Submission

ID	DATE	PROBLEM	STATUS	CPU	LANG
	TEST CASES				
8193836	20:02:30	Fire	✔ **Accepted**	1.69 s	C++
	✔✔				

Submission contains 1 file: download zip archive

FILENAME	FILESIZE	SHA-1 SUM	
fire2.cpp	3469 bytes	2a34237afef2c462efc6ce16e9d75f3f364b5ecc	download

Edit and resubmit this submission.

fire2.cpp

```
1   //https://open.kattis.com/problems/fire2
2   #include <bits/stdc++.h>
3   class Fire{
4     int n, m;
5     public:
6     bool inrangeguys(int x, int y) {
7         return x >= 0 && x <= n+1 && y >= 0 && y <= m+1;
8     }
9
10    bool inrangefire(int x, int y) {
11        return x >= 1 && x <= n && y >= 1 && y <= m;
12    }
13
14    void bfs(std::vector<std::vector<char>>& v, std::queue<std::pair<int,int>>& oldq,
    std::queue<std::pair<int,int>>& newq, bool fire) {
15        std::vector<std::pair<int,int>> movement = {{1,0},{-1,0},{0,1},{0,-1}};
16        while(!oldq.empty()) {
17            int currx = oldq.front().first;
18            int curry = oldq.front().second;
19            oldq.pop();
20            for(auto i : movement) {
21                int nextx = currx + i.first;
22                int nexty = curry + i.second;
23                if(fire) {
24                    if(inrangefire(nextx,nexty)) {
25                        if(v[nextx][nexty] == '.' || v[nextx][nexty] == '@') {
26                            v[nextx][nexty] = '*';
27                            newq.push({nextx,nexty});
28                        }
29                    }
30                }
31                else {
32                    if(inrangeguys(nextx,nexty)) {
```

Help

```
 33                     if(v[nextx][nexty] == '.') {
 34                         v[nextx][nexty] = '@';
 35                         newq.push({nextx,nexty});
 36                     }
 37                 }
 38             }
 39         }
 40     }
 41  }

 43  void checkguys(std::vector<std::vector<char>>& v, std::queue<std::pair<int,int>>& guys)
    {
 44      std::queue<std::pair<int,int>> newguys;
 45      while(!guys.empty()) {
 46          std::pair<int,int> i = guys.front();
 47          guys.pop();
 48          if(v[i.first][i.second] != '*') {
 49              newguys.push(i);
 50          }
 51      }
 52      guys = newguys;
 53  }

 55  void solve() {
 56      std::cin >> m >> n;

 58      std::vector<std::vector<char>> v;
 59      v.resize(n+2, std::vector<char>(m+2, '.'));

 61      for(int i = 1; i <= n; i++) {
 62          for(int j = 1; j <= m; j++) {
 63              std::cin >> v[i][j];
 64          }
 65      }

 67      std::queue<std::pair<int,int>> guys;
 68      std::queue<std::pair<int,int>> fire;

 70      for(int i = 0; i < n+2; i++) {
 71          for(int j = 0; j < m+2; j++) {
 72              if(v[i][j] == '@') {
 73                  guys.push({i,j});
 74              }
 75              if(v[i][j] == '*') {
 76                  fire.push({i,j});
 77              }
 78          }
 79      }

 81      int time = 0;
 82      bool works = false;
 83      while(!fire.empty() || !guys.empty()) {
 84          std::queue<std::pair<int,int>> newguys;
 85          std::queue<std::pair<int,int>> newfire;

 87          bfs(v, guys, newguys, false);
 88          bfs(v, fire, newfire, true);

 90          for(int i = 0; i < n+2; i++) {
 91              if(v[i][0] == '@' || v[i][m+1] == '@') works = true;
 92          }
 93          for(int j = 0; j < m+2; j++) {
 94              if(v[0][j] == '@' || v[n+1][j] == '@') works = true;
 95          }

 97          checkguys(v, newguys);

 99          guys = newguys;
100          fire = newfire;
101          time++;
```

```
102
103            if(works) break;
104        }
105
106        if(works) {
107            std::cout << time << std::endl;
108        }
109        else {
110            std::cout << "IMPOSSIBLE" << std::endl;
111        }
112    }
113
114 };
115
116 int main(){
117   std::ios_base::sync_with_stdio(false);
118   std::cin.tie(NULL);
119   Fire F1= Fire();
120   int cases;
121   std::cin >> cases;
122   while(cases--) {
123     F1.solve();
124   }
125   return 0;
126 }
```

Help