〈 GBC_ Algorithm PA 5 〉

27 기 최 하영

1. Fact

- 한 덩어리의 빙산이 주어질 때, 이 빙산이 두 덩어리 이상으로 분리되는 최초의 시간(년)을 구하는 프로그램 작성하기

- 각 부분에 해당하는 높이는 동서남북 네 방향에 0이 저장된 칸의 개수만큼 줄어들며, 0보다 더 줄어들지는 않음.

- 입력 : (1번째 줄) 이차원 배열의 행의 개수 N, 열의 개수 M ($3 \leq n, m \leq 300$)
  (2~n 번째 줄) 각 줄마다 배열의 각 행을 나타내는 M 개의 정수
  (단, 각 칸에 들어가는 값은 0 이상 10 이하)

- 출력 : 빙산이 분리되는 최초의 시간(년)
  만약, 빙산이 다 녹을 때까지 분리되지 않으면 0 을 출력함

2. Overviews

1) 빙산의 분리 여부를 while 문에 넣어 반복
2) 분리 여부 → DFS 로 판별 (BFS 도 가능)
3) 분리 안 됐을 때, 빙산을 녹여야 함
   분리 됐을 때, 연도 출력 후 종료
4) 바로 녹이면 다음 칸에 영향 주기 때문에 임시 배열 만들어서 0 의 개수 count
5) 저장된 count 만큼 원래 빙산에서 빼 줌
6) 반복문 계속 돌려서 분리 안되면 0, 분리 되면 몇 년인지 출력

3. Algorithm

1) 이차원 배열의 행의 개수 n, 열의 개수 m ($3 \leq n, m \leq 300$) 입력 받기
2) while(빙산이 0 개 || 빙산이 2 개) → DFS 로 빙산의 개수를 판별
   i) DFS 가 0 번 돌면 전체가 0 이므로, 종료
   ii) DFS 가 1 번 돌면 전체 연결되어 있으므로, year++ 해주고 다시 녹이기
   iii) DFS 가 2 번 돌면 조건 만족하므로, year 출력

3) 빙산이 2 개로 분리되지 않았을 때, 녹이는 방법

→ DFS 를 돌 때, 각 칸의 상하좌우를 탐색하여 0 일 경우 counting 해준다.

(+) 녹일 때, 탐색해도 되지만 그러면 DFS 에서 했던 상하좌우 탐색을 똑같이 반복해야 하므로, DFS 안에서 탐색하고 나서 바로 구해주는 것이 효율적인 것 같다.

해당 인덱스의 Counting 한 값을 임시 배열에 저장

Ex)

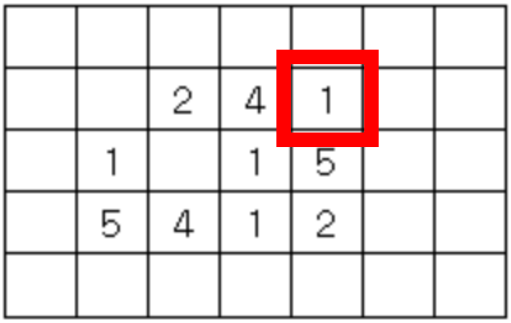

그림 2

1 은 상하좌우에 0 이 2 개이므로 tmp[1][4] = 2 와 같이 저장하여

원래 iceberg[1][4] = 1 에서 빼줘야 할 값을 만들어 놓는다

➔ iceberg = iceberg – tmp 라는 식 도출 가능함

4. Time complexity

O(n^2) → 이중 루프에서 입력자료 처리

```
for(int i = 0; i < n; i++){
    for(int j = 0; j < m; j++)
        cin >> iceberg[i][j];
}
```