Task A11. Tunnels

⏳ 1.25 s   💾 1024 MB

Author: Emil Indzhev

The Rabbit has dug a system of tunnels which has N levels (numbered top to bottom: from 0 to $N-1$). Each level has M chambers (numbered left to right: from 0 to $M-1$). We can think of the chambers as being cells on an N by M grid. There are tunnels between all pairs of adjacent chambers (sharing a side). Below all of this, there is a big treasure room (effectively spanning the whole level N). There are tunnels between it and all chambers on level $N-1$. However, some of the NM chambers have collapsed and are thus blocked (as are all of their tunnels).

The Rabbit wants to protect his treasure so he has booby-trapped almost all vertical tunnels. More precisely, each level (including the treasure room's level) has exactly one safe (non-trapped) incoming vertical tunnel. Additionally, it is guaranteed there is a safe path (a path passing through no trapped tunnels) from the surface to the treasure room.

Alice wants to get to the Rabbit's treasure room, but going into the tunnel system blind would be extremely dangerous. First, she studied the chambers using sonar and figured out which ones are blocked. However, she still has no idea which tunnels are trapped and which are safe. Next, she started observing the Rabbit. She knows that he's always in a rush, so he always takes the shortest safe (no trapped tunnels) path from the surface to the treasure (and vice-versa), i.e. the unique safe path that does not repeat chambers. She has also seen him entering the tunnel system (namely level 0) from the surface above chamber K ($0 \le K < M$). This allows her to safely enter the tunnel system. Finally, Alice brought a bloodhound that knows the Rabbit's smell. Using the dog, she can go to a reachable chamber (in her current level) and investigate it – check whether the Rabbit has been there or not. After a series of these investigations, when she's certain, she can go to a reachable chamber and go deeper (to the next level) from there. She doesn't want to risk her life, so she must be absolutely sure that the chosen vertical tunnel is safe. She wants to reach the treasure safely using as few investigations as possible in the worst case (she's also in a rush, as she has to attend a tea party).

Formally, we define the worst case number of investigations of an exploration strategy for a given test case (N, M, K and set of blocked chambers) as so: Enumerate all possible sets of safe tunnels, compute the Rabbit's path on each, and run the strategy on each of those cases (independently). The strategy's worst case number of investigations is the maximum number of investigations it does on any one of those cases (assuming it is valid and manages to safely get to the treasure in all of them; otherwise, we say its worst case number of investigations is infinity). We then enumerate all possible strategies and find the minimum possible worst case number of investigations. That is the maximum allowed number of investigations for the test case.

Alice is a clever girl, but the Rabbit's tunnel system is vast, so even she can't figure out an optimal exploration strategy. Help her by writing a program that explores the tunnel system and reaches the treasure room using no more than the allowed number of investigations for the test case, without going through blocked chambers or passing through trapped tunnels. (Your program needs to fulfill all of these requirements to correctly pass a given test case.)

Implementation details

You have to implement the function `solve`:

```
void solve(int n, int m, int k, const std::vector<std::vector<bool>>& blocked)
```

It will be called once per test case with N, M and K, as well as a description of the chambers (indexed as `blocked[level][position]`). It must perform the exploration of the tunnels (starting from level 0, chamber K) to reach the level of the treasure room (level N) using no more than the minimum possible worst case number of investigations for the test case. From this function (and other functions you write), you can call the functions `investigate` and `goDeeper`:

```
bool investigate(int s)
```

Calling `investigate` represents going to chamber S in the current level (it must be reachable, i.e. there must not be any blocked chambers on the way) and checking whether the Rabbit passes through this chamber on his way to the treasure. The function returns `true`, if he does, and `false`, otherwise.

```
void goDeeper(int s)
```

Calling `goDeeper` represents going to chamber S in the current level (it must be reachable) and then descending down to the next level using its vertical tunnel. The tunnel must be safe (i.e. not trapped) and the chamber below must not be blocked. After that, you continue your exploration from chamber S of the next level (unless you reached level N, in which case your `solve` function should return).

Your code will be compiled together with a grader, so it should not implement a `main` function, nor should you read from `stdin` or write to `stdout`. You also need to include the header `tunnels.h`.

The grader may sometimes be adaptive (but still deterministic), i.e. it may decide what `investigate` returns or whether `goDeeper` is successful, based on your program's past actions, without having a fixed set of safe tunnels. However, it is guaranteed that everything that the grader does will be valid with respect to some possible set of safe tunnels. The grader may terminate the execution, if you call `goDeeper` and try to pass through a trapped tunnel or if the grader decides that it is no longer possible for you to solve the test case using the required number of investigations. Additionally, the runtime of the grader is not counted towards the time limit.

Local testing

To test your program locally, a local grader and a header file are provided. The local grader is not adaptive and does no computations or verifications. It reads N, M, K and a matrix of 0s and 1s (without spaces), calls your `solve` and then outputs whenever your program calls `investigate` or `goDeeper` (expecting input for each `investigate` – a 0 or a 1 that it will return). You are free to modify the local grader.

Constraints

- $1 \le N, M \le 5000$

Subtasks

Subtask	Points	Constraint
1	5	$M = 2$
2	16	No chambers are blocked.
3	19	$N, M \le 100$
4	19	$N, M \le 400$
5	20	$N, M \le 1000$
6	21	None.

You get the points for a given subtask, only if you correctly pass all tests in it and all other subtasks that are included in it.

Sample test

Input	Interaction
2 3 1 001 100	solve(2, 3, 1, {{0, 0, 1}, {1, 0, 0}}) goDeeper(1) investigate(2): return true goDeeper(2)

The minimum possible worst case number of investigations for this test is 1, so the solution would pass this test.