(Analysis by Benjamin Qi)

Suppose that sweet corn grows in $(1, 1)$. Consider the minimum j such that alfalfa grows in $(1, j)$.

- Sweet corn grows in $(1, y)$ if $y < j$ and alfalfa grows in $(1, y)$ otherwise.
- Every square (x, y) satisfying $y < j$ contains either a sweet corn sprinkler or no sprinkler.
- There must be a sweet corn sprinkler at $(1, j - 1)$.

Now,

- If $j = N + 1$ then sweet corn grows in every square.
- Otherwise, run the solution recursively on the remaining $N \times (N + 1 - j)$ subrectangle; namely, those squares (x, y) such that $y \geq j$. Find the minimum k such that sweet corn grows in (k, j), and continue in a similar fashion.

In general, an assignment of sweet corn or alfalfa to each square corresponds to a down-right path from $(1, 1)$ to some square (x, y) that satisfies $x = N + 1$ or $y = N + 1$. In the above example, the first three squares of the path are $(1, 1) \to (1, j) \to (k, j)$. The squares that are just before where the path changes direction (such as $(1, j - 1)$) must contain a sprinkler of a certain type (so their states are fixed), while every other square that does not contain a cow can be in one of two states: either place no sprinkler or place a sprinkler of the same type as the crop that grows in that square. A path that changes direction d times fixes the states of $d + 1$ squares, so the states of the remaining squares can be assigned in $2^{(\#\text{ unoccupied squares})-d-1}$ ways. It suffices to sum 2^{-d-1} over all paths and then multiply the answer by $2^{(\#\text{ unoccupied squares})}$ at the end. In the code below, $p \equiv 2^{-1} \pmod{10^9 + 7}$.

We can do this naively in $O(N^3)$ and use prefix sums to get $O(N^2)$. It is probably easier to write the $O(N^3)$ solution first and then figure out how to optimize it.

Dhruv Rohatgi's code:

```
#include <iostream>
#include <algorithm>
#include <cstdio>
using namespace std;
#define MOD 1000000007

int N;
long long p = 500000004LL;
char A[2005][2005];
char B[2005][2005];
int r[2005][2005];
int b[2005][2005];
int psr[2005][2005];
int psb[2005][2005];

int main()
{
        freopen("sprinklers2.in","r",stdin);
        freopen("sprinklers2.out","w",stdout);
        cin >> N;
```

```
        for(int i=0;i<N;i++)
                cin >> (A[i+1]+1);
        for(int i=2;i<=N+1;i++)
                if(A[i-1][1] == '.')
                        b[i][0] = psb[i][0] = p;
        for(int j=1;j<=N;j++)
                if(A[1][j] == '.')
                        r[1][j] = psr[1][j] = p;
        for(int i=2;i<=N+1;i++)
                for(int j=1;j<=N;j++)
                {
                        if(A[i][j] == '.')
                        {
                                r[i][j] = (p*psb[i][j-1])%MOD;
                        }
                        if(A[i-1][j+1] == '.')
                        {
                                b[i][j] = (p*psr[i-1][j])%MOD;
                        }
                        psr[i][j] = (psr[i-1][j] + r[i][j])%MOD;
                        psb[i][j] = (psb[i][j-1] + b[i][j])%MOD;
                }
        int ans = (psr[N][N] + psb[N+1][N])%MOD;
        for(int i=1;i<=N;i++)
                for(int j=1;j<=N;j++)
                        if(A[i][j]=='.')
                                ans = (2LL*ans)%MOD;
        cout << ans << '\n';
}
```