Problem

Becca and Terry are microbiologists who have a friendly rivalry. When they need a break from their research, they li e to play a game together. The game is played on a matrix of unit cells with R rows and C columns. Initially, each ce l is either empty, or contains radioactive material.

On each player's turn, if there are no empty cells in the matrix, that player loses the game. Otherwise, they choose an empty cell and place a colony of bacteria there. Bacteria colonies come in two types: H (for "horizontal") and V (for vertical").

When a type H colony is placed into an empty cell, it occupies that cell (making it non-empty), and also tries to spre d into the cell immediately to the west (if there is one) and the cell immediately to the east (if there is one).
When a type V colony is placed into an empty cell, it occupies that cell (making it non-empty), and also tries to spre d into the cell immediately to the south (if there is one) and the cell immediately to the north (if there is one).
Whenever a colony (of either type) tries to spread into a cell:

If the cell contains radioactive material, the colony mutates and the player who placed the colony loses the game.
If that cell is empty, the colony occupies that cell (making it non-empty), and then the rule above is triggered again (.e. the colony will try to spread further).
If the cell already contains bacteria (of any type), the colony does not spread into that cell.
Notice that it may be possible that all of a player's available moves would cause them to lose the game, and so they a e doomed. See the sample case explanations below for examples of how the game works.

Becca makes the first move, and then the two players alternate moves until one of them loses the game. If both playe s play optimally, who will win? And, if Becca will win, how many distinct winning opening moves does she have? (wo opening moves are distinct if and only if they either use different cells, or different kinds of colony, or both.)

Input

The first line of the input gives the number of test cases, T. T test cases follow. Each case begins with one line conta ning two integers R and C: the number of rows and columns, respectively, in the matrix. Then, there are R more row of C characters each. The j-th character on the i-th of these lines represents the j-th column of the i-th row of the ma rix. Each character is either . (an empty cell) or # (a cell with radioactive material).

Output

For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1), and y is a integer: either 0 if Becca will not win, or, if Becca will win, the number of distinct winning opening moves she can ake, as described above.

Limits

Time limit: 30 seconds per test set.
Memory limit: 1GB.
$1 \leq T \leq 100$.

Test set 1 (Visible)

$1 \leq R \leq 4$.
$1 \leq C \leq 4$.

Test set 2 (Hidden)

$1 \leq R \leq 15$.
$1 \leq C \leq 15$.

Sample

Input

Output

```
5
2 2
..
.#
4 4
.#..
..#.
#...
...#
3 4
#.##
....
#.##
1 1
.
1 2
##
```

```
Case #1: 0
Case #2: 0
Case #3: 7
Case #4: 2
Case #5: 0
```

In Sample Case #1, Becca cannot place an H colony in the southwest empty cell or a V colony in the northeast empt cell, because those would spread onto a radioactive cell and Becca would lose. She has only two possible strategies hat do not cause her to lose immediately:

Place an H colony in the northwest or northeast empty cells. The colony will also spread to the other of those two cel s.
Place a V colony in the northwest or southwest empty cell. The colony will also spread to the other of those two cell .
If Becca chooses strategy 1, Terry can place a V colony in the southwest empty cell. If Becca chooses strategy 2, Te ry can place an H colony in the northeast empty cell. Either way, Becca has no empty cells to choose from on her ne t turn, so she loses and Terry wins.

In Sample Case #2, any of Becca's opening moves would cause a mutation.

In Sample Case #3, five of Becca's possible opening moves would cause a mutation, but the other seven are winning She can place an H colony in any of the cells of the second row, or she can place a V colony in any of the cells of th second column. In either case, she leaves two disconnected sets of 1 or 2 cells each. In each of those sets, only one t pe of colony can be played, and playing it consumes all of the empty cells in that set. So, whichever of those sets Ter y chooses to consume, Becca can consume the other, leaving Terry with no moves.

In Sample Case #4, both of Becca's two distinct possible opening moves are winning.

In Sample Case #5, Becca has no possible opening moves.

Solution:

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <set>
#include <cstring>

using namespace std;
static int grundy[16][16][16][16];

int solve(
 int r0, int c0, int r1, int c1,
 const vector<string>& field)
{
 if(r0 == r1 || c0 == c1){ return 0; }
 if(grundy[r0][c0][r1][c1] >= 0){ return grundy[r0][c0][r1][c1]; }
 set<int> st;
 // Vertical
 for(int c = c0; c < c1; ++c){
  bool accept = true;
  for(int r = r0; r < r1; ++r){
   if(field[r][c] == '#'){ accept = false; }
  }
  if(!accept){ continue; }
  st.insert(solve(r0, c0, r1, c, field) ^ solve(r0, c + 1, r1, c1, field));
 }
 // Horizontal
 for(int r = r0; r < r1; ++r){
  bool accept = true;
  for(int c = c0; c < c1; ++c){
   if(field[r][c] == '#'){ accept = false; }
  }
  if(!accept){ continue; }
  st.insert(solve(r0, c0, r, c1, field) ^ solve(r + 1, c0, r1, c1, field));
 }
 // Grundy
 int g = 0;
 for(const auto& x : st){
  if(g < x){ break; }
  ++g;
 }
 grundy[r0][c0][r1][c1] = g;
 return g;
}

int main(){
```

```
 ios_base::sync_with_stdio(false);
 int num_cases;
 cin >> num_cases;
 for(int case_num = 1; case_num <= num_cases; ++case_num){
  int h, w;
  cin >> h >> w;
  vector<string> field(h);
  for(int i = 0; i < h; ++i){ cin >> field[i]; }
  memset(grundy, -1, sizeof(grundy));
  const int g = solve(0, 0, h, w, field);
  int answer = 0;
  if(g > 0){
   // Vertical
   for(int c = 0; c < w; ++c){
    bool accept = true;
    for(int r = 0; r < h; ++r){
     if(field[r][c] == '#'){ accept = false; }
    }
    if(!accept){ continue; }
    if((solve(0, 0, h, c, field) ^ solve(0, c + 1, h, w, field)) == 0){
     answer += h;
    }
   }
   // Horizontal
   for(int r = 0; r < h; ++r){
    bool accept = true;
    for(int c = 0; c < w; ++c){
     if(field[r][c] == '#'){ accept = false; }
    }
    if(!accept){ continue; }
    if((solve(0, 0, r, w, field) ^ solve(r + 1, 0, h, w, field)) == 0){
     answer += w;
    }
   }
  }
  cout << "Case #" << case_num << ": " << answer << endl;
 }
 return 0;
}
```