



**#3190. 뱀**

<https://www.acmicpc.net/problem/3190>

24.12.02



# Problem

<https://www.acmicpc.net/problem/3190>

시간: 1시간

복잡한 구현 문제. 문제의 로직을 어떻게 코드로 옮길 수 있을까 고민해보자!

백

성공 다국어

☆ 한국어 ▾

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	128 MB	82069	35383	23842	41.519%

## 문제

'Dummy' 라는 도스게임이 있다. 이 게임에는 뱀이 나와서 기어다니는데, 사과를 먹으면 뱀 길이가 늘어난다. 뱀이 이리저리 기어다니다가 벽 또는 자기자신의 몸과 부딪히면 게임이 끝난다.

게임은 NxN 정사각 보드위에서 진행되고, 몇몇 칸에는 사과가 놓여져 있다. 보드의 상하좌우 끝에 벽이 있다. 게임이 시작할때 뱀은 맨위 맨좌측에 위치하고 뱀의 길이는 1 이다. 뱀은 처음에 오른쪽을 향한다.

뱀은 매 초마다 이동을 하는데 다음과 같은 규칙을 따른다.

- 먼저 뱀은 몸길이를 늘려 머리를 다음칸에 위치시킨다.
- 만약 벽이나 자기자신의 몸과 부딪히면 게임이 끝난다.
- 만약 이동한 칸에 사과가 있다면, 그 칸에 있던 사과가 없어지고 꼬리는 움직이지 않는다.
- 만약 이동한 칸에 사과가 없다면, 몸길이를 줄여서 꼬리가 위치한 칸을 비워준다. 즉, 몸길이는 변하지 않는다.

사과의 위치와 뱀의 이동경로가 주어질 때 이 게임이 몇 초에 끝나는지 계산하라.



# Problem example

아래의 입력 예제 설명





```
6
3
3 4
2 5
5 3
3
3 D
15 L
17 D
```

spin

3
D

15
L

17
D

Time : 0







# Problem example

아래의 입력 예제 설명

```
6
3
3 4
2 5
5 3
3
3 D
15 L
17 D
```

spin	3	15	17
	D	L	D

Time : 1



# Problem example

아래의 입력 예제 설명





```
6
3
3 4
2 5
5 3
3
3 D
15 L
17 D
```

spin

3
D

15
L

17
D

Time : 2



# Problem example

아래의 입력 예제 설명

```
6
3
3 4
2 5
5 3
3
3 D
15 L
17 D
```

spin	3	15	17
	D	L	D

			→		
				🍏	
			🍏		
		🍏			

Time : 3

spin	15	17
	L	D

			↓		
				🍏	
			🍏		
		🍏			

Time : 3



# Problem example





아래의 입력 예제 설명

```
6
3
3 4
2 5
5 3
3
3 D
15 L
17 D
```

spin

15
L

17
D

Time : 4



# Problem example

아래의 입력 예제 설명

```
6
3
3 4
2 5
5 3
3
3 D
15 L
17 D
```

spin

15
L

17
D

			↓	🍏	
			↓		
		🍏			

Time : 5









# Problem example

아래의 입력 예제 설명

```
6
3
3 4
2 5
5 3
3
3 D
15 L
17 D
```

spin	15	17
	L	D

Time : 6



# Problem example





아래의 입력 예제 설명

```
6
3
3 4
2 5
5 3
3
3 D
15 L
17 D
```

spin

15
L

17
D

Time : 7







# Problem example

아래의 입력 예제 설명

```
6
3
3 4
2 5
5 3
3
3 D
15 L
17 D
```

spin	15	17
	L	D

Time : 8

# Problem implementation

문제 요구 사항 정리

1. 게임 시작 시 뱀 몸의 길이 1, 좌상단에서 시작(1, 1)
2. 매 초 머리를 늘려 이동, 방향 전환(왼쪽 L, 오른쪽 D)도 존재
3. 사과 유무에 따라 몸 길이 증가 or 유지
4. 벽이나 자기 몸에 부딪히면 게임 끝



# Hint

## Step 1.

2. 매 초 머리를 늘려 이동, 방향 전환(왼쪽 L, 오른쪽 D)도 존재

- **N X N 보드 위에 뱀을 이동시키는 시뮬레이션을 진행할 때 방향 벡터 필요.**
  - $dx = [0, 1, 0, -1]$
  - $dy = [1, 0, -1, 0]$
- **방향 전환시 왼쪽(반시계), 오른쪽(시계)을 유념하자**
  - 반시계(상 -> 좌 -> 하 -> 우)
  - 시계(우 -> 하 -> 좌 -> 상)
  - 내 경우 방향 벡터를 시계방향(우하좌상) 순으로 초기화 해놓고 반시계 방향 구현시에는 3칸 이동.
  - 시계 방향 회전시에는  $dir = (dir + 1) \% 4$
  - 반시계 방향 회전시에는  $dir = (dir + 3) \% 4$



# Hint

## Step 2.

3. 사과 유무에 따라 몸 길이 증가 or 유지, 4. 벽이나 자기 몸에 부딪히면 게임 끝

- 벽이나 자기 몸에 부딪히면 게임이 끝난다.
  - 벽은 보드 밖, 즉 머리의 좌표 (x,y)가 1과 N 사이인지 확인하면 된다.
  - 하지만 뱀의 머리가 자기 몸에 부딪히는지 아닌지 판단하기 위해서는 뱀의 몸이 차지하고 있는 칸의 좌표들을 저장하고 있어야 한다.
    - 뱀의 움직임을 잘 생각해보자. 뱀의 움직임의 순서는 먼저 들어온 칸이 먼저 나간다. 즉 FIFO, queue.
- 사과 유무에 따라 몸길이 증가 or 유지.
  - 1. 사과를 먹을 경우: 머리 이동 후 queue에서 popleft하지 않음
  - 2. 사과를 먹지 않을 경우: 머리 이동 후 queue에서 popleft함

# ✨ Solution

```
dx = [0, 1, 0, -1]
dy = [1, 0, -1, 0]
queue = deque()
direction = 0
queue.append((1, 1))
time = 0
while True:
    if transition_idx < len(transitions):
        transition_time, transition_dir = transitions[transition_idx]
        if transition_time == time:
            if transition_dir == 'L':
                direction = (direction + 3) % 4
            elif transition_dir == 'D':
                direction = (direction + 1) % 4
            transition_idx += 1
    x, y = queue[len(queue) - 1]
    newX = x + dx[direction]
    newY = y + dy[direction]
    time += 1
    if (newX < 1 or newX > N) or (newY < 1 or newY > N) or ([newX, newY] in queue):
        break
    if not ([newX, newY] in apples):
        queue.popleft()
    else:
        apples.remove([newX, newY])
```

다른 풀이



# Assignment

## 백준 #2065. 나룻배 (골드2)

복잡한 시뮬레이션 문제 & 자료구조 활용 문제