MAZE

- Một mê cung hình chữ nhật được biểu diễn bởi 0-1 ma trận NxM trong đó A[i,j] = 1 thể hiện ô (i,j) là tường gạch và A[i,j] = 0 thể hiện ô (i,j) là ô trống, có thể di chuyển vào. Từ 1 ô trống, ta có thể di chuyển sang 1 trong 4 ô lân cận (lên trên, xuống dưới, sang trái, sang phải) nếu ô đó là ô trống. Xuất phát từ 1 ô trống trong mê cung, hãy tìm đường ngắn nhất thoát ra khỏi mê cung.
- **Input**
  - Dòng 1: ghi 4 số nguyên dương n, m, r, c trong đó n và m tương ứng là số hàng và cột của ma trận A (1 <= n,m <= 999) và r, c tương ứng là chỉ số hàng, cột của ô xuất phát.
  - Dòng i+1 (i=1,...,n): ghi dòng thứ i của ma trận A
- **Output**
  - Ghi giá số bước cần di chuyển ngắn nhất để thoát ra khỏi mê cung, hoặc ghi giá trị -1 nếu không tìm thấy đường đi nào thoát ra khỏi mê cung.

MAZE

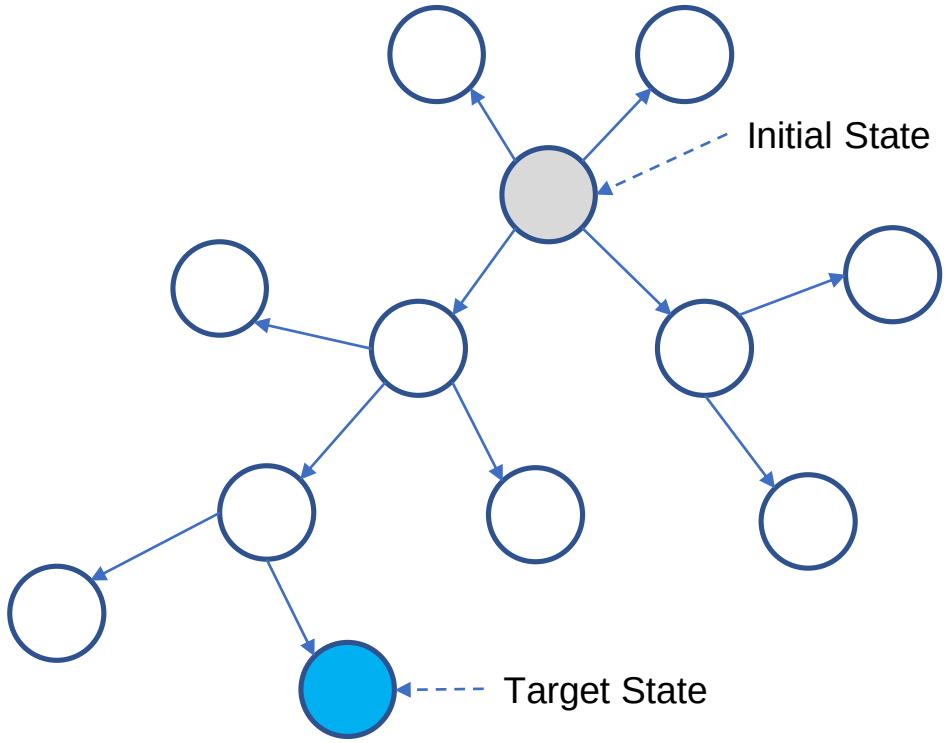
- Example

stdin	stdout
8 12 5 6 1 1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 1 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 1 1 0 0 1 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 1 0 1 1 1 0 1 0 1	7

Implementation

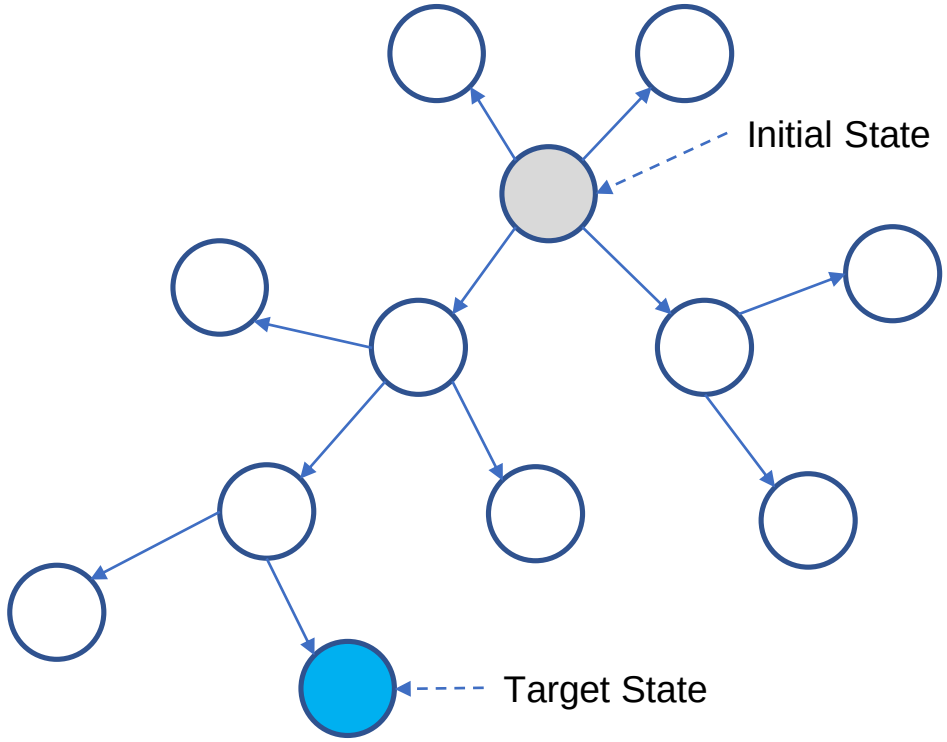
- Find the shortest path in the state transition diagram (graph)

```
findPath(s0, N){// N(s): set of neighboring states of s
  Initialize a Queue
  Queue.PUSH(s0); visited[s0] = true;
  while Queue not empty do{
    s = Queue.POP();
    for x ∈ N(s) do{
       if not visited[x] and check(x) then{
            if target(x) then return x;
            else{
               Queue.PUSH(x);  visited[x] = true;
            }
       }
    }
  }
}
```



Implementation

- Find the shortest path in the state transition diagram (graph)



Implementation

```
#include <bits/stdc++.h>

using namespace std;

typedef pair<int,int> ii;

const int maxN = 999 + 100;

const int oo= 1e9 + 7;

int a[maxN][maxN] , m , n , r, c , d[maxN][maxN];

int dx[] = {1 , 0, -1 ,  0} ,

    dy[] = {0 , 1,  0 , -1};

queue<ii> qe;
```

Implementation

```
int solve(){
    qe.push(ii(r,c));  d[r][c] = 0;   a[r][c] = 1;
    while(!qe.empty()){
        ii u = qe.front(); qe.pop();
        for(int i = 0 ; i < 4 ; i++){
            int x = dx[i] + u.first;          int y = dy[i] + u.second;
            if(x < 1 || x > m || y < 1 || y > n) return d[u.first][u.second] + 1;
            if(a[x][y] != 1){
                    d[x][y] = d[u.first][u.second] + 1;
                    qe.push(ii(x,y));
                    a[x][y] = 1;
            }
        }
    }
    return -1;
}
```

Implementation

```
int main(){
    ios_base::sync_with_stdio(false);cin.tie(0);
    cin >> m >> n >> r >> c;
    for(int i = 1 ; i <= m ; i++) for(int j = 1 ; j <= n ; j++) cin >> a[i][j];
    int ans = solve();
    cout << ans;
    return 0;
}
```