); (4,3) -> (4,4)

**Note**: students coding in Java should keep their class name as Main

Ans:

#include<bits/stdc++.h>

using namespace std;

void setValue(int \*Maze,int N)

{

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

{

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

{

cin>>\*((Maze+i\*N) + j);

}

}

}

int pathCount(int \*Maze,int N,int i,int j,int count)

{

//cout<<"i :"<<i<<" j:"<<j<<endl;

int a,b,c;

if(i == N-1 && j == N-1)

return count;

count++;

if(i+1 < N && j+1 < N && i == j && \*((Maze+(i+1)\*N) + (j+1)) == 0)//diagonally

a = pathCount((int \*)Maze,N,i+1,j+1,count);

else

a = INT\_MAX;

if(j+1 < N && \*((Maze+i\*N) + (j+1)) == 0)//

b = pathCount((int \*)Maze,N,i,j+1,count);//Right

else

b = INT\_MAX;

if(i+1 < N && \*((Maze+(i + 1)\*N) + j) == 0)//Down(Below)

c = pathCount((int\*)Maze,N,i+1,j,count);

else

c = INT\_MAX;

int x = min(a,b);

return min(x,c);

}

int main(){

int notc;

//cout<<"Enter No of Test Cases:";

cin>>notc;

int result[notc];

int M,N;

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

{

cin>>M>>N;

if(M == N)

{

int Maze[N][N];

setValue((int \*)Maze,N);

int x = pathCount((int \*)Maze,N,0,0,0);

result[i] = x;

}

else

{

//cout<<"Not a sqaure matrix:";

}

}

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

{

cout<<result[i]<<endl;

}

return 0;

}

#include<bits/stdc++.h>

using namespace std;

void setValue(int \*Maze,int N)

{

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

{

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

{

cin>>\*((Maze+i\*N) + j);

}

}

}

int pathCount(int \*Maze,int N,int i,int j,int count)

{

//cout<<"i :"<<i<<" j:"<<j<<endl;

int a,b,c;

if(i == N-1 && j == N-1)

return count;

count++;

if(i+1 < N && j+1 < N && i == j && \*((Maze+(i+1)\*N) + (j+1)) == 0)//diagonally

a = pathCount((int \*)Maze,N,i+1,j+1,count);

else

a = INT\_MAX;

if(j+1 < N && \*((Maze+i\*N) + (j+1)) == 0)//

b = pathCount((int \*)Maze,N,i,j+1,count);//Right

else

b = INT\_MAX;

if(i+1 < N && \*((Maze+(i + 1)\*N) + j) == 0)//Down(Below)

c = pathCount((int\*)Maze,N,i+1,j,count);

else

c = INT\_MAX;

int x = min(a,b);

return min(x,c);

}

int main(){

int notc;

//cout<<"Enter No of Test Cases:";

cin>>notc;

int result[notc];

int M,N;

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

{

cin>>M>>N;

if(M == N)

{

int Maze[N][N];

setValue((int \*)Maze,N);

int x = pathCount((int \*)Maze,N,0,0,0);

result[i] = x;

}

else

{

//cout<<"Not a sqaure matrix:";

}

}

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

{

cout<<result[i]<<endl;

}

return 0;

}