

BFS (연습)

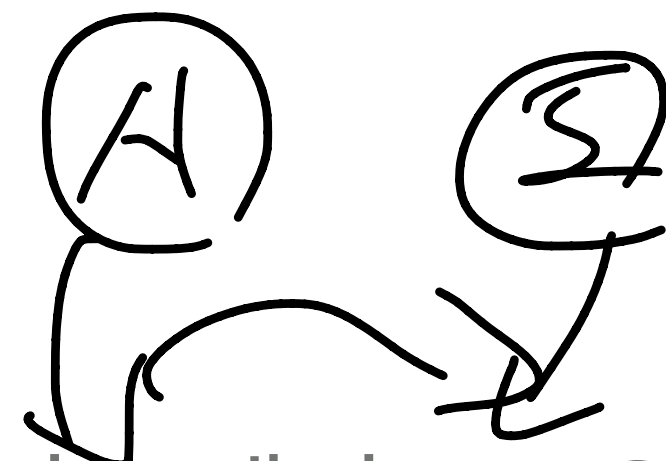
최백준 choi@startlink.io

뱀과 사다리 게임

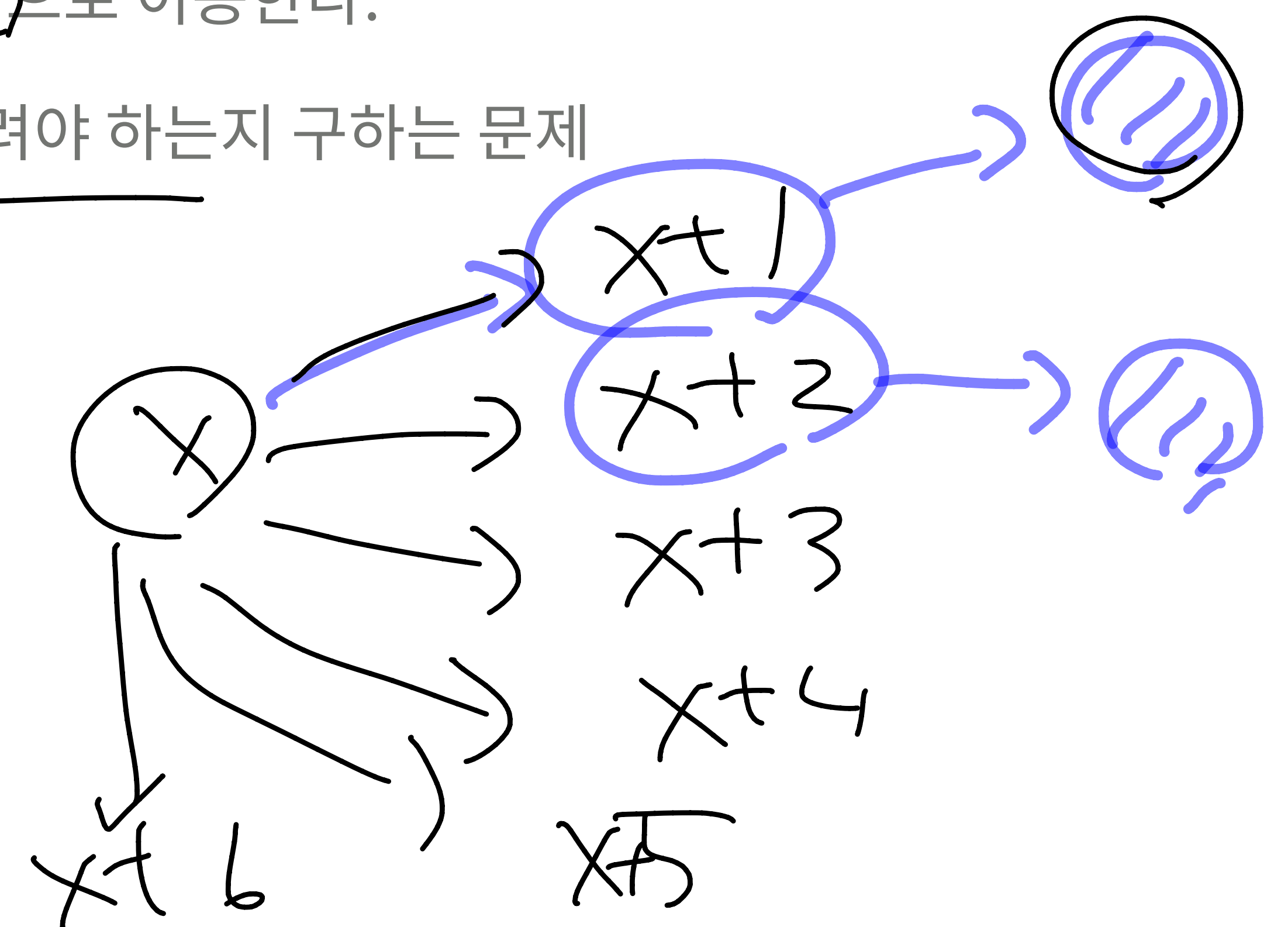
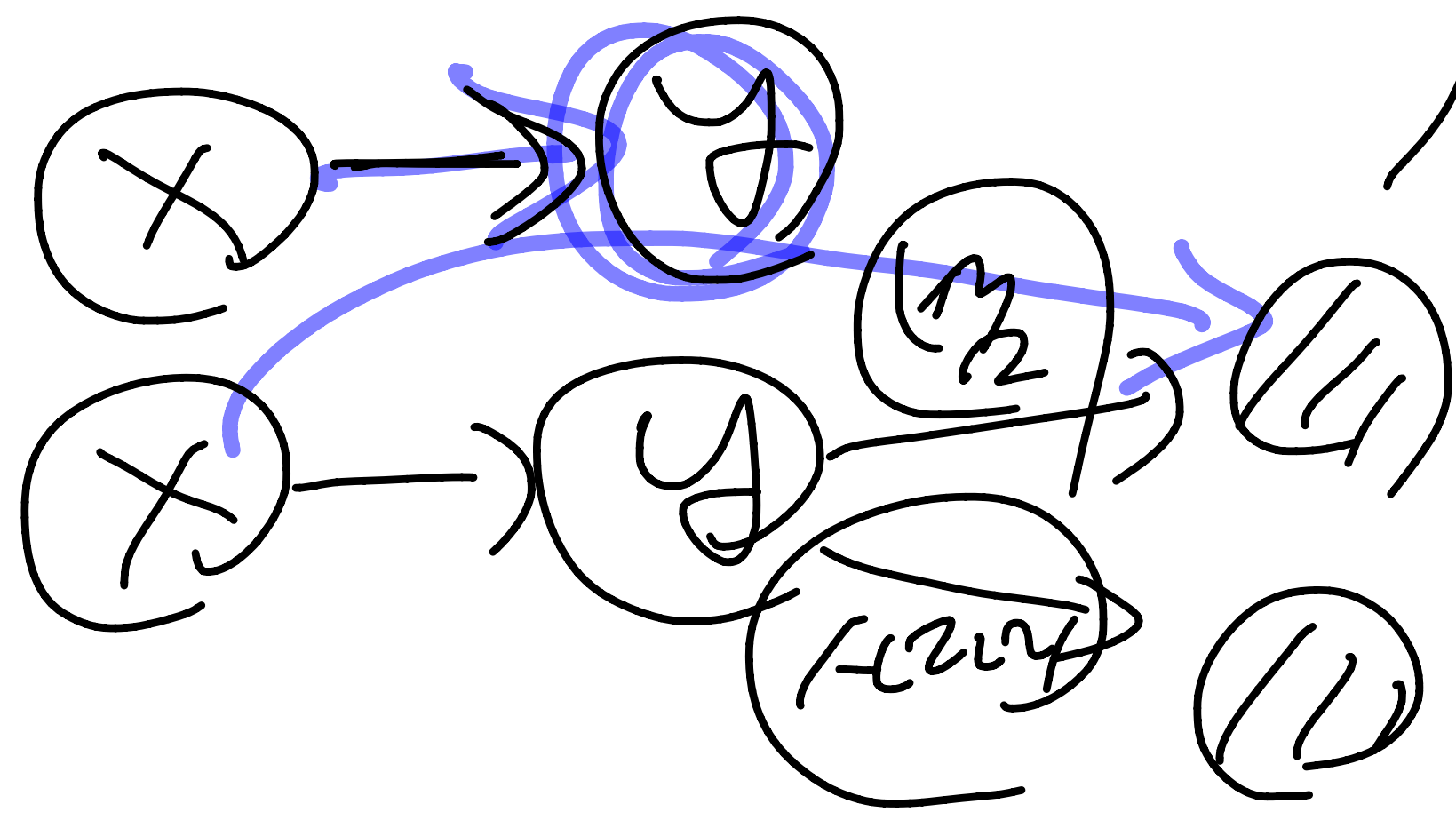
<https://www.acmicpc.net/problem/16928>

사다리 / 뱀 X

2



- 100개의 칸으로 나누어져 있는 게임이 있다. 1에서 100으로 가야 한다.
- 주사위를 굴려 나온 수만큼 이동할 수 있으며, 도착한 칸이 사다리인 경우에는 사다리를 타고 더 큰 번호의 칸으로, 뱀인 경우에는 더 작은 번호의 칸으로 이동한다.
- 주사위에 나온 수를 정할 수 있을 때, 최소 몇 번 굴려야 하는지 구하는 문제



뱀과 사다리 게임

<https://www.acmicpc.net/problem/16928>

- 최소를 구하는 문제이기 때문에, BFS로 해결할 수 있다.

뱀과 사다리 게임

<https://www.acmicpc.net/problem/16928>

- 까다로운 점은 뱀과 사다리의 처리
- 도착한 이후에 뱀이나 사다리인 경우를 처리해야 함

뱀과 사다리 게임

<https://www.acmicpc.net/problem/16928>

5

$$\underline{\text{next}[x] = y}$$

- 게임에서 뱀과 사다리의 구분은 중요하지만, 구현에서는 별로 중요하지 않다.

- $x \rightarrow y$ 로 간다는 점만 중요하다.

- 새로운 배열 next[x]를 만들어서, x에 도착한 이후에 가야할 곳을 모두 기록

- 뱀이나 사다리인 경우

 - $\text{next}[x] = y$

- 일반 칸인 경우

 - $\text{next}[x] = x$



뱀과 사다리 게임

<https://www.acmicpc.net/problem/16928>

- 소스: <http://codeplus.codes/747f3a8b31dc46c09f052d69271dfed9>

데스 나이트

<https://www.acmicpc.net/problem/16948>

- 데스 나이트는 (r, c) 에서 $(r-2, c-1), (r-2, c+1), (r, c-2), (r, c+2), (r+2, c-1), (r+2, c+1)$ 로 이동할 수 있는 말이다.

- 크기가 $N \times N$ 인 체스판과 두 칸 $(r1, c1), (r2, c2)$ 가 주어졌을 때 $(r1, c1)$ 에서 $(r2, c2)$ 로 가는 최소 이동 횟수를 구하는 문제

$5 \leq N \leq 200$

12/3 연습

정답: N^2

$O(N^2)$

데스 나이트

<https://www.acmicpc.net/problem/16948>

- 데스 나이트는 (r, c) 에서 $(r-2, c-1)$, $(r-2, c+1)$, $(r, c-2)$, $(r, c+2)$, $(r+2, c-1)$, $(r+2, c+1)$ 로 이동할 수 있는 말이다.
- 크기가 $N \times N$ 인 체스판과 두 칸 $(r1, c1)$, $(r2, c2)$ 가 주어졌을 때, $(r1, c1)$ 에서 $(r2, c2)$ 로 가는 **최소 이동 횟수**를 구하는 문제
- $5 \leq N \leq 200$
- BFS로 해결할 수 있는 문제이다.

데스 나이트

<https://www.acmicpc.net/problem/16948>

- 소스: <http://codeplus.codes/68198b618e664c69a5bcbdf8d9447382>

DSLR

<https://www.acmicpc.net/problem/9019>

- 네 자리 숫자 A와 B가 주어졌을 때

- A -> B로 바꾸는 최소 연산 횟수

- D: N -> 2*N

- S: N -> N-1

- L: 한 자리씩 왼쪽으로

- R: 한 자리씩 오른쪽으로

7877

0000 -> 9999

[0, 5000]

1234 \xrightarrow{D} 2468

\xrightarrow{S} 1233

\xrightarrow{L} 2341

\xrightarrow{R} 4123

DSLR

<https://www.acmicpc.net/problem/9019>

- 이 문제는 최소값을 구해야 하는건 맞지만
- 어떠한 과정을 거쳐야 하는지를 구해야 한다
- 배열을 하나 더 이용해서 어떤 과정을 거쳤는지를 저장해야 한다
- $\text{how}[i] = i$ 를 어떻게 만들었는지

DSL R

0 → 55559

12

<https://www.acmicpc.net/problem/9019>

- 1 → 9을 만드는 경우
- 1~10까지만 있다고 가정

dist[1] = 0, from[1] = 1, how[1] = ?

dist[2] = ?, from[2] = ?, how[2] = ?

dist[3] = ?, from[3] = ?, how[3] = ?

dist[4] = ?, from[4] = ?, how[4] = ?

dist[5] = ?, from[5] = ?, how[5] = ?

dist[6] = ?, from[6] = ?, how[6] = ?

dist[7] = ?, from[7] = ?, how[7] = ?

dist[8] = ?, from[8] = ?, how[8] = ?

dist[9] = ?, from[9] = ?, how[9] = ?

dist[10] = ?, from[10] = ?, how[10] = ?

DSL R

13

<https://www.acmicpc.net/problem/9019>

- 1 -> 9을 만드는 경우

- 1~10까지만 있다고 가정

- 큐: 1

1 → 2
1 → 3
1 → 4

dist[1] = 0, from[1] = -1, how[1] = ''
dist[2] = 1, from[2] = 1, how[2] = 'D'
dist[3] = ?, from[3] = ?, how[3] = ?
dist[4] = ?, from[4] = ?, how[4] = ?
dist[5] = ?, from[5] = ?, how[5] = ?
dist[6] = ?, from[6] = ?, how[6] = ?
dist[7] = ?, from[7] = ?, how[7] = ?
dist[8] = ?, from[8] = ?, how[8] = ?
dist[9] = ?, from[9] = ?, how[9] = ?
dist[10] = 1, from[10] = 1, how[10] = 'L'

DSL R

<https://www.acmicpc.net/problem/9019>

- 1 -> 9을 만드는 경우
- 1~10까지만 있다고 가정
- 큐: 1
- 1 -> 2 (D)
- 1 -> 0 (S)
- 1 -> 10 (L)
- 1 -> 1000 (R)
- 큐: 2 10

`dist[1] = 0, from[1] = -1, how[1] = ''`

`dist[2] = 1, from[2] = 1, how[2] = D`

`dist[3] = ?, from[3] = ?, how[3] = ?`

`dist[4] = ?, from[4] = ?, how[4] = ?`

`dist[5] = ?, from[5] = ?, how[5] = ?`

`dist[6] = ?, from[6] = ?, how[6] = ?`

`dist[7] = ?, from[7] = ?, how[7] = ?`

`dist[8] = ?, from[8] = ?, how[8] = ?`

`dist[9] = ?, from[9] = ?, how[9] = ?`

`dist[10] = 1, from[10] = 1, how[10] = L`

DSL R

15

<https://www.acmicpc.net/problem/9019>

- 1 -> 9을 만드는 경우
- 1~10까지만 있다고 가정
- 큐: 2 10

2 ^D→ 4

dist[1]	= 0,	from[1]	= -1,	how[1]	= ''
dist[2]	= ①,	from[2]	= 1,	how[2]	= ①D
dist[3]	= ?,	from[3]	= ?,	how[3]	= ?
dist[4]	= <u>2</u> ,	from[4]	= ②,	how[4]	= ①D
dist[5]	= ?,	from[5]	= ?,	how[5]	= ?
dist[6]	= ?,	from[6]	= ?,	how[6]	= ?
dist[7]	= ?,	from[7]	= ?,	how[7]	= ?
dist[8]	= ?,	from[8]	= ?,	how[8]	= ?
dist[9]	= ?,	from[9]	= ?,	how[9]	= ?
dist[10]	= 1,	from[10]	= 1,	how[10]	= L

DSL R

<https://www.acmicpc.net/problem/9019>

- 1 -> 9을 만드는 경우
- 1~10까지만 있다고 가정
- 큐: 2 10
- 2 -> 4 (D)
- 2 -> 1 (S)
- 2 -> 20 (L)
- 2 -> 2000 (R)
- 큐: 10 4

`dist[1] = 0, from[1] = -1, how[1] = ''`

`dist[2] = 1, from[2] = 1, how[2] = D`

`dist[3] = ?, from[3] = ?, how[3] = ?`

`dist[4] = 2, from[4] = 2, how[4] = D`

`dist[5] = ?, from[5] = ?, how[5] = ?`

`dist[6] = ?, from[6] = ?, how[6] = ?`

`dist[7] = ?, from[7] = ?, how[7] = ?`

`dist[8] = ?, from[8] = ?, how[8] = ?`

`dist[9] = ?, from[9] = ?, how[9] = ?`

`dist[10] = 1, from[10] = 1, how[10] = L`

DSL R

<https://www.acmicpc.net/problem/9019>

- 1 -> 9을 만드는 경우
- 1~10까지만 있다고 가정
- 큐: 10 4

`dist[1] = 0, from[1] = -1, how[1] = ''`

`dist[2] = 1, from[2] = 1, how[2] = D`

`dist[3] = ?, from[3] = ?, how[3] = ?`

`dist[4] = 2, from[4] = 2, how[4] = D`

`dist[5] = ?, from[5] = ?, how[5] = ?`

`dist[6] = ?, from[6] = ?, how[6] = ?`

`dist[7] = ?, from[7] = ?, how[7] = ?`

`dist[8] = ?, from[8] = ?, how[8] = ?`

`dist[9] = ?, from[9] = ?, how[9] = ?`

`dist[10] = 1, from[10] = 1, how[10] = L`

DSL R

<https://www.acmicpc.net/problem/9019>

- 1 -> 9을 만드는 경우
- 1~10까지만 있다고 가정
- 큐: 10 4
- 10 -> 20 (D)
- 10 -> 9 (S)
- 10 -> 100 (L)
- 10 -> 1 (R)
- 큐: 4 9

```
dist[1] = 0, from[1] = -1, how[1] = ''
dist[2] = 1, from[2] = 1, how[2] = D
dist[3] = ?, from[3] = ?, how[3] = ?
dist[4] = 2, from[4] = 2, how[4] = D
dist[5] = ?, from[5] = ?, how[5] = ?
dist[6] = ?, from[6] = ?, how[6] = ?
dist[7] = ?, from[7] = ?, how[7] = ?
dist[8] = ?, from[8] = ?, how[8] = ?
dist[9] = 2, from[9] = 10, how[9] = S
dist[10] = 1, from[10] = 1, how[10] = L
```

DSL R

<https://www.acmicpc.net/problem/9019>

- 1 -> 9을 만드는 경우
- 1~10까지만 있다고 가정
- 큐: 4 9

`dist[1] = 0, from[1] = -1, how[1] = ''`

`dist[2] = 1, from[2] = 1, how[2] = D`

`dist[3] = ?, from[3] = ?, how[3] = ?`

`dist[4] = 2, from[4] = 2, how[4] = D`

`dist[5] = ?, from[5] = ?, how[5] = ?`

`dist[6] = ?, from[6] = ?, how[6] = ?`

`dist[7] = ?, from[7] = ?, how[7] = ?`

`dist[8] = ?, from[8] = ?, how[8] = ?`

`dist[9] = 2, from[9] = 10, how[9] = S`

`dist[10] = 1, from[10] = 1, how[10] = L`

DSL R

<https://www.acmicpc.net/problem/9019>

- 1 -> 9을 만드는 경우
- 1~10까지만 있다고 가정
- 큐: 4 9
- 4 -> 8 (D)
- 4 -> 3 (S)
- 4 -> 40 (L)
- 4 -> 4000 (R)
- 큐: 8 3

dist[1] = 0, from[1] = -1, how[1] = ''
dist[2] = 1, from[2] = 1, how[2] = D
dist[3] = 3, from[3] = 4, how[3] = S
dist[4] = 2, from[4] = 2, how[4] = D
dist[5] = ?, from[5] = ?, how[5] = ?
dist[6] = ?, from[6] = ?, how[6] = ?
dist[7] = ?, from[7] = ?, how[7] = ?
dist[8] = 3, from[8] = 4, how[8] = D
dist[9] = 2, from[9] = 10, how[9] = S
dist[10] = 1, from[10] = 1, how[10] = L

DSL R

<https://www.acmicpc.net/problem/9019>

- 1 → 9을 만드는 경우
- 1~10까지만 있다고 가정
- 모두 채워보면
- 오른쪽과 같다

```
dist[1] = 0, from[1] = -1, how[1] = ''
dist[2] = 1, from[2] = 1, how[2] = D
dist[3] = 3, from[3] = 4, how[3] = S
dist[4] = 2, from[4] = 2, how[4] = D
dist[5] = 5, from[5] = 6, how[5] = S
dist[6] = 4, from[6] = 3, how[6] = D
dist[7] = 4, from[7] = 8, how[7] = S
dist[8] = 3, from[8] = 4, how[8] = D
dist[9] = 2, from[9] = 10, how[9] = S
dist[10] = 1, from[10] = 1, how[10] = L
```

DSL R

<https://www.acmicpc.net/problem/9019>

- 1 -> 9을 만드는 경우
- 1~10까지만 있다고 가정
- 9에서 시작해서
- 1을 찾아나가면 된다
- 방법 (역순): S

`dist[1] = 0, from[1] = -1, how[1] = ''`

`dist[2] = 1, from[2] = 1, how[2] = D`

`dist[3] = 3, from[3] = 4, how[3] = S`

`dist[4] = 2, from[4] = 2, how[4] = D`

`dist[5] = 5, from[5] = 6, how[5] = S`

`dist[6] = 4, from[6] = 3, how[6] = D`

`dist[7] = 4, from[7] = 8, how[7] = S`

`dist[8] = 3, from[8] = 4, how[8] = D`

`dist[9] = 2, from[9] = 10, how[9] = S`

`dist[10] = 1, from[10] = 1, how[10] = L`

DSL R

23

<https://www.acmicpc.net/problem/9019>

- 1 -> 9을 만드는 경우
- 1~10까지만 있다고 가정
- 9에서 시작해서
- 1을 찾아나가면 된다
- 방법 (역순): SL

`dist[1] = 0, from[1] = -1, how[1] = ''`

`dist[2] = 1, from[2] = 1, how[2] = D`

`dist[3] = 3, from[3] = 4, how[3] = S`

`dist[4] = 2, from[4] = 2, how[4] = D`

`dist[5] = 5, from[5] = 6, how[5] = S`

`dist[6] = 4, from[6] = 3, how[6] = D`

`dist[7] = 4, from[7] = 8, how[7] = S`

`dist[8] = 3, from[8] = 4, how[8] = D`

`dist[9] = 2, from[9] = 10, how[9] = S`

`dist[10] = 1, from[10] = 1, how[10] = L`

DSLR

24

<https://www.acmicpc.net/problem/9019>

- 1 -> 9을 만드는 경우
- 1~10까지만 있다고 가정
- 9에서 시작해서
- 1을 찾아나가면 된다
- 방법 (역순): SL

LS

dist[1] = 0, from[1] = -1, how[1] = ' '

dist[2] = 1, from[2] = 1, how[2] = D

dist[3] = 3, from[3] = 4, how[3] = S

dist[4] = 2, from[4] = 2, how[4] = D

dist[5] = 5, from[5] = 6, how[5] = S

dist[6] = 4, from[6] = 3, how[6] = D

dist[7] = 4, from[7] = 8, how[7] = S

dist[8] = 3, from[8] = 4, how[8] = D

dist[9] = 2, from[9] = 10, how[9] = S

dist[10] = 1, from[10] = 1, how[10] = L

DSL

<https://www.acmicpc.net/problem/9019>

```
int next = (now*2) % 10000;  
if (check[next] == false) {  
    q.push(next);  
    check[next] = true;  
    dist[next] = dist[now]+1;  
    from[next] = now;  
    how[next] = 'D';  
}
```

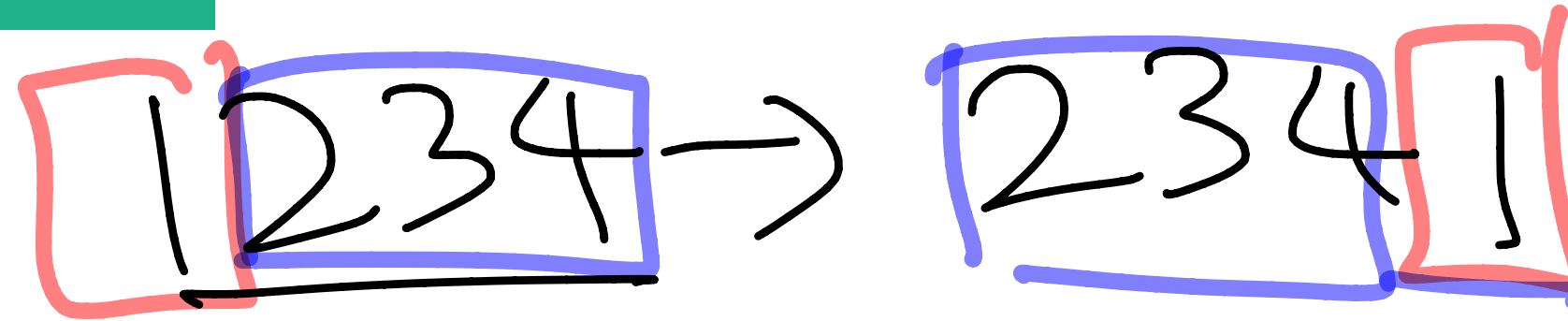
now → next

DSLR

<https://www.acmicpc.net/problem/9019>

```
next = now-1;
if (next == -1) next = 9999;
if (check[next] == false) {
    q.push(next);
    check[next] = true;
    dist[next] = dist[now]+1;
    from[next] = now;
    how[next] = 'S';
}
```

DSL



27

<https://www.acmicpc.net/problem/9019>

```
next = (now%1000)*10 + now/1000; ←  
if (check[next] == false) {  
    q.push(next);  
    check[next] = true;  
    dist[next] = dist[now]+1;  
    from[next] = now;  
    how[next] = 'L'; ←  
}
```

DSL

<https://www.acmicpc.net/problem/9019>

```
next = (now/10) + (now%10)*1000;  
if (check[next] == false) {  
    q.push(next);  
    check[next] = true;  
    dist[next] = dist[now]+1;  
    from[next] = now;  
    how[next] = 'R';  
}
```

1234 → [4|23]

DSL

$A \rightarrow B$

29

<https://www.acmicpc.net/problem/9019>

```
string ans = "";
```

```
while (B) != (A) {
```

```
    ans += how[B];
```

```
    B = from[B];
```

```
}
```

```
reverse(ans.begin(), ans.end());
```

```
cout << ans << '\n';
```

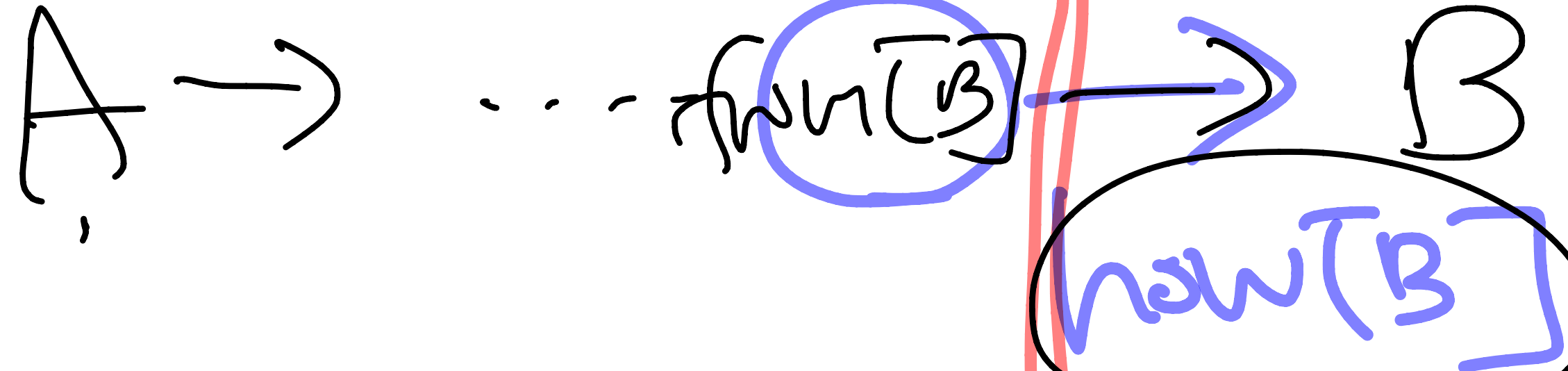
$\frac{\text{from}[B]}{11} \rightarrow B$

$A \rightarrow \text{how}[B] \rightarrow B$

DSL R

<https://www.acmicpc.net/problem/9019>

```
void print(int A, int B) {  
    if (A == B) return;  
    print(A, from[B]);  
    cout << how[B];  
}
```



DSLR

<https://www.acmicpc.net/problem/9019>

- 이 문제는 최소값을 구해야 하는건 맞지만
- 어떠한 과정을 거쳐야 하는지를 구해야 한다
- 배열을 하나 더 이용해서 어떤 과정을 거쳤는지를 저장해야 한다
- $\text{how}[i] = i$ 를 어떻게 만들었는지 (**모두 기록**)
- 위와 같이 어떻게 만들었는지를 모두 기록하면 안된다
- 모두 기록하면 공간이 매우 많이 필요하게 된다

DSL R

<https://www.acmicpc.net/problem/9019>

32

10000

- 1 -> 9을 만드는 경우
- 1~10까지만 있다고 가정
- 모두 채워보면
- 오른쪽과 같다

~~dist[1] = 0, from[1] = -1, how[1] = ''
dist[2] = 1, from[2] = 1, how[2] = D
dist[3] = 3, from[3] = 4, how[3] = DDS
dist[4] = 2, from[4] = 2, how[4] = DD
dist[5] = 5, from[5] = 6, how[5] = DDSDS
dist[6] = 4, from[6] = 3, how[6] = DDSD
dist[7] = 4, from[7] = 8, how[7] = DDDS
dist[8] = 3, from[8] = 4, how[8] = DDD
dist[9] = 2, from[9] = 10, how[9] = LS
dist[10] = 1, from[10] = 1, how[10] = L~~

DSL R

<https://www.acmicpc.net/problem/9019>

- 소스: <http://codeplus.codes/3dda382caf15493fa9e5ae91605860d0>

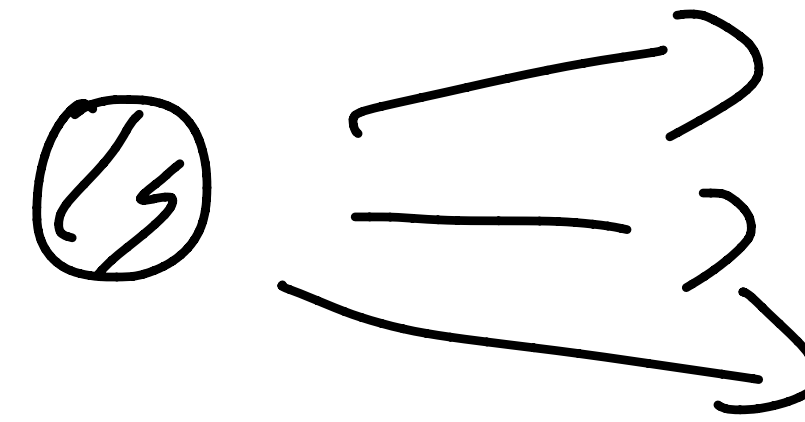
연구소

<https://www.acmicpc.net/problem/14502>

- $N \times M$ 크기의 직사각형 지도가 있고, 1×1 크기의 칸으로 나누어져 있다. ($3 \leq N, M \leq 8$)
- 칸: 빈 칸, 벽
- 일부 빈 칸에는 바이러스가 있고, 인접한 빈 칸으로 계속해서 퍼져 나간다
- 벽을 3개 세워 바이러스가 퍼질 수 없는 곳의 크기를 구하는 문제

빈 칸 \rightarrow 벽

최소



- $N \times M$ 크기의 직사각형 지도가 있고, 1×1 크기의 칸으로 나누어져 있다. ($3 \leq N, M \leq 8$)
- 칸: 빈 칸, 벽
- 일부 빈 칸에는 바이러스가 있고, 인접한 빈 칸으로 계속해서 퍼져 나간다.
- 벽을 세운다는 내용을 잠시 제외하면
- 입력으로 주어진 상태에서 바이러스가 퍼질 수 없는 영역의 크기는 BFS로 구할 수 있다.

DFS/BFS의 목적

· 한 공간에서 시작해 인접한 모든

공간을 방문

DFS

<https://www.acmicpc.net/problem/14502>

- $N \times M$ 크기의 직사각형 지도가 있고, 1×1 크기의 칸으로 나누어져 있다. ($3 \leq N, M \leq 8$)
- 칸: 빈 칸, 벽
- 일부 빈 칸에는 바이러스가 있고, 인접한 빈 칸으로 계속해서 퍼져 나간다.
- 벽을 세운다는 내용을 잠시 제외하면
- 입력으로 주어진 상태에서 바이러스가 퍼질 수 없는 영역의 크기는 BFS로 구할 수 있다.
- 칸은 정점, 인접한 칸의 관계는 간선으로 나타내면, 바이러스에서 시작해서 연결된 모든 정점을 방문하는 문제가 되어버리기 때문.
- 시간 복잡도: $O(NM)$

<https://www.acmicpc.net/problem/14502>

$$(NM)^3 \times NM$$

- $N \times M$ 크기의 직사각형 지도가 있고, 1×1 크기의 칸으로 나누어져 있다. ($3 \leq N, M \leq 8$)
- 칸: 빈 칸, 벽
- 일부 빈 칸에는 바이러스가 있고, 인접한 빈 칸으로 계속해서 퍼져 나간다.
- 벽을 3개 세우는 경우의 수: $(NM)^3$
- 벽을 세운 다음 안전 영역의 크기를 구하는 방법: BFS 또는 DFS, $O(NM)$
- 총 $O((NM)^4)$ 가 나오는데, $N, M \leq 8$ 이기 때문에, 시간 안에 해결할 수 있다.

$$(8 \times 8)^4 = 64^4 = (2^6)^4 = 2^{24} = \underline{16777216}$$

$O((NM)^4)$

<https://www.acmicpc.net/problem/14502>

$$(NM)^3$$

- BFS 소스: <http://codeplus.codes/a89404cb9a2d4df3adbc81d43b66fde5>
- DFS 소스: <http://codeplus.codes/5737ce19725f4074952d9b26d6e55529>

$$NM \times K$$

$$2^N \quad (2 \times 3)$$

돌 그룹

DFS, BFS

$(1500)^3$

39

<https://www.acmicpc.net/problem/12886>

- 돌이 3개의 그룹으로 나누어져 있고, 각 그룹에는 돌이 A, B, C개 있다. $(A, B, C \leq 500)$
- 돌은 단계별로 움직이고, 각 단계는 다음과 같다.
- 크기가 같지 않은 두 그룹을 고른다. 돌의 개수가 작은 쪽을 X, 큰 쪽을 Y라고 한다.
- X에 있는 돌의 개수를 $X+X$ 개로, Y에 있는 돌의 개수를 $Y-X$ 로 만든다.
- A, B, C가 주어졌을 때, 모든 그룹에 들어있는 돌의 개수를 같게 만들 수 있는지 구하는 문제

정답:

(A, B, C)

$(499, 500)$

$\hookrightarrow (998, 1)$

돌 그룹

<https://www.acmicpc.net/problem/12886>

- BFS로 해결할 수 있다.
- 정점: (A, B, C) 또는 (A, B)
- 전체 정점의 개수: $A+B+C$ 개

$$(X, Y) \rightarrow (X+X, Y-X)$$

$X+Y$ $X+Y$

$$(1500)^3 = 3375000000$$

$$N = A + B + C$$

$$33^3$$

$$33^3 B$$

$$36^3 B$$

$$\frac{2^{30} B}{10^9 B} = 16 B$$

$$(A, B)$$

$$C = N - A - B$$

$$1500^2 = \underline{2250000}$$

돌 그룹

<https://www.acmicpc.net/problem/12886>

- 소스: <http://codeplus.codes/cf0e413c77944fd68937666f9030a99d>

DFS

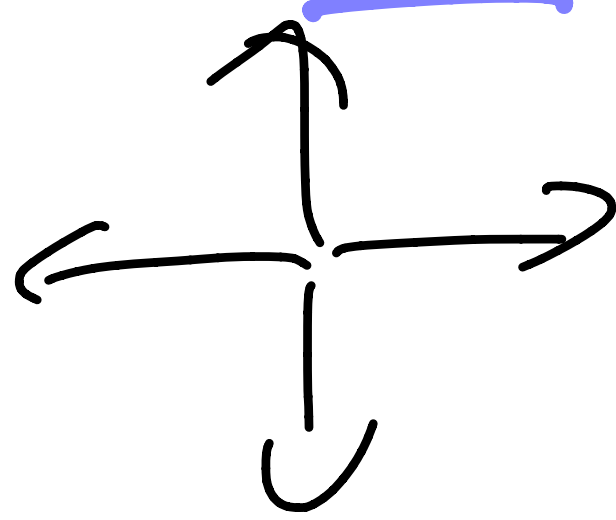
벽 부수고 이동하기

<https://www.acmicpc.net/problem/2206>

- NxM의 행렬로 나타내는 지도에서 (1, 1)에서 (N,M)으로 최단 거리로 이동하는 문제
- 0은 빈 칸, 1은 벽
- 단, 벽은 한 번 부수고 지나갈 수 있다

최단 거리 찾기

최단 거리



거리 찾기

빈 칸

(r, c, k)

벽을 부수기

빈 칸

빈 칸



$(r, c, 0)$

빈 칸

$(r, c, 1)$

빈 칸

벽

벽 부수고 이동하기

<https://www.acmicpc.net/problem/2206>

- 벽을 부순다는 조건이 없으면 일반적인 미로 탐색 문제이다
- 어떤 칸에 방문했을 때, 벽을 부순 적이 있는 경우와 아직 부순 적이 없는 경우는 다른 경우이기 때문에
- 상태 (i, j) 대신에 (i, j, k) ($k == 0$ 이면 벽을 부순 적이 없음, 1이면 있음) 으로 BFS 탐색을 진행한다.

벽 부수고 이동하기

<https://www.acmicpc.net/problem/2206>

- 소스: <http://codeplus.codes/c701d4a56bbb4abc872c1ef45759a1d4>

벽 부수고 이동하기 4

<https://www.acmicpc.net/problem/16946>

DFS/BFS

이

총

45

- $N \times M$ 크기의 지도가 있다.
- 0은 이동할 수 있는 곳(빈 칸), 1은 이동할 수 없는 곳(벽)
- 두 칸이 변을 공유할 때, 인접하다고 한다.
- 각각의 벽을 빈 칸으로 바꾸고, 그 위치에서 이동할 수 있는 칸의 개수 % 10을 구해보자.

1	1	0	0	1
0	0	1	1	1
0	1	0	1	0
1	0	1	0	1

4	6	0	0	3
0	0	7	3	2
0	6	0	4	0
5	0	4	0	3

N, M

$O(NM)$

$N, M \leq 1,000$

$(NM)^2$

$(1000)^2 = 10^6$ $(1000 \times 1000)^2$

벽 부수고 이동하기 4

<https://www.acmicpc.net/problem/16946>

DFS/BFS

- 이동할 수 있는 빈 칸을 모두 그룹 짓고
- 몇 개의 칸을 이루어져 있는지 계산해보자
- 그룹 1의 크기: 7
- 그룹 2의 크기: 1
- 그룹 3의 크기: 7

1	1	0	0	1
0	0	0	1	1
0	1	0	1	0
1	0	1	0	1
0	0	0	0	0

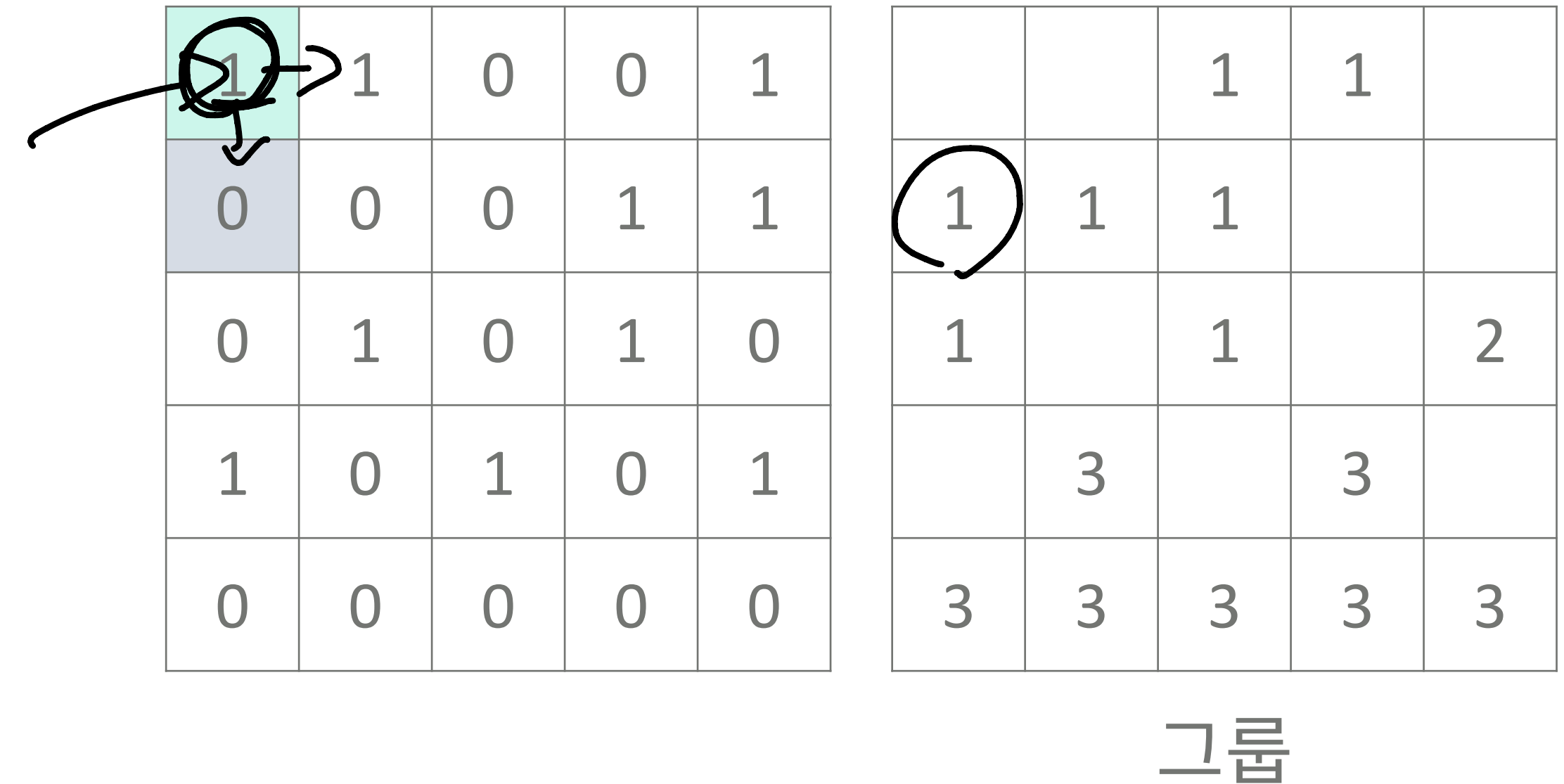
		1	1	
1	1	1		
1		1		2
	3		3	
3	3	3	3	3

그룹

벽 부수고 이동하기 4

<https://www.acmicpc.net/problem/16946>

- 이동할 수 있는 빈 칸을 모두 그룹 짓고
- 몇 개의 칸을 이루어져 있는지 계산해보자
- 그룹 1의 크기: 7
- 그룹 2의 크기: 1
- 그룹 3의 크기: 7
- 근처에 있는 빈 칸의 그룹은 1 이다. 따라서 정답 = 1 + 7 = 8



벽 부수고 이동하기 4

48

<https://www.acmicpc.net/problem/16946>

- 이동할 수 있는 빈 칸을 모두 그룹 짓고
- 몇 개의 칸을 이루어져 있는지 계산해보자
- 그룹 1의 크기: 7
- 그룹 2의 크기: 1
- 그룹 3의 크기: 7
- 근처에 있는 빈 칸의 그룹은 1, 2, 3 이다. 따라서 정답 = $1 + 7 + 1 + 7 = 16$

1	1	0	0	1
0	0	0	1	1
0	1	0	1	0
1	0	1	0	1
0	0	0	0	0

		1	1	
1	1	1		
1		1		2
	3		3	
3	3	3	3	3

그룹

벽 부수고 이동하기 4 $O(NM) + O(NM) = O(NM)$

49

<https://www.acmicpc.net/problem/16946>

- 이동할 수 있는 빈 칸을 모두 그룹 짓고
- 몇 개의 칸을 이루어져 있는지 계산해보자
- 그룹 1의 크기: 7
- 그룹 2의 크기: 1
- 그룹 3의 크기: 7
- 근처에 있는 빈 칸의 그룹은 1, 3 이다. 따라서 정답 = $1 + 7 + 7 = 15$

1	1	0	0	1
0	0	0	1	1
0	1	0	1	0
1	0	1	0	1
0	0	0	0	0

		1	1	
1	1	1		
1	1			2
	3		3	
3	3	3	3	3

그룹

벽 부수고 이동하기 4

50

<https://www.acmicpc.net/problem/16946>

- 소스: <http://codeplus.codes/da35b6bbcfffe4990a3dadd336c62f166>

벽 부수고 이동하기 2

<https://www.acmicpc.net/problem/14442>

- $N \times M$ 의 행렬로 나타내는 지도에서 (1, 1)에서 (N, M)으로 최단 거리로 이동하는 문제
- 0은 빈 칸, 1은 벽
- 단, 벽은 K 번까지 부수고 지나갈 수 있다

$k+1$

$(r, c, 0) \rightarrow$ 벽 \times
 $(r, c, 1) \rightarrow$ 벽 1번
 \vdots
 $(r, c, k) \rightarrow$ 벽 k 번

벽 부수고 이동하기 2

<https://www.acmicpc.net/problem/14442>

- 벽을 부순다는 조건이 없으면 일반적인 미로 탐색 문제이다
- 어떤 칸에 방문했을 때, 벽을 부순 적이 있는 경우와 아직 부순 적이 없는 경우는 다른 경우이기 때문에
- 상태 (i, j) 대신에 (i, j, b) (b 는 벽을 부순 횟수) 으로 BFS 탐색을 진행한다.

벽 부수고 이동하기 2

53

<https://www.acmicpc.net/problem/14442>

- 소스: <http://codeplus.codes/5f295983780f48a6b69f15c19cc87478>

벽 부수고 이동하기 3

<https://www.acmicpc.net/problem/16933>

- $N \times M$ 의 행렬로 나타내는 지도에서 (1, 1)에서 (N, M)으로 최단 거리로 이동하는 문제
- 0은 빈 칸, 1은 벽
- 이동할 때마다 낮과 밤이 바뀐다.
- 단, 벽은 K번까지 부수고 지나갈 수 있고, 낮에만 부술 수 있다.

벽 부수고 이동하기 3

<https://www.acmicpc.net/problem/16933>

- 벽을 부순다는 조건이 없으면 일반적인 미로 탐색 문제이다
- 어떤 칸에 방문했을 때, 벽을 부순 적이 있는 경우와 아직 부순 적이 없는 경우는 다른 경우다.
- 낮과 밤도 다른 경우이다.
- 상태 (i, j) 대신에 (i, j, b, t) (b 는 벽을 부순 횟수, $t = \text{낮/밤}$)으로 BFS 탐색을 진행한다.

벽 부수고 이동하기 3

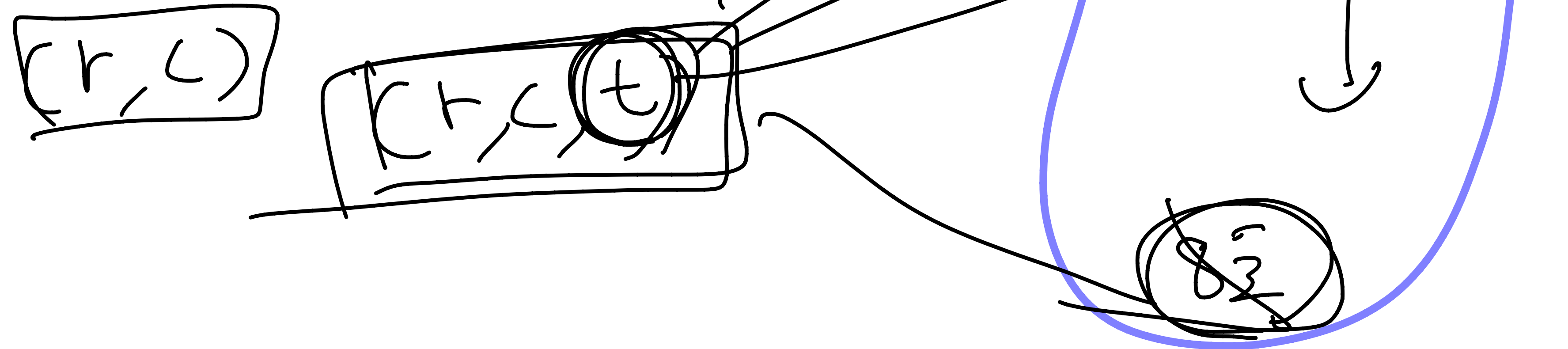
56

<https://www.acmicpc.net/problem/16933>

- 소스: <http://codeplus.codes/46e1b14b1cfa4915ab878124a2207dd7>

7

- 크기가 8×8 인 체스판이 있고, 모든 칸에는 빈 칸 또는 벽이다.
- 가장 왼쪽 아랫 칸에서 가장 오른쪽 윗 칸으로 이동할 수 있는지 없는지 구하는 문제
- 벽은 1초에 한 칸씩 아래로 내려온다.
- 벽이 있는 칸으로 이동할 수 없고, 이동한 칸에 벽이 내려오면 더 이상 이동할 수 없다.



움직이는 미로 탈출

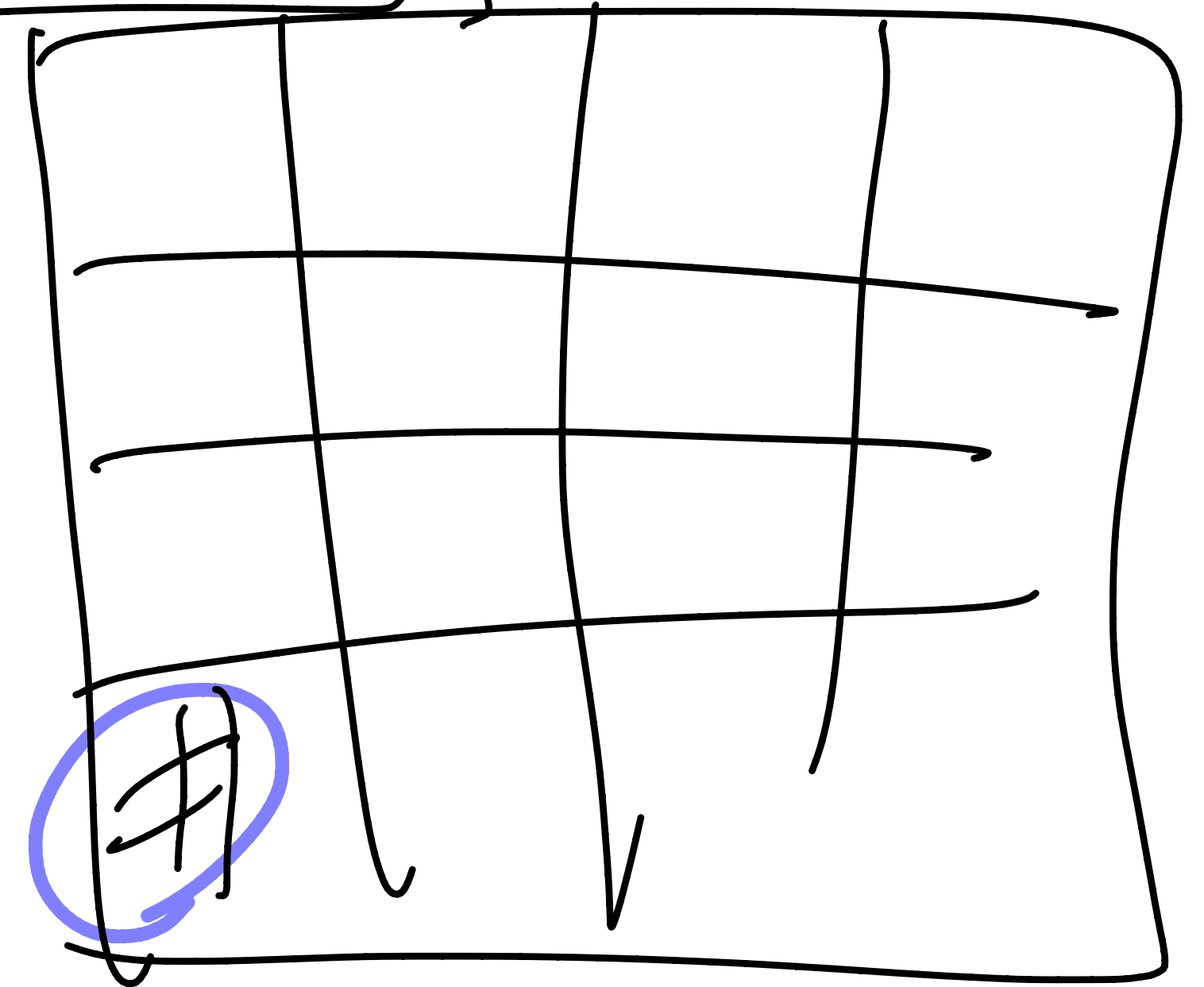
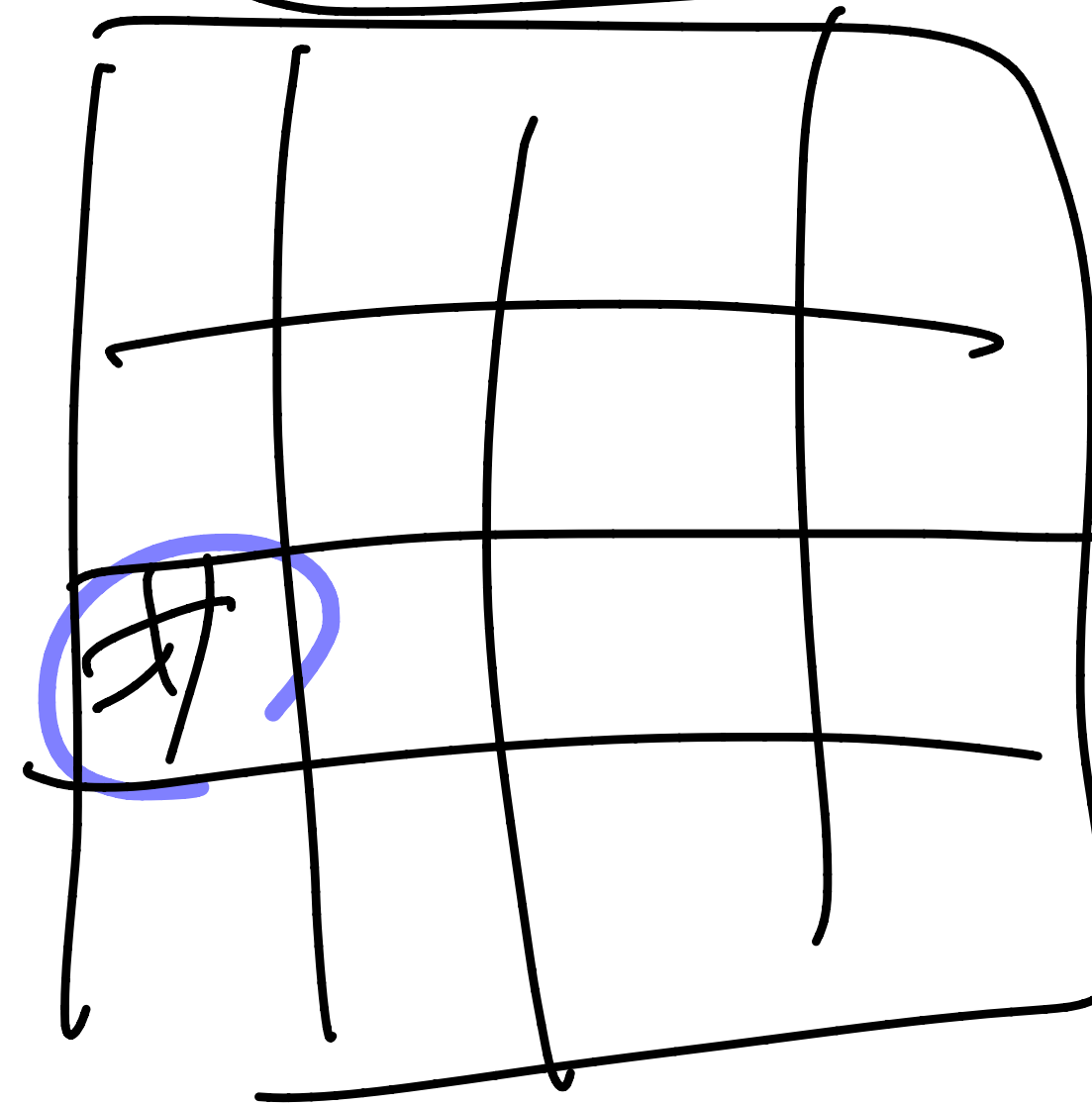
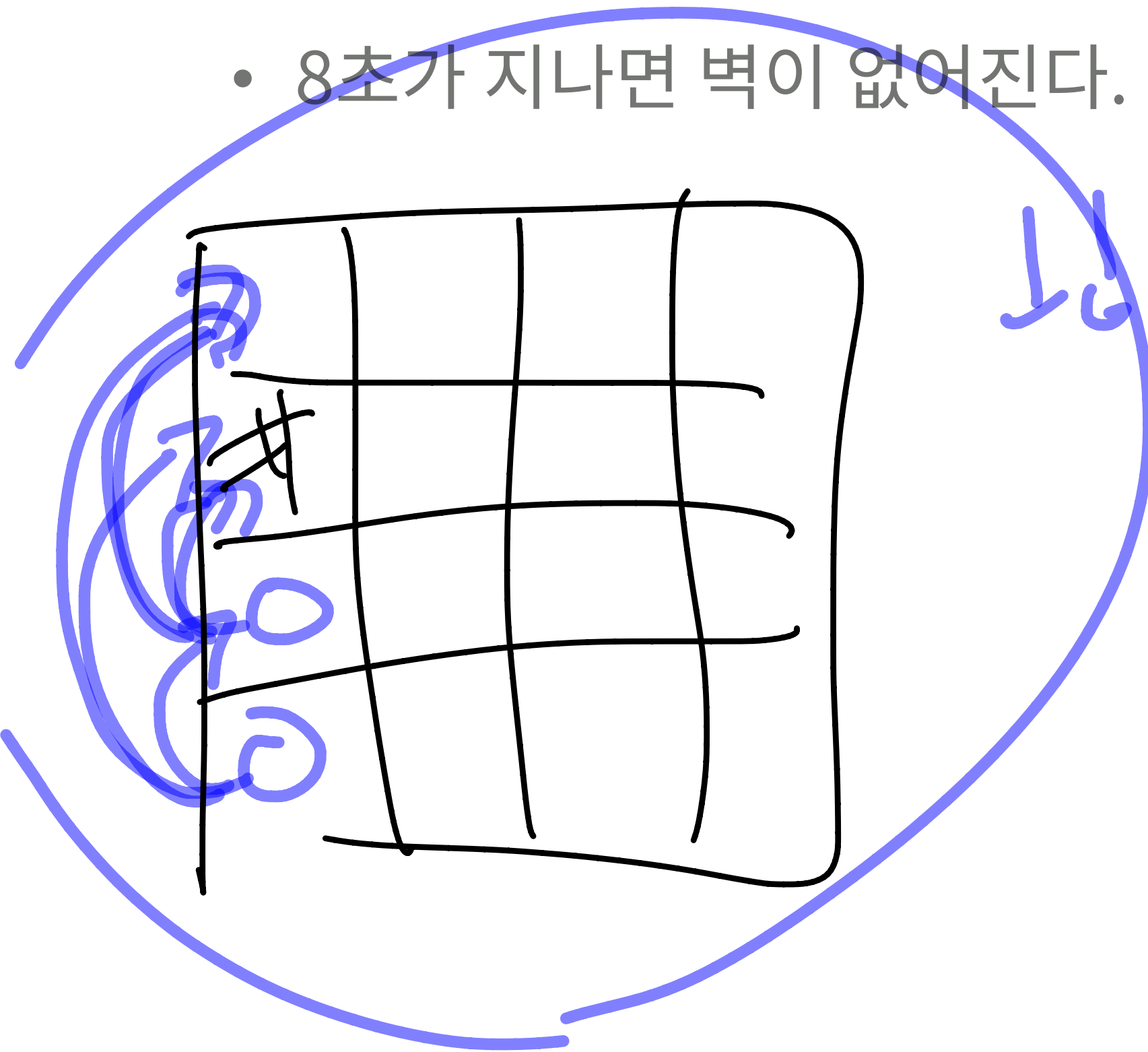
<https://www.acmicpc.net/problem/16954>

- 8초가 지나면 벽이 없어진다.

(r, c)

t 초 후 벽 ?

$$A[r-t, c] = 1, 0$$



움직이는 미로 탈출

<https://www.acmicpc.net/problem/16954>

- 지도를 총 9개 준비해서, 0초 후, 1초 후, 2초 후, ..., 8초 후를 만들고 BFS를 수행할 수 있다.
- (r, c, t) : t 초 후에 (r, c) 에 있을 때 최소 시간
- 8초 후부터는 t 를 증가시키는 의미가 없다.

움직이는 미로 탈출

<https://www.acmicpc.net/problem/16954>

- 실제로는 지도를 9개나 만들 필요는 없다.
- 특정 시점이 t 초 후에 벽이 있는지 없는지는 알아낼 수 있기 때문이다.
- t 초 후에 (r, c) 로 벽이 내려왔다면, 그 벽은 $(r-t, c)$ 에 있던 벽이다.

움직이는 미로 탈출

<https://www.acmicpc.net/problem/16954>

- 소스: <http://codeplus.codes/70104a7579a544baa240d4af789f0943>

탈출

<https://www.acmicpc.net/problem/3055>

- 지도는 R행 C열이다
- 비어있는 곳은 '.'
- 물이 차있는 지역은 '*'
- 돌은 'X'
- 비버의 굴은 'D'
- 고슴도치의 위치는 'S'

탈출

<https://www.acmicpc.net/problem/3055>

- 먼저, 물이 언제 차는지 미리 구해놓은 다음에
- 고슴도치를 그 다음에 이동시킨다

탈출

<https://www.acmicpc.net/problem/3055>

- 지도 상태

.	D	.	*
.	.	.	.
.	.	X	.
S	.	*	.
.	.	.	.

- 물이 차는 시간

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

- 고슴도치의 이동

0			

탈출

65

<https://www.acmicpc.net/problem/3055>

- 지도 상태

.	D	.	*
.	.	.	.
.	.	X	.
S	.	*	.
.	.	.	.

- 물이 차는 시간

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

- 고슴도치의 이동

1			
0			
1			

탈출

<https://www.acmicpc.net/problem/3055>

- 지도 상태
- 물이 차는 시간
- 고슴도치의 이동

.	D	.	*
.	.	.	.
.	.	X	.
S	.	*	.
.	.	.	.

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

2			
1			
0			
1			

탈출

67

<https://www.acmicpc.net/problem/3055>

- 지도 상태

.	D	.	*
.	.	.	.
.	.	X	.
S	.	*	.
.	.	.	.

- 물이 차는 시간

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

- 고슴도치의 이동

3			
2			
1			
0			
1			

탈출

<https://www.acmicpc.net/problem/3055>

- 지도 상태
- 물이 차는 시간
- 고슴도치의 이동

.	D	.	*
.	.	.	.
.	.	X	.
S	.	*	.
.	.	.	.

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

3	4		
2			
1			
0			
1			

탈출

<https://www.acmicpc.net/problem/3055>

- 소스: <http://codeplus.codes/e5c5ee82b92147f79d756ad7dfc4c74b>

아기 상어

<https://www.acmicpc.net/problem/16236>

- $N \times N$ 크기의 공간에 물고기 M 마리와 아기 상어 1마리가 있다. $N \leq 20$
- 공간은 1×1 크기의 정사각형 칸으로 나누어져 있다. 한 칸에는 물고기가 최대 1마리 존재한다.
- 아기 상어와 물고기는 모두 크기를 가지고 있다. 아기 상어의 크기는 2, 1초에 상하좌우로 인접한 한 칸씩 이동한다.
- 아기 상어는 자신의 크기보다 큰 물고기가 있는 칸은 지나갈 수 없다. 아기 상어는 자신의 크기보다 작은 물고기만 먹을 수 있다. 크기가 같은 물고기는 먹을 수 없지만, 그 물고기가 있는 칸은 지나갈 수 있다.
- 상어가 이동하는 방법은 다음 페이지에 있다.

아기 상어

<https://www.acmicpc.net/problem/16236>

- 더 이상 먹을 수 있는 물고기가 공간에 없다면 아기 상어는 엄마 상어에게 도움을 요청한다.
- 먹을 수 있는 물고기가 1마리라면, 그 물고기를 먹으러 간다.
- 먹을 수 있는 물고기가 1마리보다 많다면, 거리가 가장 가까운 물고기를 먹으러 간다.
 - 거리는 아기 상어가 있는 칸에서 물고기가 있는 칸으로 이동할 때, 지나야하는 칸의 개수의 최소값이다.
 - 거리가 가까운 물고기가 많다면, 가장 위에 있는 물고기, 그러한 물고기가 여러마리라면, 가장 왼쪽에 있는 물고기를 먹는다.
- 이동은 1초가 걸리고, 먹는데 걸리는 시간은 없다. 물고기를 먹으면 빈 칸이 된다.
- 아기 상어가 자신의 크기와 같은 수의 물고기를 먹을 때마다 크기가 1 증가한다.
- 엄마 상어에게 요청하지 않고, 몇 초 동안 물고기를 잡아먹을 수 있는지 구하는 문제

BFS

아기 상어

<https://www.acmicpc.net/problem/16236>

72

$$N \leq 20$$

- 거리가 가장 가까운 물고기를 찾기 위해 BFS를 이용해야 한다.

$$\text{BFS: } O(N^2)$$

$$\underbrace{N^2} \times \underline{N^2} = O(N^4)$$

$$20^4 = \boxed{160000}$$

아기 상어

<https://www.acmicpc.net/problem/16236>

- BFS 한 번의 시간 복잡도는 $O(N^2)$ 이다.
- 만약, 모든 칸에 물고기가 있고, 상어가 이 물고기를 다 먹을 수 있다면 최대 N^2 번 BFS를 수행한다.
- 총 $O(N^4)$ 이고, $N \leq 20$ 이기 때문에, 시간 안에 해결할 수 있다.

아기 상어

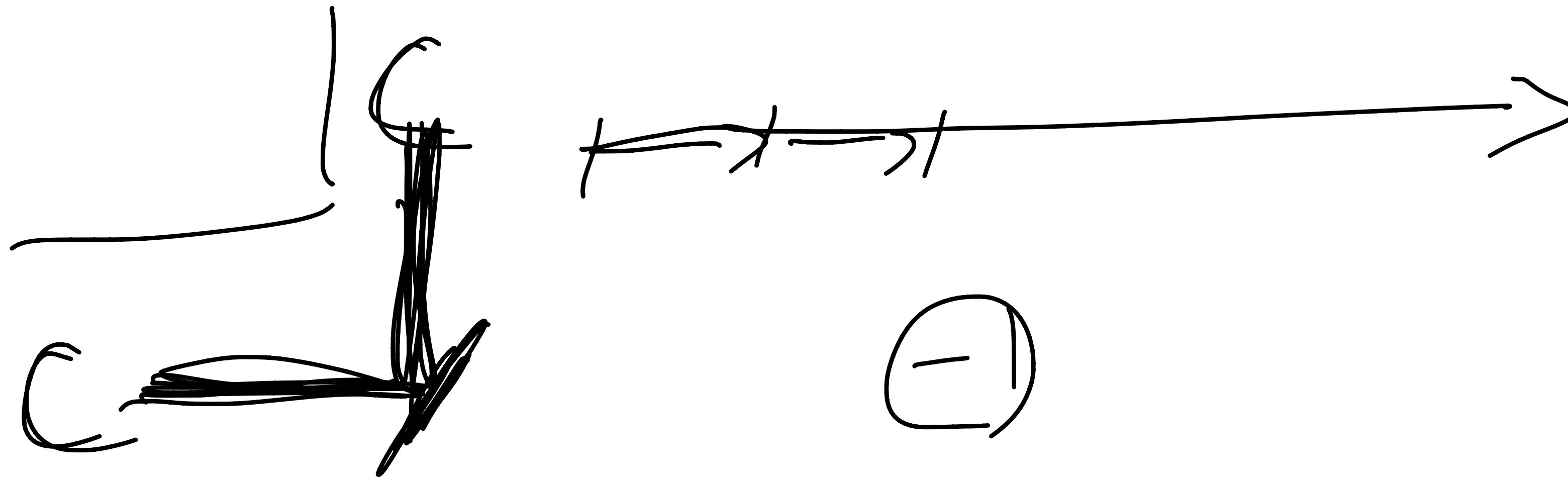
<https://www.acmicpc.net/problem/16236>

- 소스: <http://codeplus.codes/8ee2772a17864f4e8faf294045dac761>

레이저 통신

<https://www.acmicpc.net/problem/6087>

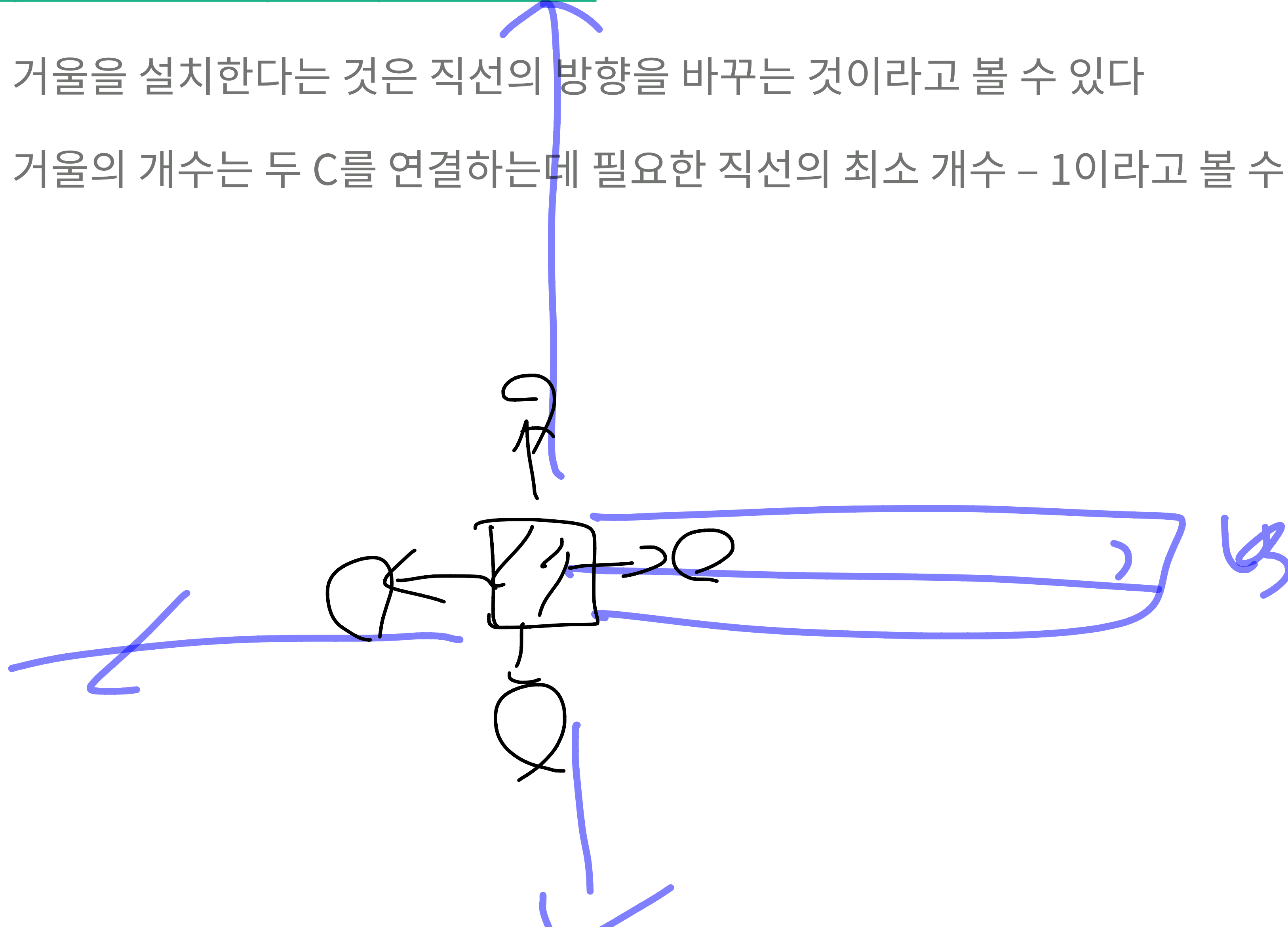
- 크기가 $W \times H$ 인 지도가 주어졌을 때
- 두 C를 레이저로 연결하기 위해서 설치해야 하는 거울 개수의 최소값을 구하는 문제



레이저 통신

<https://www.acmicpc.net/problem/6087>

- 거울을 설치한다는 것은 직선의 방향을 바꾸는 것이라고 볼 수 있다
- 거울의 개수는 두 C를 연결하는데 필요한 직선의 최소 개수 - 1이라고 볼 수 있다.



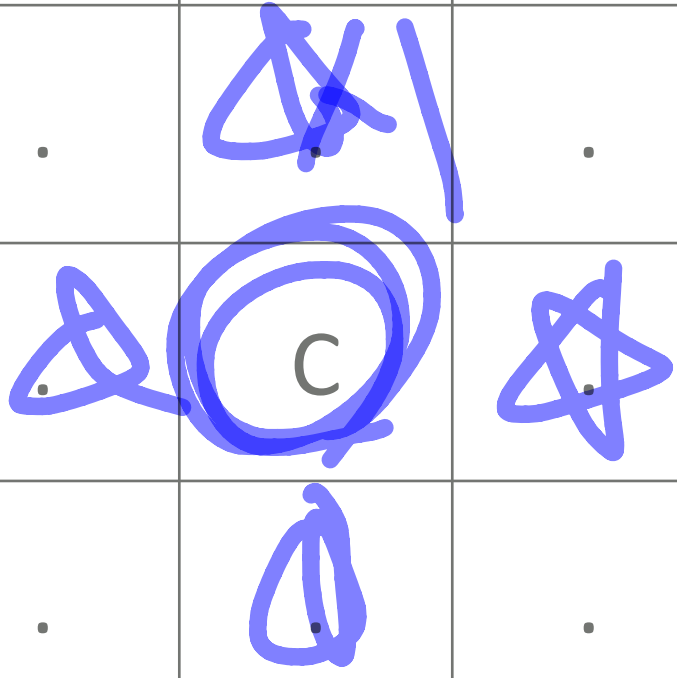

레이저 통신

<https://www.acmicpc.net/problem/6087>

- BFS에서 다음 정점을 인접한 네 방향에 있는 점만 넣는 것이 아니고
- 네 방향에 있는 모든 점을 넣는 방식으로 바뀌서 해결하면 된다.

레이저 통신

<https://www.acmicpc.net/problem/6087>

.
.	C
.	*
*	*	*	*	*	.	*
.	.	.	.	*	.	.
.		.	.	*	.	.
.	C	*	.	*	.	.
.	

-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	0	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1

레이저 통신

<https://www.acmicpc.net/problem/6087>

.
.	C
.	*
*	*	*	*	*	.	*
.	.	.	.	*	.	.
.	.	.	.	*	.	.
.	C	.	.	*	.	.
.

-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	0	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1

레이저 통신

<https://www.acmicpc.net/problem/6087>

.
.	C
.	*
*	*	*	*	*	.	*
.	.	.	.	*	.	.
.	.	.	.	*	.	.
.	C	.	.	*	.	.
.

-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1
1	0	1	1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1

레이저 통신

<https://www.acmicpc.net/problem/6087>

.
.	C
.	*
*	*	*	*	*	.	*
.	.	.	.	*	.	.
.	.	.	.	*	.	.
.	C	.	.	*	.	.
.

-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1
1	0	1	1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1

레이저 통신

<https://www.acmicpc.net/problem/6087>

.
.	C
.	*
*	*	*	*	*	.	*
.	.	.	.	*	.	.
.	.	.	.	*	.	.
.	C	.	.	*	.	.
.

-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1
1	0	1	1	-1	-1	-1
2	1	2	2	2	2	2

레이저 통신

<https://www.acmicpc.net/problem/6087>

- 소스: <http://codeplus.codes/43c4215bc4404bd2a984cbbf32b663b5>

소수 경로

<https://www.acmicpc.net/problem/1963>

- 두 네자리 소수 N과 M이 주어졌을 때
- N을 M으로 바꾸는 최소 변환 횟수를 구하는 문제
- 한 번에 N에서 한 자리만 바꿀 수 있고
- 바꾼 숫자도 소수이어야 한다

BFS?

루우오그

소수 경로

85

<https://www.acmicpc.net/problem/1963>

- $N = \underline{1033}, M = \underline{8179}$
- $N \rightarrow M$
- 1033 1733 3733 3739 3779 8779 8179

소수 경로

<https://www.acmicpc.net/problem/1963>

- 소스: <http://codeplus.codes/f31d7b7f26864ffa96caa53503d8c68a>

적록색약

④

87

<https://www.acmicpc.net/problem/10026>

- $N \times N$ 크기의 격자가 있고, 각 칸에는 R, G, B 중 하나의 색이 색칠되어져 있다.
- 같은 색상이 인접하는 경우 두 구역은 같은 그림이다.
- 적록색약인 사람은 빨간색과 초록색의 차이를 느끼지 못한다.

RRRB
GGBB
BBBR
BBRR
RRRR

- 적록색약인 사람이 보면 구역: 3개, 아닌 사람이 보면 4개
- $1 \leq N \leq 100$

적록색약

<https://www.acmicpc.net/problem/10026>

- BFS를 일반 사람이 봤을 때와 적록색약이 봤을 때로 나누어서 두 번 구현하면 된다.

적록색약

<https://www.acmicpc.net/problem/10026>

- 소스: <http://codeplus.codes/b66b00dd97584e9cb51474bbaeb938b3>

4 연산

<https://www.acmicpc.net/problem/14395>

• 정수 s 의 값을 t 로 바꾸는 최소 연산 횟수를 구하는 문제

• $s = s + s$; (출력: +)

• $s = s - s$; (출력: -)

• $s = s * s$; (출력: *)

• $s = s / s$; (출력: /) (s 가 0이 아닐때만 사용 가능)

• $1 \leq s, t \leq 10^9$

$\hookrightarrow 10^9$

DSLIR

크기 2x2x4

90

A \rightarrow B

DSR

Ok

4 연산

<https://www.acmicpc.net/problem/14395>

- BFS 탐색 문제이다
- 만들어지는 수의 개수는 최대 10^9 개 이다!?

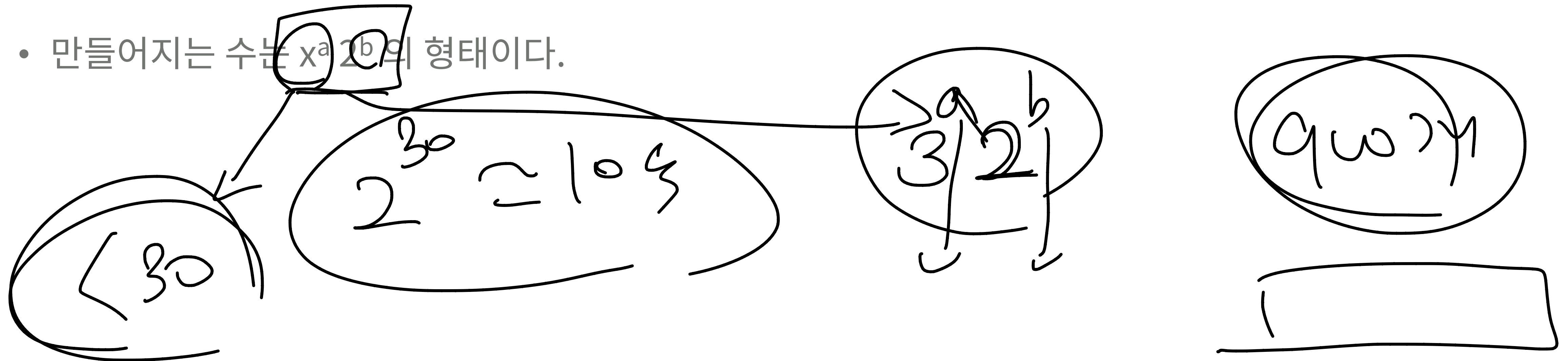
4 연산

<https://www.acmicpc.net/problem/14395>

- BFS 탐색 문제이다
- 만들어지는 수의 개수는 최대 10^9 개 이다!?
- 아니다.
- $x \rightarrow x^2$ 또는 $2 \times x$ 의 형태로 변형만 가능하기 때문에
- 만들어지는 수는 $x^a 2^b$ 의 형태이다.

$$x \rightarrow x + x = 2x$$
$$x \rightarrow x \times x = x^2$$

$$x - x = 0$$
$$x / x = 1 \rightarrow 2$$



4 연산

<https://www.acmicpc.net/problem/14395>

- 소스: <http://codeplus.codes/3ee9946006ed4471836af613ec1cdc9b>