

BOJ 17370.

육각형 우리 속의 재미

소프트웨어학과 201921017 이지우

Step1 – 접근

문제

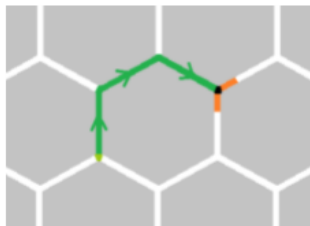
무한히 많은 정육각형이 서로 맞닿아 놓인 형태의 개미 우리가 있다. 다음 그림과 같은 형태이고, 하얀색 변으로만 개미가 다닐 수 있다.



개미 우리의 모습

곤충 관찰이 취미인 유이는 세 정육각형이 서로 맞닿아 있는 어떤 점 위에 개미를 하나 올렸다. 이렇게 우리에 올라온 개미는 그 자신에게 미지의 영역인 우리를 페로몬을 뿌리며 탐색하기 시작했다. 처음 개미는 점과 연결된 세 변 중 하나를 향해 이동을 시작하는데, 편의를 위해 이 첫 번째 이동이 북쪽을 향하도록 돌려서 보자.

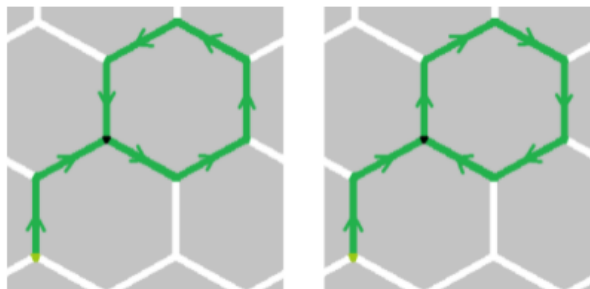
만약 개미가 변이 세 갈래로 갈라지는 점에 도착하면, 자신이 이동해온 변을 제외한 나머지 두 변 중 하나를 골라 그 방향으로 회전하여 탐색을 계속한다.



연두색은 시작 지점, 초록색은 개미가 탐색하며 페로몬을 뿌린 경로. 검은색은 개미, 주황색은 다음 이동을 위해 선택 가능한 두 변을 나타낸다.

Step1 – 접근

개미가 이전에 방문했던, 즉, 페로몬이 뿌려진 지점에 도착하면 이곳이 이미 익숙한 영역이라는 착각에 빠지고 더 이상의 탐색을 멈춘다. 이렇게 탐색을 멈췄을 때, 방향을 회전한 횟수가 정확히 N 번이 되는 경우의 수를 구해보자.



방향을 7번 회전하는 두 경로. 페로몬의 궤적은 동일해도 개미의 이동 방향에 따라 경로를 구별하도록 한다.

입력

첫 번째 줄에 하나의 정수 $N(1 \leq N \leq 22)$ 이 주어진다.

출력

첫 번째 줄에 개미가 방향 회전을 N 번 하고 멈추는 경우의 수를 출력한다.

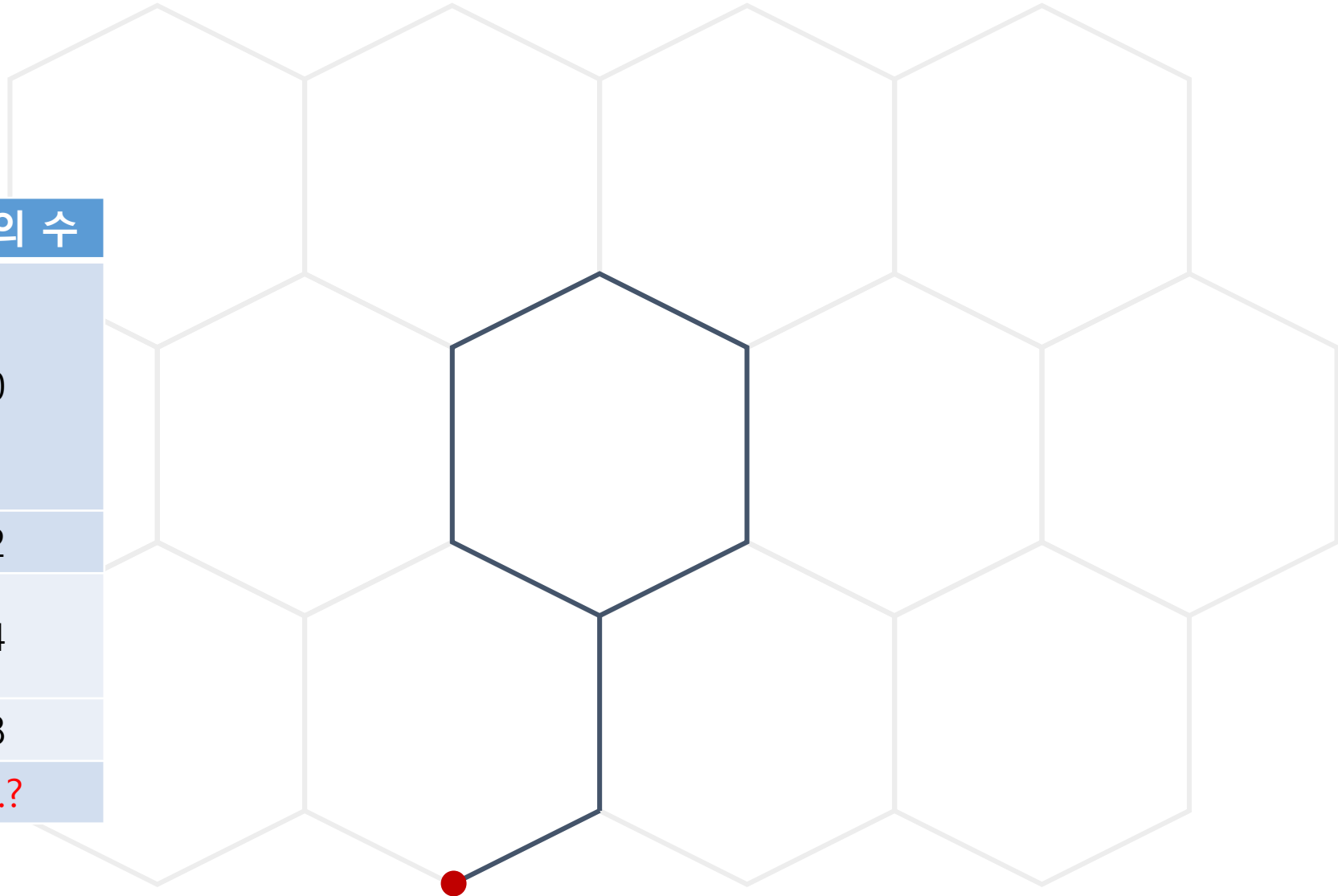
Step1 - 접근



Step1 – 접근

① DP인가?

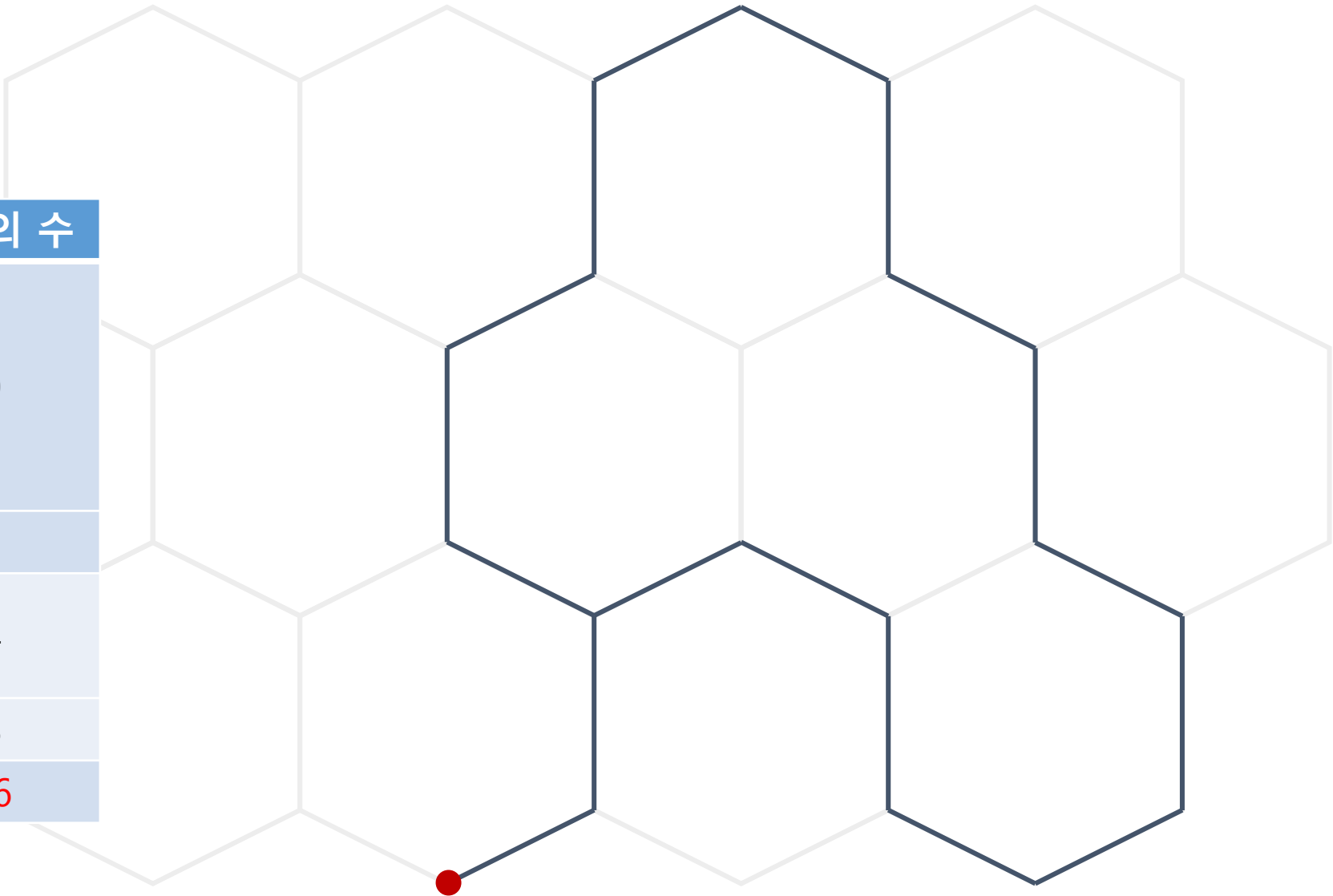
| N | 경우의 수 |
|---|-------|
| 1 | 0 |
| 2 | |
| 3 | |
| 4 | |
| 5 | 2 |
| 6 | 4 |
| 7 | |
| 8 | 8 |
| 9 | 8..? |



Step1 – 접근

① DP인가?

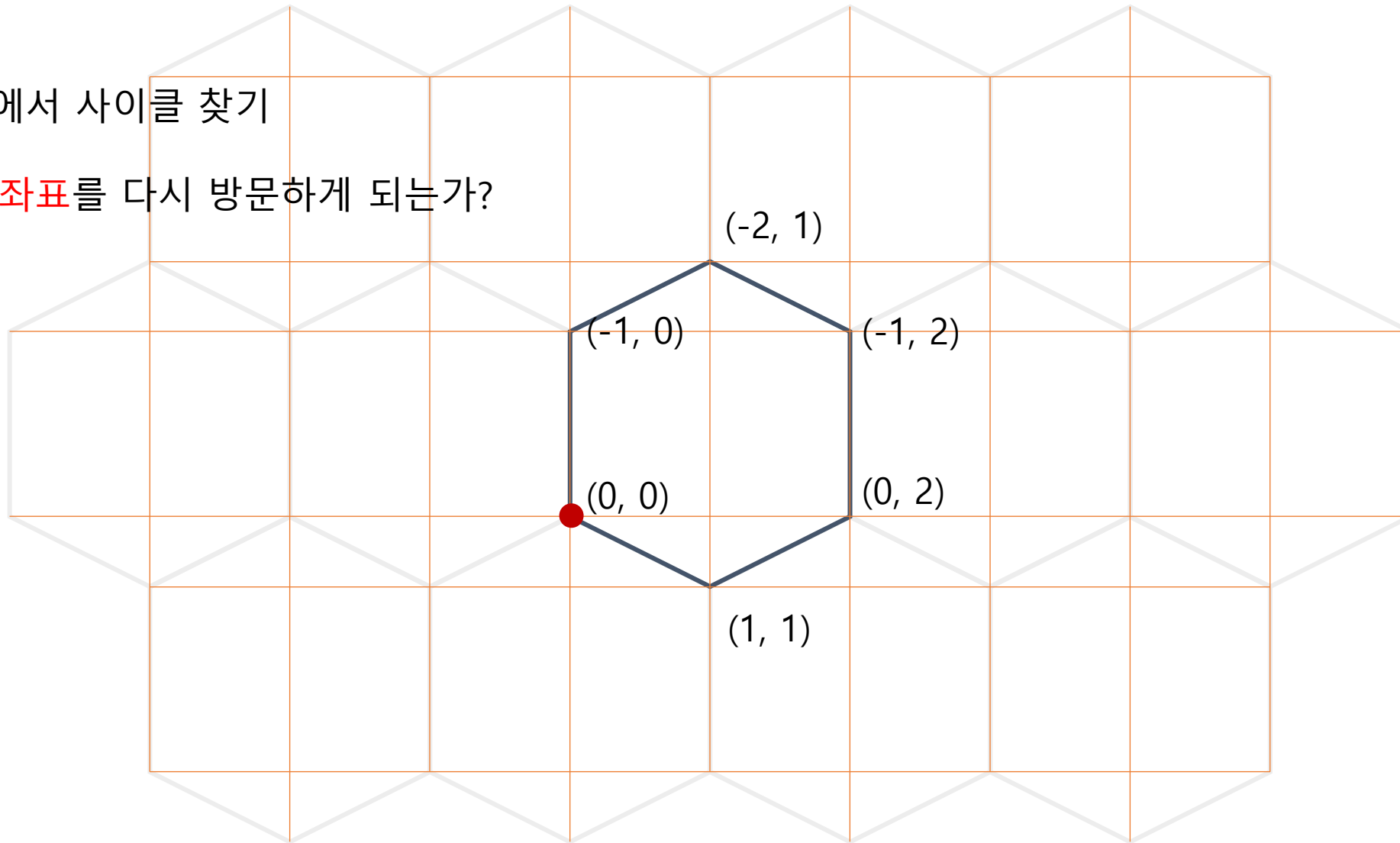
| N | 경우의 수 |
|---|-------|
| 1 | 0 |
| 2 | |
| 3 | |
| 4 | |
| 5 | 2 |
| 6 | 4 |
| 7 | |
| 8 | 8 |
| 9 | 26 |



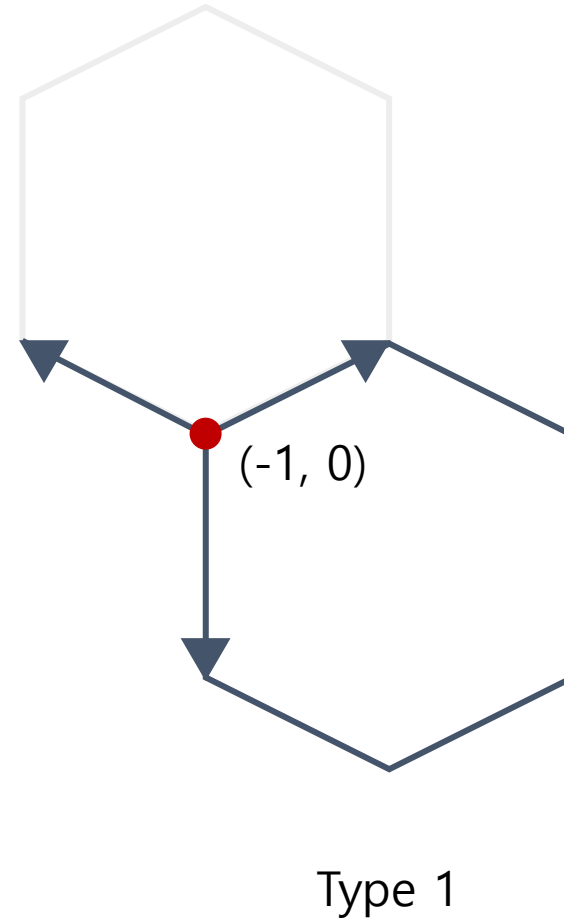
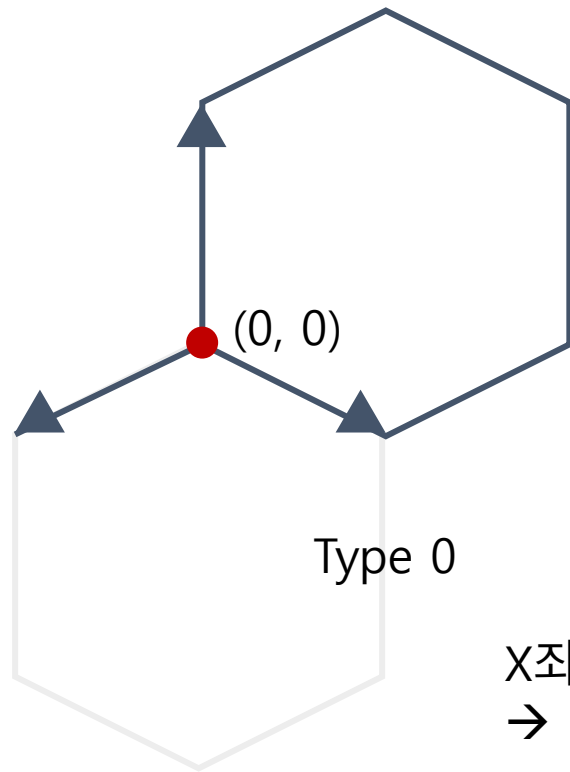
Step1 – 접근

② 그래프에서 사이클 찾기

- 방문한 좌표를 다시 방문하게 되는가?

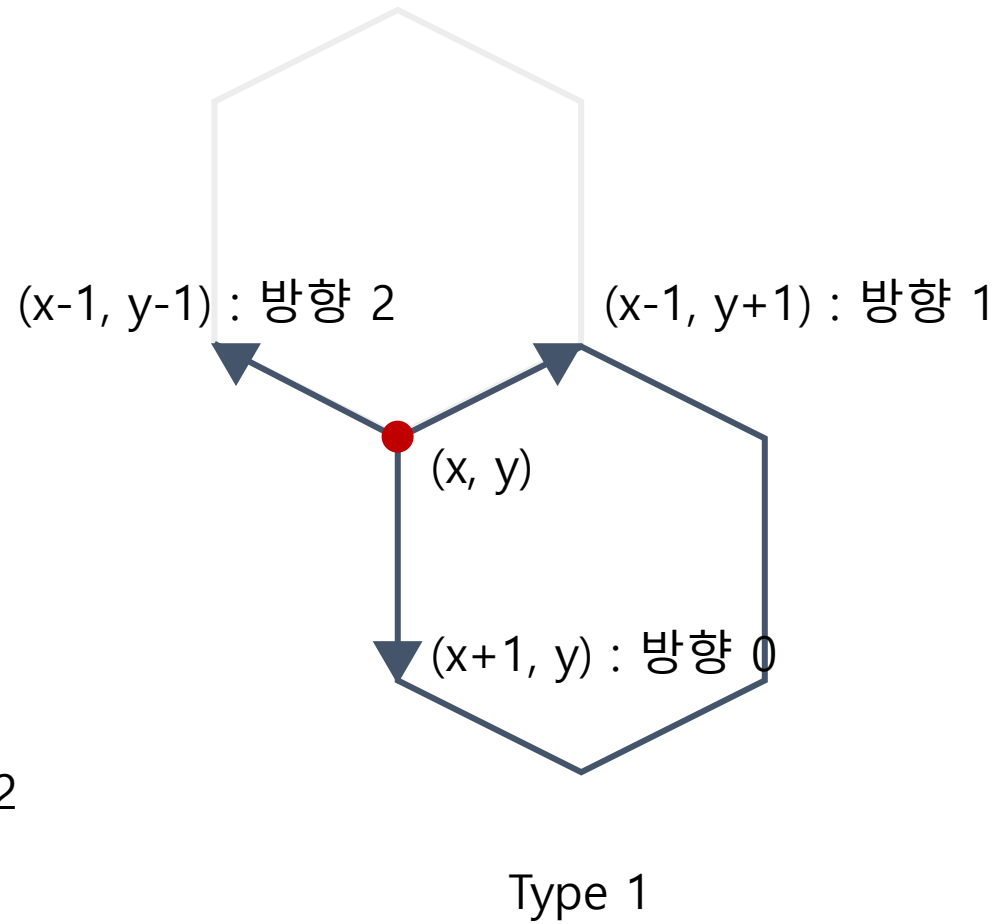
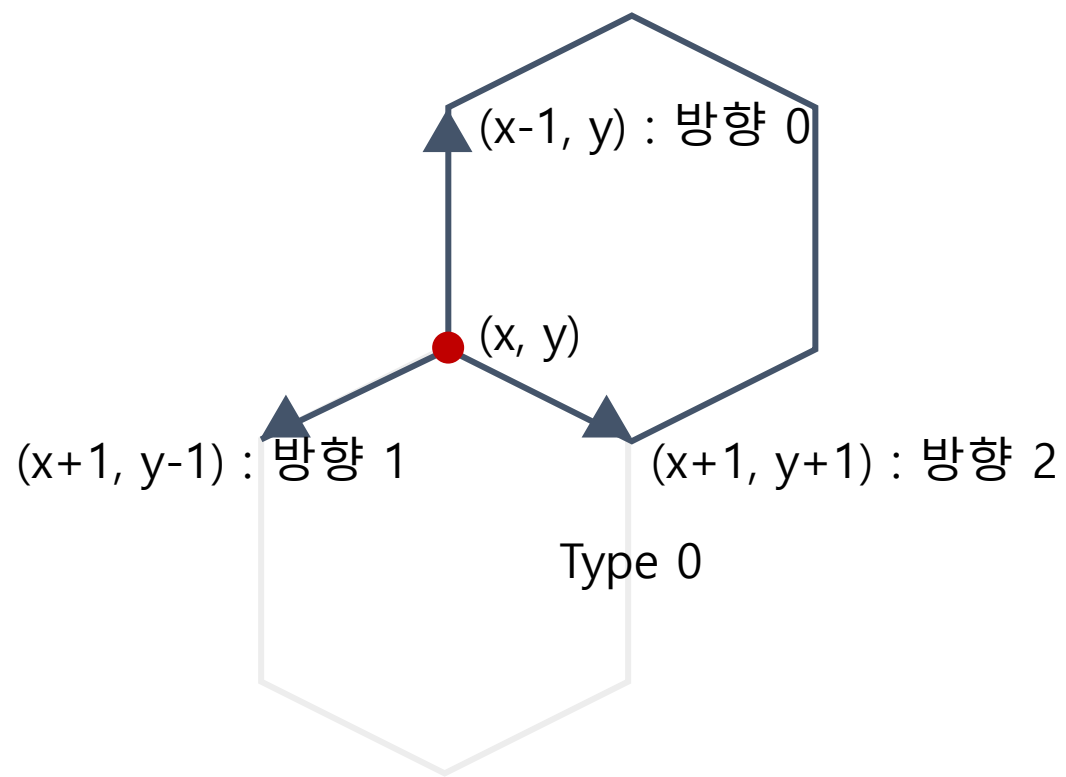


Step1 - 접근



x좌표에 따라 갈 수 있는 길이 달라짐
→ 개미가 한 번 회전할 때마다 갈 수 있는 길의 type이 바뀜

Step1 - 접근



Step2 - 구현

현재 위치에서 갈 수 있는 다음 좌표의 방향이 바뀜

```
#include <iostream>
#include <vector>

using namespace std;

typedef pair<int, int> pii;

int dir[2][3][2] = {
    { { -1, 0 }, { 1, -1 }, { 1, 1 } }, dir[0] : type 0
    { { 1, 0 }, { -1, 1 }, { -1, -1 } }, dir[1] : type 1
};

int n, answer = 0;
bool visited[1'000][1'000];
```

Step2 - 구현

```
void dfs(int prev, pii curr, int type, int cnt) {  
    if (cnt > n)  
        return;  
  
    if (visited[curr.first][curr.second]) {  
        if (cnt == n)  
            answer++;  
    }  
    else {  
        visited[curr.first][curr.second] = true;  
  
        for (int i = 0; i < 3; i++) { 갈 수 있는 3개의 방향  
            if (i == prev) 직전에 왔던 방향으로 갈 수 없음(continue)  
                continue;  
  
            pii next = {  
                curr.first + dir[type][i][0],  
                curr.second + dir[type][i][1]  
            };  
  
            dfs(i, next, !type, cnt + 1); 다음 위치로 이동  
                                                다음 위치에서는 갈 수 있는 방향(type)이 바뀜!  
        }  
  
        visited[curr.first][curr.second] = false;  
    }  
}
```

Step2 - 구현

```
int main(void)
{
    cin >> n;

    visited[100][100] = true;
    dfs(0, { 101, 100 }, 0, 0);

    cout << answer << '\n';





    return 0;
}
```

*첫번째 이동이 북쪽을 향하므로,



바로 직전 좌표($x+1, y$) 좌표에 대한 visited를 true로!

Step3 – 결과

| 아이디 | 문제 | 결과 | 메모리 | 시간 | 언어 |
|---------|---|---------|---------|--------|---------------|
| january |  17370 | 맞았습니다!! | 2996 KB | 32 ms | C++17 / 수정 |
| january |  17370 | 시간 초과 | | | PyPy3 / 수정 |
| january |  17370 | 시간 초과 | | | Python 3 / 수정 |
| january |  17370 | 맞았습니다!! | 2024 KB | 232 ms | C++17 / 수정 |

End