SIT/2021

빙산

- Implementation, Graph, DFS -

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

Contents

- Problem
- Approach
- Code

Problem

- 문제 설명
  - 너무 길어... 다 알잖아?

빙산 성공 출처 분류

Gold IV

Breadth-first search | Depth-first search | Graph theory | Graph traversal | Implementation

난이도 제공: solved.ac — 난이도 투표하러 가기

  - 이런 문제 특징
    - 태그가 김
    - 코드도 김
    - 오류 완전 많이 남
    - 귀찮음

    - 삼성 SW 역량 테스트에 많이 나오는 형식
    - 근데 이건 KOI네 ㅎㅎ

Olympiad > 한국정보올림피아드 > KOI 2006 > 초등부 2번

문제

지구 온난화로 인하여 북극의 빙산이 녹고 있다. 빙산을 그림 1과 같이 2차원 배열에 표시한다고 하자. 빙산의 각 부분별 높이 정보는 배열의 각 칸에 양의 정수로 저장된다. 빙산 이외의 바다에 해당되는 칸에는 0이 저장된다. 그림 1에서 빈칸은 모두 0으로 채워져 있다고 생각한다.

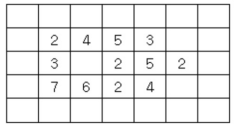


그림 1. 행의 개수가 5이고 열의 개수가 7인 2차원 배열에 저장된 빙산의 높이 정보

빙산의 높이는 바닷물에 많이 접해있는 부분에서 더 빨리 줄어들기 때문에, 배열에서 빙산의 각 부분에 해당되는 칸에 있는 높이는 일년마다 그 칸에 동서남북 네 방향으로 붙어있는 0이 저장된 칸의 개수만큼 줄어든다. 단, 각 칸에 저장된 높이는 0보다 더 줄어들지 않는다. 바닷물은 호수처럼 빙산에 둘러싸여 있을 수도 있다. 따라서 그림 1의 빙산은 일년후에 그림 2와 같이 변형된다.

그림 3은 그림 1의 빙산이 2년 후에 변한 모습을 보여준다. 2차원 배열에서 동서남북 방향으로 붙어있는 칸들은 서로 연결되어 있다고 말한다. 따라서 그림 2의 빙산은 한 덩어리이지만, 그림 3의 빙산은 세 덩어리로 분리되어 있다.

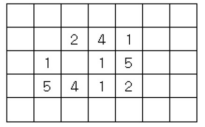


그림 2

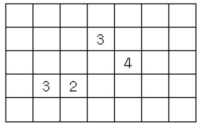


그림 3

한 덩어리의 빙산이 주어질 때, 이 빙산이 두 덩어리 이상으로 분리되는 최초의 시간(년)을 구하는 프로그램을 작성하시오. 그림 1의 빙산에 대해서는 2가 답이다. 만일 전부 다 녹을 때까지 두 덩어리 이상으로 분리되지 않으면 프로그램은 0을 출력한다.

Contents

- Problem
- Approach
- Code

Approach

- 풀이
  - 사실 직관적으로 일일이 다 돌려가면서 찾으면 됨
  - 총 노드의 개수가 총 $300 \times 300 = 90{,}000$이므로 Bruteforcing 가능

- Pseudo Code
  1. 빙산이 전부 녹을 때까지
     1. 섬의 개수가 2개 이상이면
        1. 끝
     2. 빙산을 녹이자

Contents

- Problem
- Approach
- Code

Code

- Global
  - 기본적으로 3개의 함수로 구성
    - dfs: 하나의 빙산 탐색
    - CountIceberg: 총 빙산의 갯수 세기
    - Melt: 빙산 녹이기

```
void dfs( int row, int col ) {…
}

int CountIceberg( ) {…
}

int Melt( ) {…
}
```

  - 이외에 여러 함수들이 공유할 전역 변수 선언
    - maxRow, maxCol: n, m의 크기
    - map: 빙산 지도
    - visited: 방문 정보
      - 쓸 때마다 초기화

```
int maxRow, maxCol;
int map[300][300];
bool visited[300][300];
```

Code

- main function (1/2)
  - Input
    - 빠른 I/O로 입력 받기
    - 입력 받으면서 !0의 개수 세기
  - 문제 풀기
    - 모든 빙산이 녹을 때 까지
      - 빙산의 개수가 2개 이상이면
        - 답 저장하고 그만두기
      - 빙산 녹이기
        - 녹은 후의 cell 개수 저장
      - 시간 증가

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

    cin >> maxRow >> maxCol;

    int nonZero = 0;
    for ( int i = 0; i < maxRow; i++ ) {
        for ( int j = 0; j < maxCol; j++ ) {
            cin >> map[i][j];
            if ( !map[i][j] )
                nonZero++;
        }
    }
```

```
    int time = 0, ans = 0;
    while ( nonZero ) {
        if ( CountIceberg( ) >= 2 ) {
            ans = time;
            break;
        }

        nonZero = Melt( );
        time++;
    }
```

Code

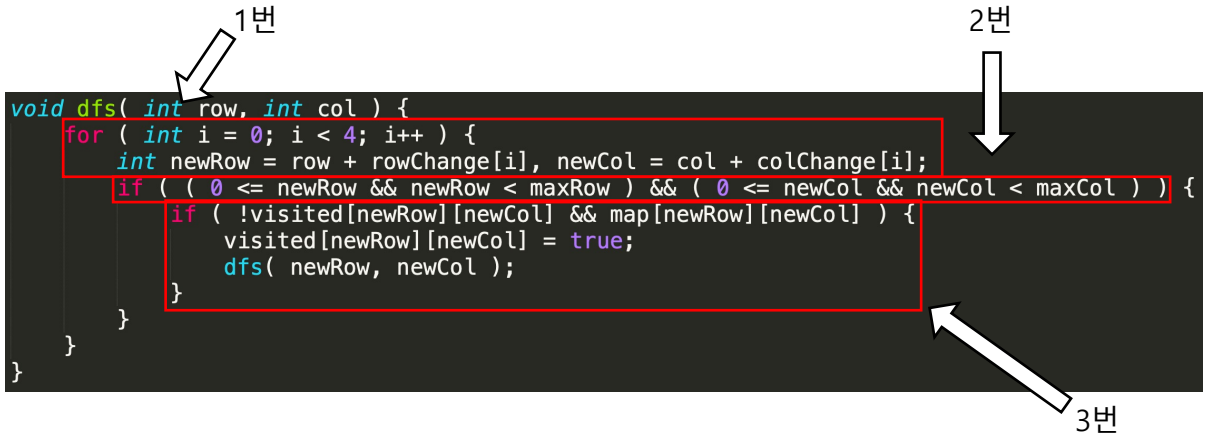
- main function (2/2)
  - 답 출력
    - while 문을 탈출 하는 경우
      1. 빙산이 두개 이상으로 나눠지는 경우
         - ans에 답을 저장했을 것임
      2. 빙산이 다 녹았을 경우
         - ans는 초기값일 것임
    - 따라서 그냥 답 출력

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

Code

- dfs function
  - 빙산의 시작점이 주어지면 모든 빙산 탐색

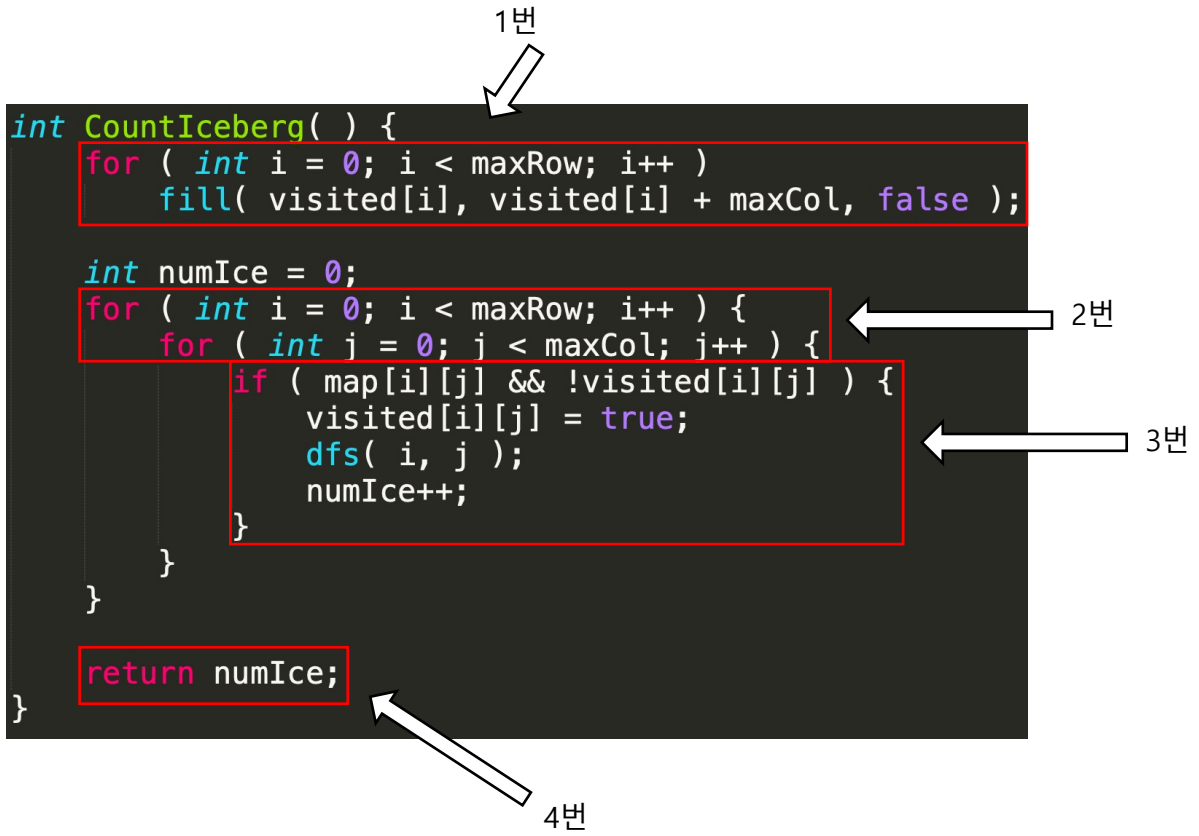
1. 동서남북에 빙산이 있는지 체크
   - newRow, newCol: 동서남북의 row, col
2. indexOutOfRange 나지 않게 범위 체크!
3. 만약 방문하지 않았고, 빙산이라면(!0)
   - 방문!



Code

- CountIceberg function
  - 빙산의 개수 세는 함수

1. visited 배열 초기화
2. 모든 노드 방문하면서
3. 빙산이면서 방문 안했으면
   - dfs로 모든 빙산 탐색
   - 빙산 개수++
4. 빙산 개수 return



Code

- Melt function
  - 모든 노드 방문하면서 주변의 바다만큼 녹이기
    - temp: 녹은 후의 지도 상태

1. 모든 노드 방문하면서
2. 주변 바다 개수 세기
3. 빙산 녹이기 및 0이하 0으로
4. 녹은 후 지도 상태를
원래 지도에 복사

1번 2번

```
int Melt( ) {
    int temp[300][300], nonZero = 0;
    for ( int row = 0; row < maxRow; row++ ) {
        for ( int col = 0; col < maxCol; col++ ) {
            temp[row][col] = map[row][col];
            if ( map[row][col] ) {
                int ocean = 0;
                for ( int i = 0; i < 4; i++ ) {
                    int newRow = row + rowChange[i], newCol = col + colChange[i];
                    if ( ( 0 <= newRow && newRow < maxRow ) && ( 0 <= newCol && newCol < maxCol ) ) {
                        if ( !map[newRow][newCol] )
                            ocean++;
                    }
                }

                temp[row][col] = map[row][col] - ocean;
            }

            if ( temp[row][col] <= 0 )
                temp[row][col] = 0;
            else
                nonZero++;
        }
    }

    for ( int i = 0; i < maxRow; i++ )
        for ( int j = 0; j < maxCol; j++ )
            map[i][j] = temp[i][j];

    return nonZero;
}
```

3번

4번

감사합니다!