[22.07.02] 세균번식

TS세균번식

김 태 현

문 제

- 정사각형 모양의 접시에 특수 약품을 떨어뜨려 세균을 생성, 번식, 소멸 시킨다.
- 접시는 (1,1) ~ (N,N) 좌표 값의 셀을 가지며, 각 셀은 에너지 소비량이 존재
  에너지 소비량: 해당 셀에서 세균이 생성될 때 필요한 번식 에너지 소비량
- 생성 약품 떨어뜨리는 경우 : (R,C),타겟 세균, 번식에너지
  1. 처음
     (R,C)가 비어 있는 경우 : 타겟 세균과 동인한 종류의 세균 생성, 활성화
     타겟 세균과 같은 종류의 세균 존재 : 활성화
     다른 세균 존재 : 증료
  2. 번식에너지가 남아 있고 번식 가능한 위치가 있는 동안 반복
     1) 상하좌우로 연결된 같은 종류 세균 모두 활성화
     2) 활성화된 모든 세균의 상하좌우가 비어 있는 곳 중에 가장 우선순위 높은 위치에 세균 생성, 활성화
        우선순위) 1. 에너지 소비량 높은 순 2. 행 번호 작은 순 3. 열 번호 작은 순
- 소멸 약품 떨어뜨리는 경우 : (R,C)
  : (R,C) 위치에 상하좌우로 연결된 같은 종류의 세균을 모두 소멸

void init(int N, int Dish[][])

N: 접시 크기 (5 ~100)
Dish[i][j]: (i+1, j+1) 위치의 에너지 소비량 (1 ~ 5)

int dropMedicine(int targetType, int R, int C, int totalEnergy)

(R,C) 위치에 targetType(1~2)을 타겟으로 하는 생성 약품을 떨어뜨림
번식 에너지는 totalEnergy(<=200)
return: targetType 인 세균 개수

int cleanBacteria(int R, int C)

(R,C) 위치에 소멸 약품을 떨어뜨림
return: 소멸된 세균과 같은 종류의 남은 세균 총 개수
소멸된 세균이 없는 경우 -1

※ 제약사항

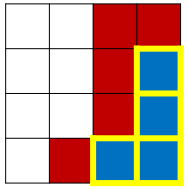
N <= 100
dropMedicine() <= 1,500
함수 호출시마다 활성화되는 세균 개수 <= 1,000
cleanBacteria() <= 500

dropMedicine()

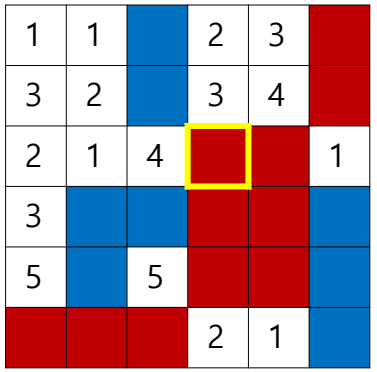
1. **시작**
   - (R,C) 가 빈 경우, 생성 후 활성화
   - (R,C) 가 타겟 세균인 경우, 활성화
   - (R,C) 가 다른 세균인 경우, 종료

2. **에너지 남은 동안 반복**
   1) 주변 활성화
      - flood fill : dfs or bfs
      - 그 과정에서 주변이 빈 경우
        번식 후보로 등록 : pq
      - visited check : 활성화 세균 & 번식 후보

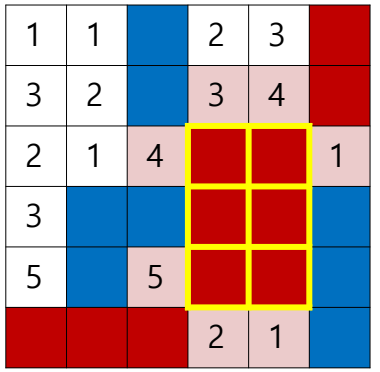
   2) 최우선순위 위치에서 번식
      - pq의 top 위치 선택하여 세균 생성, 활성화
      - 번식 가능한 위치 없는 경우 종료



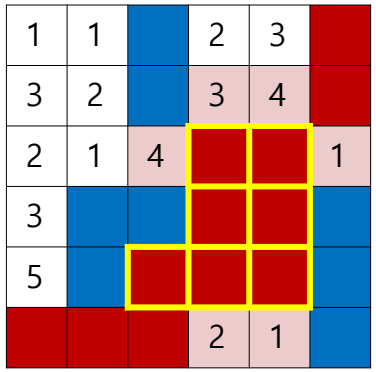
1) 시작, Energy=15



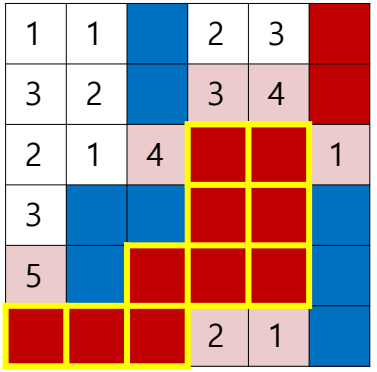
2) 활성화



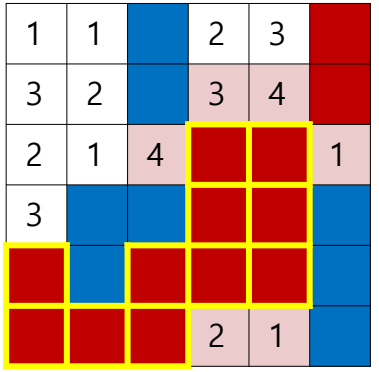
3) 번식, Energy=10



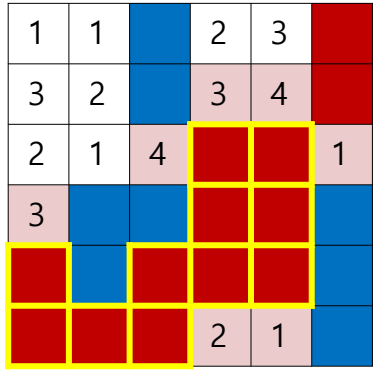
4) 활성화



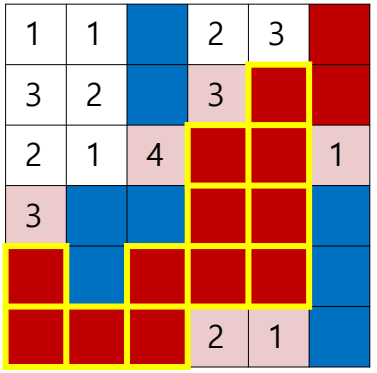
5) 번식, Energy=5



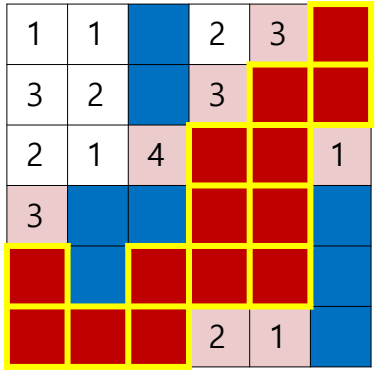
6) 활성화



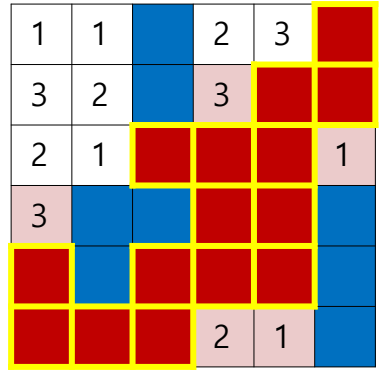
7) 번식, Energy=1



8) 활성화

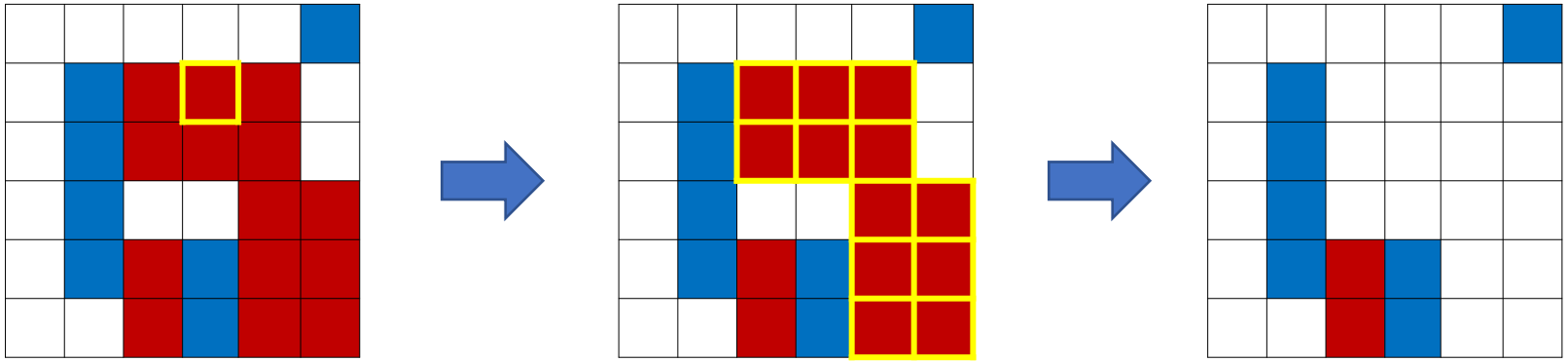


9) 번식, Energy=0



cleanBacteria()

단순 flood fill : dfs or bfs



DATA 관리

세균 정보

	1	2	3	4	5
1	1	1	0	0	1
2	0	0	0	0	1
3	1	1	1	1	1
4	0	2	2	2	2
5	1	0	2	1	1

세균 수

1	2
12	5

Queue bfs시

0	1	2	3	4
(r, c)				

소비 에너지

	1	2	3	4	5
1	3	3	5	1	1
2	4	2	4	2	2
3	1	1	1	3	2
4	3	2	2	5	2
5	1	5	2	1	1

PQ

1. 에너지 소비량 높은 순
2. R 작은 순
3. C 작은 순

고 찰

1. **상당히 오랜만에 나온 성능에 민감한 문제**

   동일한 설계여도 구현 방식에 따라 합격 여부 갈림

2. **성능 관련**

   - stl pq + stl queue 를 쓰면 시간초과 발생 할 수 있다.
   - stl set은 시간초과가 발생한다.
     일반적인 우선순위 문제처럼 우선순위 값을 관리해나가는게 아니라 매번 새롭게 등록하며,
     우선순위 기준 한 개에 최우선순위 값 한 개씩만 구해나가므로 set은 특히 더 비효율적이다.
   - sample code에서도 stl을 활용하듯이 의도적으로 stl을 불합격시키는 문제는 없다.
     다만, 성능에 민감한 문제의 경우 비효율적으로 사용하면 시간초과 발생 가능성은 있다.
   - pair 보단 custom type 이 빠르다.

3. **개인적인 견해**

   - pq는 반드시 non-stl을 해야 할 필요는 없지만 그 어떠한 변수도 없애려면 준비해도 좋다.
     lazy update vs real time update (indexed heap) 은 성능상 별 차이 없음..
   - queue는 구현하기 어렵지 않으므로 stl을 굳이 쓰지는 말자.
   - 이 문제는 해당되지 않지만 재귀를 활용한 DFS시, stack 깊이가 좀 더 깊어지면 stack overflow 발생 할 수 있다.
     stack 직접 구현한 DFS or BFS 권장


감사합니다