SAMSUNG SW AI ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성청년 SW·AI아카데미

SWEA5648 원자 소멸 시뮬레이션

SAMSUNG SW AI ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

원자 소멸 시뮬레이션

목표 : 원자들이 소멸되면서 방출하는 에너지의 총합 구하기

조건

1. 원자의 최초 위치는 2차원 평면상의 [x, y] 이다.
2. 원자는 **각자 고유의 움직이는 방향을 가지고 있다. (상하좌우 4방향)**
3. 모든 원자들의 **이동속도는 동일**하다. 즉, 1초에 1만큼의 거리를 이동한다.
4. 모든 원자들은 최초 위치에서 **동시에** 이동을 시작한다.
5. 두 개 이상의 원자가 동시에 충돌 할 경우 **충돌한 원자들은 모두 보유한 에너지를 방출하고 소멸**된다.

제약 사항

1. N : 원자들의 수 (1 ≤ N ≤ 1,000)
2. K : 각 원자들의 보유 에너지 (1 ≤ K ≤ 100)
3. x,y : 원자들의 위치 값 (-1,000≤x,y≤1,000)
4. 원자들은 2차원 평면 위에서 움직이며 원자들이 움직일 수 있는 좌표의 범위에 제한은 없다.
5. 원자들의 이동 방향은 상(0), 하(1), 좌(2), 우(3)로 주어진다. (방향 변경 X)
6. 원자들은 동시에 1초에 이동 방향으로 1만큼 이동한다.
7. 원자들의 최초 위치는 서로 중복되지 않는다.
8. 원자들은 2개 이상의 원자들이 서로 충돌할 경우 보유한 에너지를 방출하면서 바로 소멸된다.
9. 원자들이 충돌하여 소멸되며 방출되는 에너지는 다른 원자의 위치나 이동 방향에 영향을 주지 않는다.

문제 이해하기

그림과 같은 상황이고, 원자들의 에너지는 1이라 가정한다.

1초 뒤

- I, J 의 충돌 후 소멸 → 2 에너지 방출

1.5초 뒤

- A,B 원자가 충돌 수 소멸 → 2 에너지 방출

2초 뒤

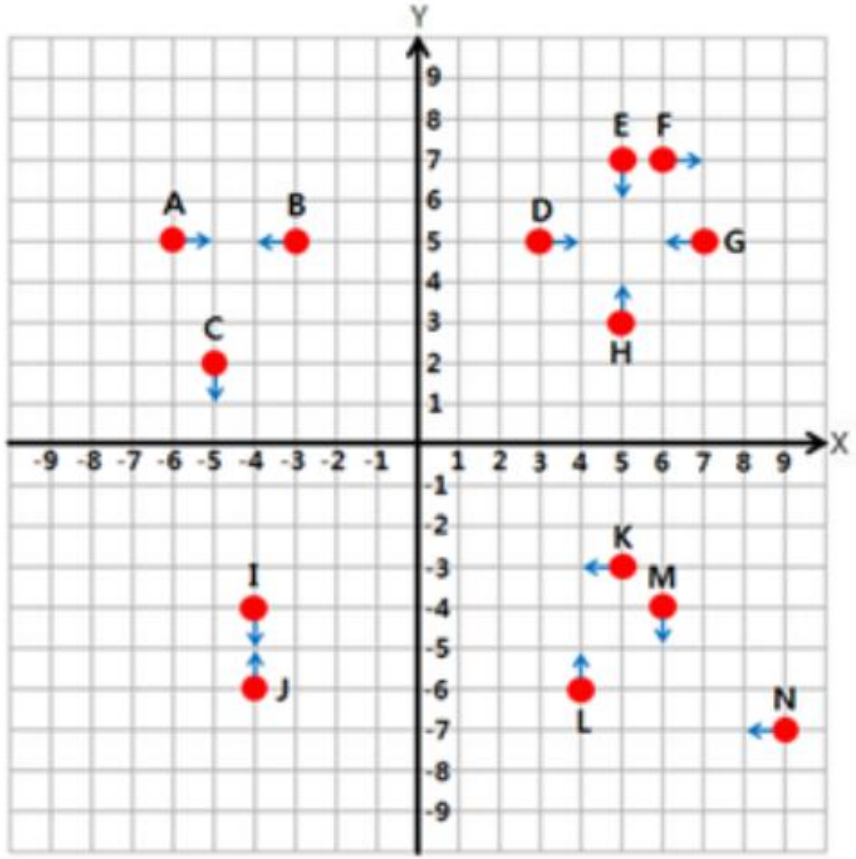
- D,E,G,H 충돌 후 소멸 → 4 에너지 방출

3초 뒤

- M,N 충돌 후 소멸 → 2 에너지 방출

방출된 에너지 총합 : 2 + 2 + 4 + 2 = 10

- C,F,K,L은 영원히 충돌하지 않고 남음



[그림-1]

접근 방법

1. 현재 원자의 위치 정보와 방향, energy 데이터가 들어오고, 시뮬레이션을 돌리면서 각각의 원자들의 상태를 추적해야 한다.

- **입력된 원자 데이터들을 관리할 자료구조가 필요하다.**
  - 구조체 사용
- **원자들의 개수 N 의 범위 : 1000**
  - vector로 관리

```
struct NODE {
    int x;
    int y;
    int dir;
    int energy;
};

vector<NODE> atom(1001);
```

데이터 입력 받기

N : 원자들의 수

x좌표, y좌표, 방향, 에너지 : 각 원자 데이터

```
for (int tc = 1; tc <= T; tc++) {
    cin >> N;
    for (int i = 0; i < N; i++) {
        int x, y, dir, energy;
        cin >> x >> y >> dir >> energy;
        atom.push_back({x,y,dir,energy});
    }

    int ans = 0;
    cout << "#" << tc << " " << ans << '\n';
```

https://gist.github.com/hoconoco/2aa1eb056883e436fd55e90a82dc5e8d

2. N 개의 원자들의 위치, 이동 방향, 보유 에너지가 주어질 때 원자들이 소멸되면서 방출하는 에너지의 총합을 구하기

1. 시뮬레이션 준비

- 원자를 표현할 구조체 구현 → 위치, 방향, 에너지 그리고 소멸 여부 (구조체 변수 생성)
- map[][] 구현 → map[y][x] 에는 해당 지점의 원자의 개수를 기록한다.

2. 시뮬레이션 시작

- 모든 원자가 소멸할 때까지 반복 (카운팅을 통해 N이 되거나 0이 될 때까지 반복)
- n개의 원자 이동
- 충돌 감지 및 소멸 처리

[참고] vector의 erase()

vector에 담긴 원자 정보를 .erase()로 소멸에 대응할 수 도 있지만,

반복될 경우 성능이 매우 저하될 수 있다 O(N^2)

→ 원자의 소멸이 있을 때마다 erase()로 제거하기 보다, 변수로 체크하는 것이 훨씬 시간 복잡도를 줄일 수 있다. 또한, 살아있는 원자 개수를 이용하는 것도 도움이 된다. O(1)

데이터 입력 시 선처리를 진행한다.

map[][] 를 만들고, 살아있는 원자 개수를 추적할 변수를 생성,
원자의 생존여부를 담당하는 구조체 변수를 생성한다.

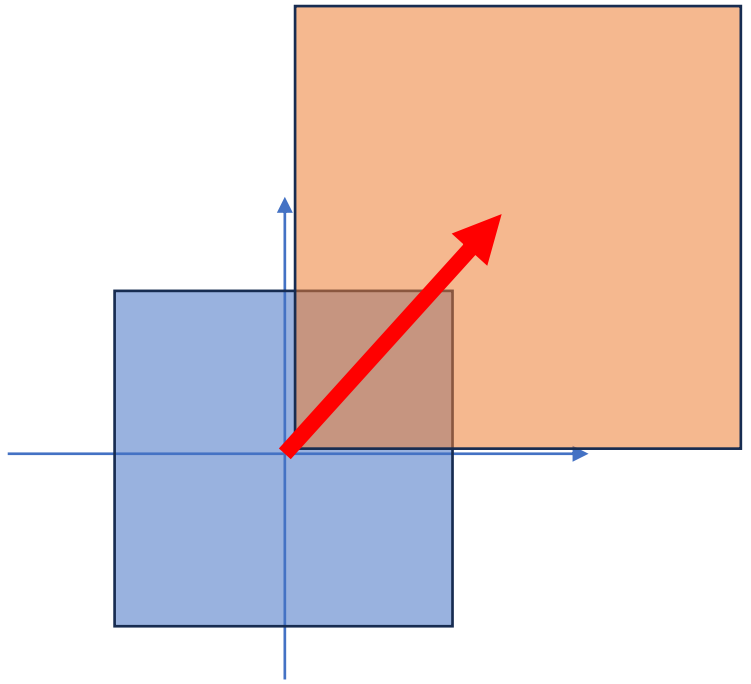
```
struct NODE {
    //좌표
    int x;
    int y;
    //방향
    int dir;
    //에너지
    int energy;
    //생존여부
    int isAlive;
};
```

```
//map 구현
int map[4001][4001];

//살아있는 원자 개수 체크
int cnt = 0;
```

```
//처음 입력 시 원자 생존 여부 : 1
atom.push_back({x,y,dir,energy, 1});

//살아있는 원자 개수 카운팅
cnt++;
//map에 원자 개수 등록
map[y][x] = 1;
```

map[][] 에 원자 개수 기록

- 이 문제는 2차원 맵 데이터가 들어오질 않고 직접 구현해야 한다.
- 원자가 같은 위치에 있는 지 체크용 map
- 원자의 좌표 범위 : 음수 → 양수 처리 (배열 인덱스 양수)
- 원자들이 이동 시 소수점 위치 충돌 가능성 → 2배 확장 (1.5초 뒤 충돌구현)
  - $(-1{,}000 \le x,y \le 1{,}000) \rightarrow (\ 0 \le x,y \le 4{,}000\ )$



좌표 정보 변환하기

음수 → 양수

x2 로 1.5초 충돌 방지

```
for (int i = 0; i < N; i++) {
    int x, y, dir, energy;
    cin >> x >> y >> dir >> energy;

    //음수 -> 양수, x2 로 1.5초 방지
    x = (x + 1000) * 2;
    y = (y + 1000) * 2;
```

시뮬레이션 구현하기 (힌트)

시뮬레이션은 모든 원자가 소멸할 때까지 반복

→ cnt 변수로 살아있는 원자 개수 추적 중

1. 모든 원자는 이동을 시작

2. 충돌 감지 및 원자 소멸

- 소멸 시 에너지 누적 : ans에 누적

```
int ans = 0;
void solve() {
    //cnt는 살아있는 원자 개수, 0이 될 때까지 반복
    while (cnt > 0) {

        //1. 모든 원자가 1회씩 이동을 시작
        for (int i = 0; i < atom.size(); i++) {

        }

        //2. 충돌 감지 및 소멸 처리
        for (int i = 0; i < atom.size(); i++) {

        }
    }
}
```

원자 이동 구현하기 (힌트)

1. 모든 원자는 이동을 시작

- **isAlive 변수로 이미 소멸한 원자는 넘기기**
- **다음 이동할 좌표 생성**
  - direction search 필요 상(0), 하(1), 좌(2), 우(3)
  - 범위 체크 : 벗어나면 죽음, 살아있는 원자 개수 감소, 해당 지역의 원자 개수 감소
- **다음 이동할 좌표의 원자 개수 증가**
- **기존 위치 원자 개수 : 0**
- **원자 위치 갱신**

원자 이동 구현하기 (힌트)

주석을 보고 연습하자

```
//1. 모든 원자가 1회씩 이동을 시작
for (int i = 0; i < atom.size(); i++) {
    //이동할 원자 뽑기
    //이미 죽었다면, 넘기기
    //이동할 좌표 ( direction search 필요 )
    //이동한다면, 범위를 체크해야겠지? -> 벗어나면 죽음
    if (                                ) {
        //해당 원자 생존 여부 : 0
        //살아있는 원자 개수 감소
        //해당 원자 있던 좌표의 원자 개수 : 0
        //넘어가야겠지?
        continue;
    }

    //새로운 위치의 원자 개수 증가
    //기존 위치 원자 개수 0
    //해당 원자 위치 갱신
}
```

2. 충돌 감지 및 원자 소멸

- **모든 원자의 위치를 조사**
- **map에 원자가 2개 이상 있다면, 충돌 발생**
  - 해당 위치의 원자 모두 소멸 처리
  - 같은 좌표 발견 시, 에너지 누적, 해당 원자 소멸 처리, 살아있는 원자 개수 감소

원자 충돌 구현하기 (힌트)

초기화도 빼면 안된다.

```
//2. 충돌 감지 및 소멸 처리D
for (int i = 0; i < atom.size(); i++) {
    //원자 뽑기
    //죽은 자는 말이 없다.

    //해당 원자의 위치에 원자 개수가 2이상이면, 충돌 발생
    if (     ) {

        //해당 위치의 원자들 전부 소멸 처리
            //죽은 자의 영혼은 넘긴다.

            //같은 좌표 발견 시
            if (    ) {
                //방출 에너지 누적
                //원자 소멸
                //개수 감소
            }
        }
    }
    //해당 위치 원자 제거
}
```

SAMSUNG SW AI ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

내일 방송에서 만나요!

삼성청년SW·AI아카데미