SCV/2021

마법사 상어와 비바라기

- Implementation, Simulation -

안태진(taejin7824@gmail.com)
GitHub(github.com/taejin1221)
상명대학교 소프트웨어학과
201821002

Contents

- Problem
- Approach
- Code

Problem

- 문제 설명
  - 격자가 있음
    - 격자 안의 숫자는 물의 양
  - 구름이 특정 방향으로 움직임
    - 끝에 도달하면 다음으로 넘어옴

<table>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>2</td></tr>
<tr><td>2</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>4</td><td>3</td><td>2</td><td>9</td><td>0</td></tr>
<tr><td colspan="2" rowspan="2">구름임</td><td>2</td><td>9</td><td>0</td></tr>
<tr><td>2</td><td>1</td><td>0</td></tr>
</table>

Problem

- 문제 설명
  - 격자가 있음
    - 격자 안의 숫자는 물의 양
  - 구름이 특정 방향으로 움직임
    - 끝에 도달하면 다음으로 넘어옴
  - 비를 내림
    - 내린 칸의 물의 양 1 증가
    - 구름이 사라짐
  - 물 복사 버그
    - 대각선에 물이 있는 격자 만큼 물의 양 증가

Problem

- 문제 설명
  - 격자가 있음
    - 격자 안의 숫자는 물의 양
  - 구름이 특정 방향으로 움직임
    - 끝에 도달하면 다음으로 넘어옴
  - 비를 내림
    - 내린 칸의 물의 양 1 증가
    - 구름이 사라짐
  - 물 복사 버그
    - 대각선에 물이 있는 격자 만큼 물의 양 증가
  - 구름이 사라진 칸이 아닌 격자 중 물의 양이 2 이상
    - 구름이 생김, 물의 양이 2 감소
    - 반복

0	0	1	0	구름임
구름임	구름임	구름임	1	0
구름임	구름임	구름임	구름임	0
1	0	7	12	0
구름임	구름임	4	3	0

Problem

- 문제 설명
  - 총 m번 움직이고 났을 때의 남아있는 물의 양의 합!

```
5 4
0 0 1 0 2
2 3 2 1 0
4 3 2 9 0
1 0 2 9 0
8 8 2 1 0
1 3
3 4
8 1
4 8
```

n (격자의 크기), m (구름의 움직임 횟수)

격자의 물의 양

dir, move (dir방향으로 move번 움직이기)

```
77
```

Problem

- 문제 설명
  - 그렇다!
  - 삼성에서 주기적으로 개최하는 역량 테스트 문제 중 하나
  - 이런 식으로 단순 구현 문제가 많이 나옴
  - 어려운 것은 아니지만, 시간이 오래걸리고,
충분한 훈련이 되어있지 않으면 자잘한 오류 많음!
  - 그러니 연습!

Contents

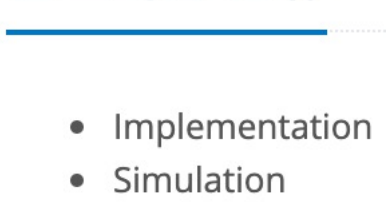
- Problem
- Approach
- Code

Approach

- 풀이
  - 알고리즘: 구현, 시뮬레이션

  - 꼼수 부리지 말고 그냥 구현하자
    - 쉽게 짤 수 있는 방법은 없다 그냥 구현하자

  - 일단 단계가 많으니 단계별로 기능을 구현
    1. 구름이 움직임
    2. 비가 내림
    3. 구름이 사라짐
    4. 물 복사
    5. 구름 생성



Approach

- 기능 (1/2)
  - 구름이 움직임 -> MoveClouds 함수
    - 말 그대로 구름을 움직이는 함수
  - 비가 내림 -> RainDance 함수
    - 비가 내린다 쏴아아아
    - 구름이 있는 칸에 1을 증가시키자
  - 구름이 사라짐 -> RemoveClouds 함수
    - 구름을 지우자
    - 굳이 만드는 이유?
      - 구름을 생성할 때 이전에 구름이 생성되었던 자리면 X
      - 따라서 이전 구름이 있던 자리를 저장 -> prevClouds
    - 하지만 물 복사를 위해선 구름을 지우는 것보단 복사를 한 뒤 지우는게 편함
      - 따라서 물 복사 다음에 지울 예정

Approach

- 기능 (2/2)
  - 물 복사-> WaterCopy 함수
    - 대각선들을 탐색 한 뒤 물의 양을 증가시키자
  - 구름 생성 -> GenerateClouds 함수
    - 물의 양이 2 이상인 곳에 구름을 생성하고, 물의 양을 2 줄이자!
    - 이때 이전에 물이 있던 자리면 생성 X!

Contents

- Problem
- Approach
- Code

Code

- module import

```
#include <iostream>

#include <vector> // vector class
#include <algorithm> // fill function
```

- define 및 상수들
  - 좌표를 저장 해야하므로 pair
    - {int, int}를 한 쌍으로 저장
  - 각 방향을 인덱스로 할 때 row와 col이 움직여야하는 거리
    - 물 복사 때에 대각선 확인은 2, 4, 6, 8!

```
typedef pair<int, int> pii;

const int rowChange[9] = { 0, 0, -1, -1, -1, 0, 1, 1, 1 };
const int colChange[9] = { 0, -1, -1, 0, 1, 1, 1, 0, -1 };
```

1부터 순서대로 ←, ↖, ↑, ↗, →, ↘, ↓, ↙

Code

- 전역 변수 및 Prototype
  - Global Variable
    - rowSize, colSize: 행의 크기, 열의 크기
    - grid: NxN 격자
    - clouds: 구름이 있는 칸의 좌표를 저장하는 vector(List, array)
    - prevClouds: [r][c]칸이 이전에 구름이 생성 되었는지를 저장하는 bool 행렬
  - 함수 Prototype
    - 위의 전역 변수를 이용하기 때문에 매개변수 필요 X
    - MoveClouds는 어느 방향으로 얼만큼 움직일 것을 알아야하기 때문에 그걸 매개변수로

```
int rowSize, colSize, grid[51][51];
vector<pii> clouds;
bool prevClouds[51][51];

void MoveClouds( int dir, int move );
void RainDance( );
void WaterCopy( );
void RemoveClouds( );
void GenerateClouds( );
```

Codes

- main
  - 값들 입력 받기
    - 입력이 많으니 빠른 입력
  - 구름의 초깃값 설정

```
int main(void) {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    int moveCommand;
    cin >> rowSize >> moveCommand;
    colSize = rowSize;

    for ( int i = 0; i < rowSize; i++ )
        for ( int j = 0; j < colSize; j++ )
            cin >> grid[i][j];
```

```
clouds.push_back( { rowSize - 2, 0 } );
clouds.push_back( { rowSize - 2, 1 } );
clouds.push_back( { rowSize - 1, 0 } );
clouds.push_back( { rowSize - 1, 1 } );
```

Code

- main
  - m번 반복할 때까지 process 반복

```
while ( moveCommand-- ) {
    int dir, move;
    cin >> dir >> move;
    MoveClouds( dir, move );
    RainDance( );
    WaterCopy( );
    RemoveClouds( );
    GenerateClouds( );
}
```

  - 답 출력
    - 그냥 세면 댐

```
int ans = 0;
for ( int i = 0; i < rowSize; i++ )
    for ( int j = 0; j < colSize; j++ )
        ans += grid[i][j];

cout << ans << '\n';
```

Code

- MoveClouds 함수
  - move번 dir 방향으로 구름을 움직이기

  - 하지만 넘어가면 원형 큐처럼 한바퀴 돌아야 함으로 최대 크기 더하고 나머지

```
void MoveClouds( int dir, int move ) {
    for ( int i = 0; i < move; i++ ) {
        for ( pii& clo : clouds ) {
            clo.first = (clo.first + rowChange[dir] + rowSize) % rowSize;
            clo.second = (clo.second + colChange[dir] + colSize) % colSize;
        }
    }
}
```

Code

- RainDance 함수
  - 구름이 있는 칸에 물을 증가 시키자!

```
void RainDance( ) {
    for ( pii& clo : clouds ) {
        grid[clo.first][clo.second]++;
    }
}
```

Code

- WaterCopy 함수
  - 구름이 있던 칸들 기준
  - 대각선들을 확인하며 물이 있는 칸의 개수를 센 뒤 격자에 복사해주기

```
void WaterCopy( ) {
    for ( pii& clo : clouds ) {
        int currRow = clo.first, currCol = clo.second;

        int copyPlus = 0;
        for ( int i = 2; i <= 8; i += 2 ) {
            int newRow = currRow + rowChange[i], newCol = currCol + colChange[i];
            if ( ( 0 <= newRow ) && ( newRow < rowSize ) && ( 0 <= newCol ) && ( newCol < colSize ) ) {
                if ( grid[newRow][newCol] )
                    copyPlus++;
            }
        }

        grid[currRow][currCol] += copyPlus;
    }
}
```

Code

- RemoveClouds 함수
  - prevClouds 함수 false로 초기화
    - 안해주면 이전이전 구름이 있던 자리까지 기억함 ㅠ
    - 구름이 있던 자리 체크

```
void RemoveClouds( ) {
    for ( int i = 0; i < rowSize; i++ )
        fill( prevClouds[i], prevClouds[i] + colSize, false );

    while ( clouds.size() ) {
        pii clo = clouds.back( ); clouds.pop_back( );
        prevClouds[clo.first][clo.second] = true;
    }
}
```

Code

- GenerateClouds 함수
  - 물의 양이 2 이상이고, 이전에 구름이 있던 자리가 아니면
    - 구름 생성, 물의 양 2 감소!

```
void GenerateClouds( ) {
    for ( int i = 0; i < rowSize; i++ ) {
        for ( int j = 0; j < colSize; j++ ) {
            if ( grid[i][j] >= 2 && !prevClouds[i][j] ) {
                clouds.push_back( { i, j } );
                grid[i][j] -= 2;
            }
        }
    }
}
```

감사합니다!