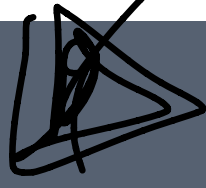


BFS

최백준 choi@startlink.io

문제 상황 \Rightarrow 그래프

가선의 가중치 1



BFS

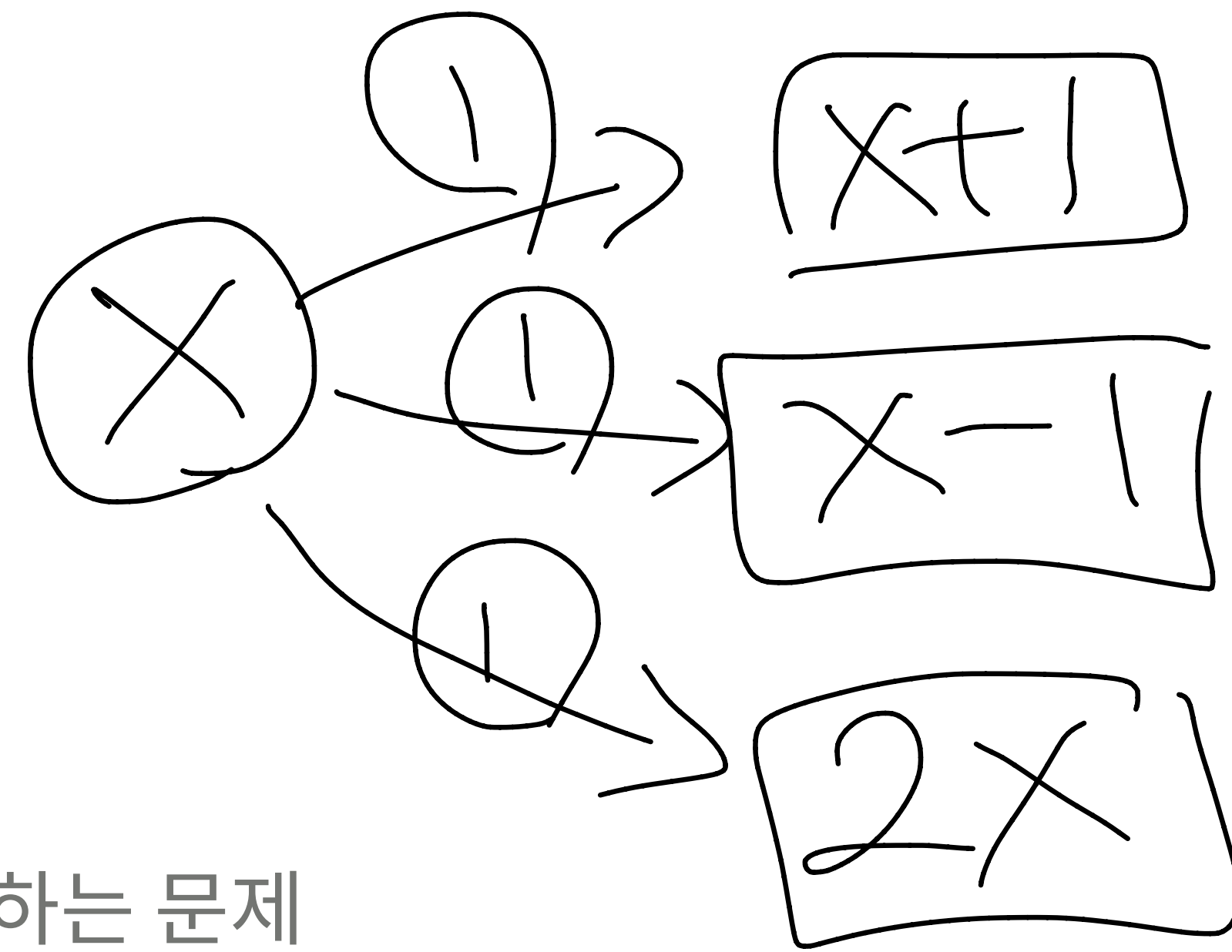
숨바꼭질 4

BFS

3

<https://www.acmicpc.net/problem/13913>

- 수빈이의 위치: N
- 동생의 위치: K
- 동생을 찾는 가장 빠른 시간과 이동하는 방법을 구하는 문제



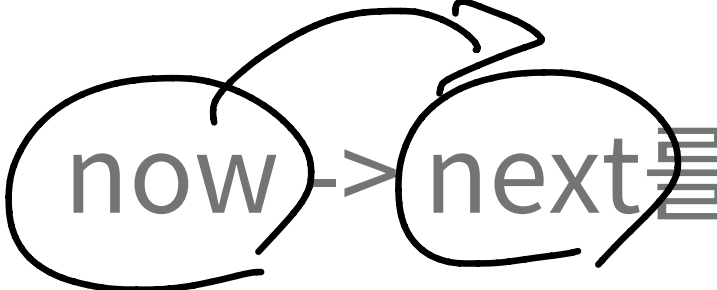
- 수빈이가 할 수 있는 행동 (위치: X)

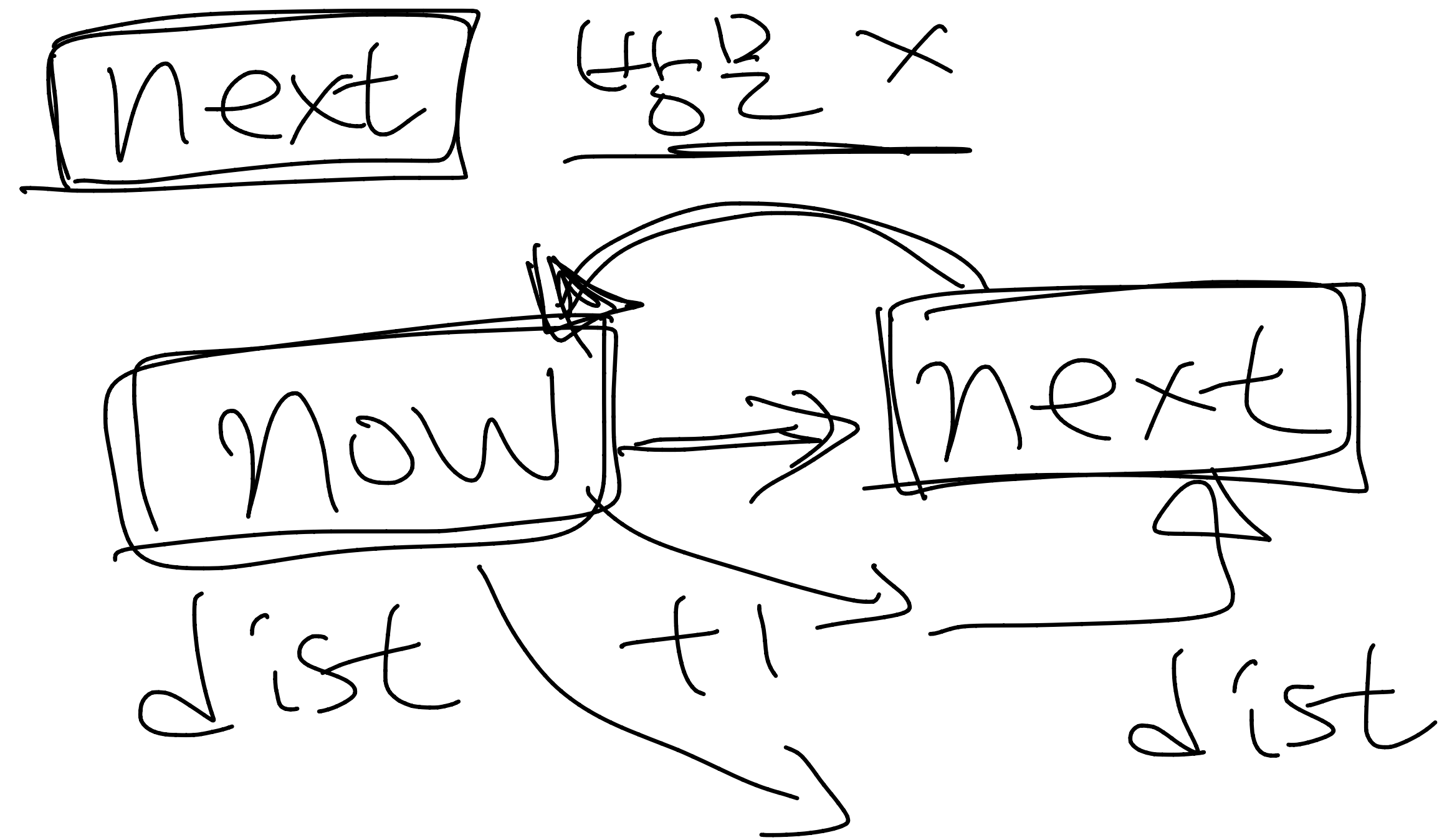
$N \rightarrow K$

1. 걷기: $X+1$ 또는 $X-1$ 로 이동 (1초)
2. 순간이동: $2 \times X$ 로 이동 (1초)

숨바꼭질 4

<https://www.acmicpc.net/problem/13913>

-  `now` → `next`를 갔다고 한다면
`if` (`check[next] == false`) {
 `q.push(next);`
 `check[next] = true;`
 `dist[next] = dist[now] + 1;`
}



숨바꼭질 4

<https://www.acmicpc.net/problem/13913>

- now -> next를 갔다고 한다면

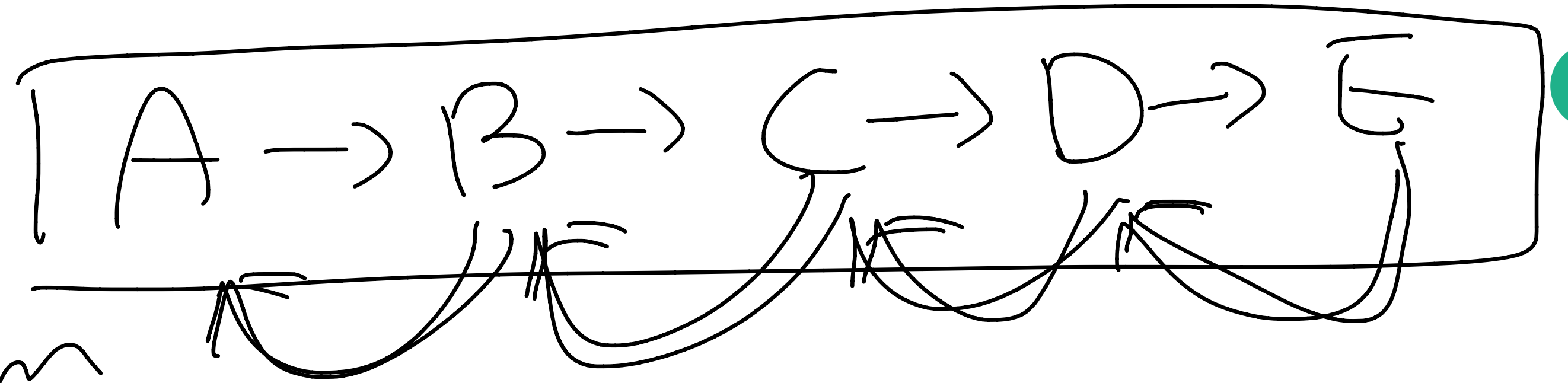
```
if (check[next] == false) {  
    q.push(next);  
    check[next] = true;  
    from[next] = now;  
    dist[next] = dist[now] + 1;  
}
```

숨바꼭질 4

<https://www.acmicpc.net/problem/13913>

- $\text{from}[i]$ = 어디에서 왔는지
- 의미: $\text{from}[i] \rightarrow i$
- N에서 K를 가는 문제 이기 때문에
- K부터 from을 통해서 N까지 가야한다.
- 즉, 역순으로 저장되기 때문에, 다시 역순으로 구하는 것이 필요하다.

from



숨바꼭질 4

<https://www.acmicpc.net/problem/13913>

```
void print(int n, int m) {  
    if (n != m) {  
        print(n, from[m]);  
    }  
    cout << m << ' ';  
}
```

NotKn m을 가는 방법

$n \rightarrow 0 \rightarrow \dots \rightarrow 0 \rightarrow \text{from}[m] \rightarrow m$

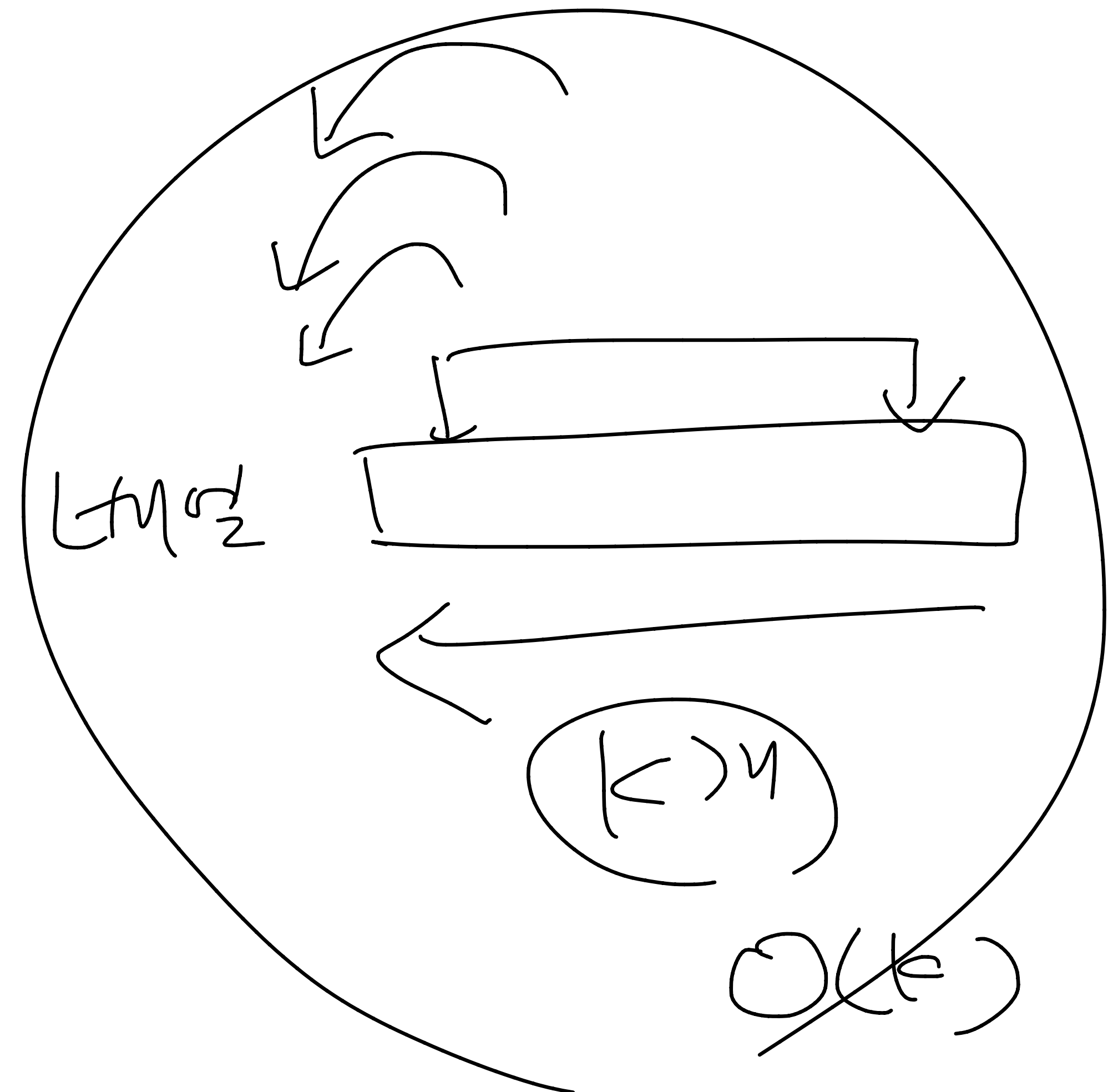
$\Rightarrow \text{print}(n, \text{from}[m]);$

숨바꼭질 4

<https://www.acmicpc.net/problem/13913>

```
stack<int> ans;  
for (int i=m; i!=n; i=from[i]) {  
    ans.push(i);  
}  
ans.push(n);  
while (!ans.empty()) {  
    cout << ans.top() << ' ';  
    ans.pop();  
}  
cout << '\n';
```

$n \Rightarrow m$



숨바꼭질 4

<https://www.acmicpc.net/problem/13913>

- 소스: <http://codeplus.codes/f09c4af1265c4b1a907db5a8f6926b29>

DSL R

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

- 네 자리 숫자 A와 B가 주어졌을 때
- A → B로 바꾸는 최소 연산 횟수

- D: $N \rightarrow 2 * N$ 0/10000
- S: $N \rightarrow N - 1$ 1999
- L: 한 자리씩 왼쪽으로
- R: 한 자리씩 오른쪽으로

BFS

7272 7612

1234 → 2468
1234 → 1233
1234 → 2341
1234 → 4123

0000 - 9999

DSL R

<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
- how[i] = i를 어떻게 만들었는지



DSL R

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

- 1 -> 9을 만드는 경우
- 1~10까지만 있다고 가정

`dist[1] = ?, from[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

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

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

`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

14

<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

`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

16

<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`

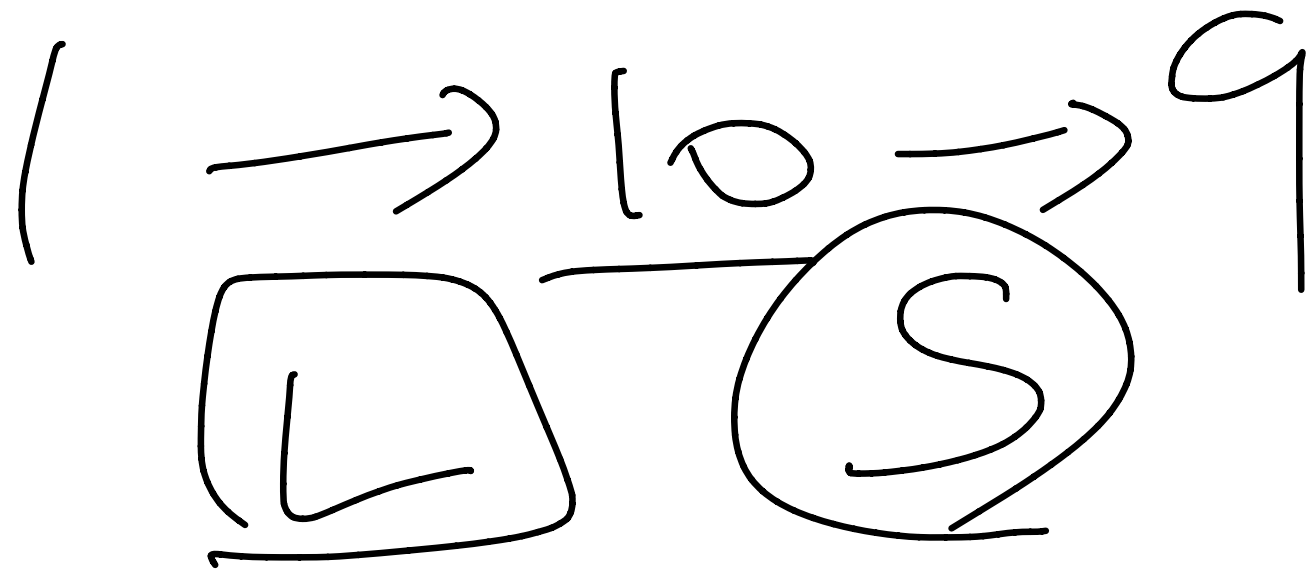
`dist[9] = 2, from[9] = 10, how[9] = S`

`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

<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`

DSL R

<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

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

D

25

```
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';  
}
```

DSL R

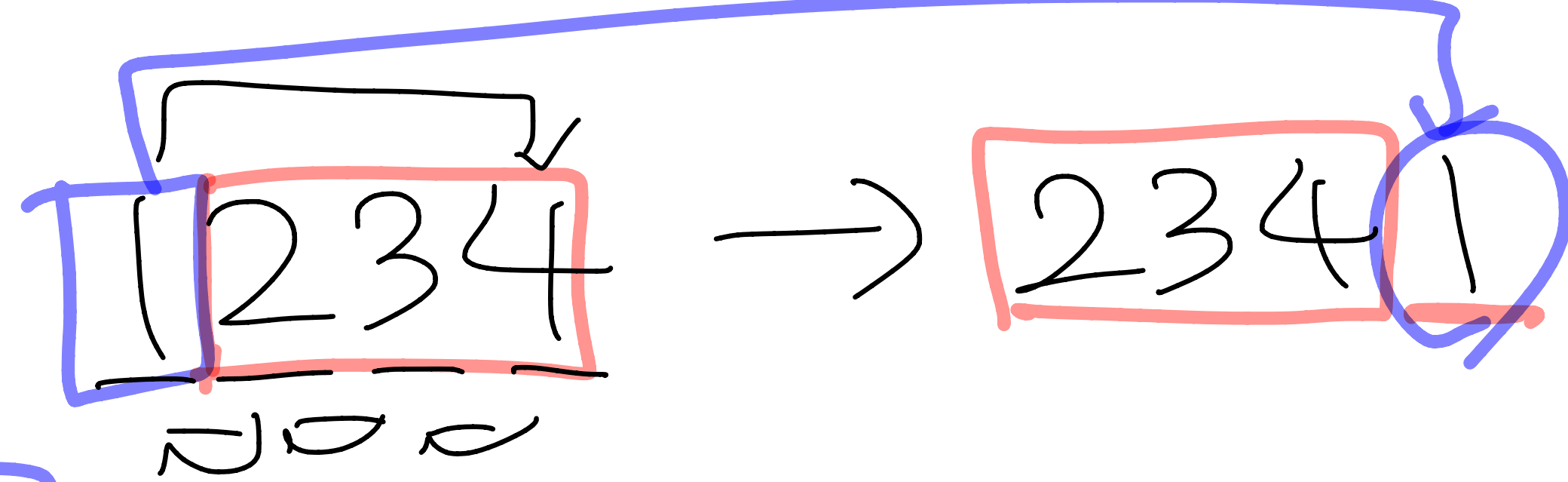
<https://www.acmicpc.net/problem/9019>

```
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 R

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

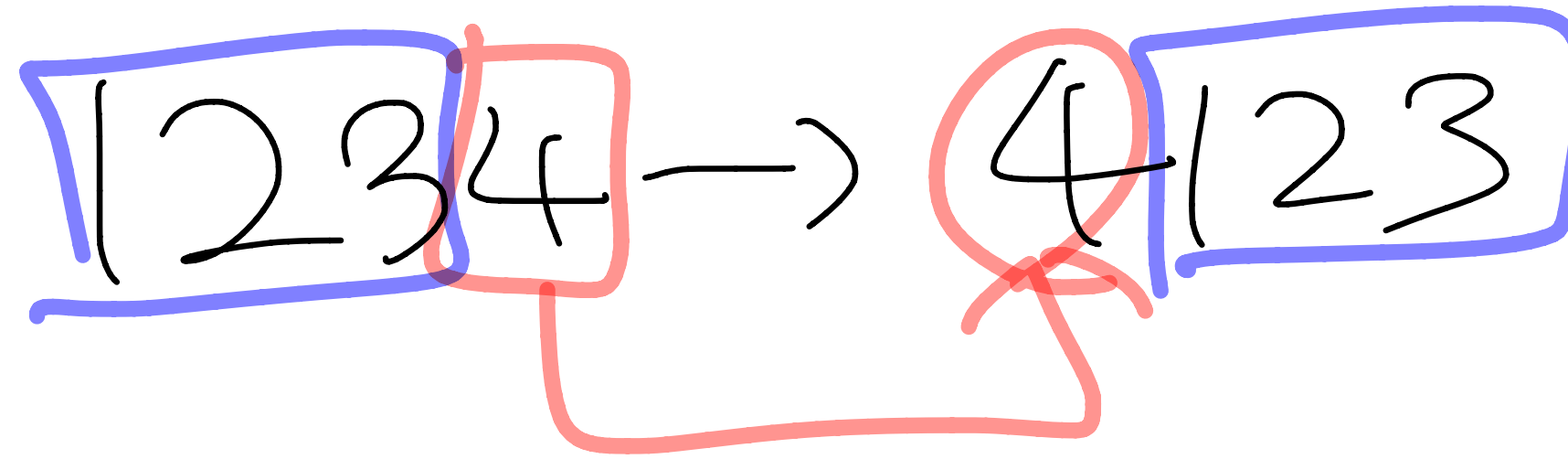
```
next = (now%1000)*10 + now/1000;  
if (check[next] == false) {  
    q.push(next);  
    check[next] = true;  
    dist[next] = dist[now]+1;  
    from[next] = now;  
    how[next] = 'L';  
}
```



DSL R

<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';  
}
```



DSL R

<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';
```

DSL

<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
- how[i] = i를 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>

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

10000²

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

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

- 소스: <http://codeplus.codes/17986c10103f4058989582f8a9c47cbc>

$$4 \text{ 바이트} \times 10^9$$

$$4 \times 10^9 \text{ 바이트}$$

$$4 \times \frac{10^9}{2} = 4 \times 10^8$$

퍼즐

<https://www.acmicpc.net/problem/1525>

- 8퍼즐을 푸는 문제

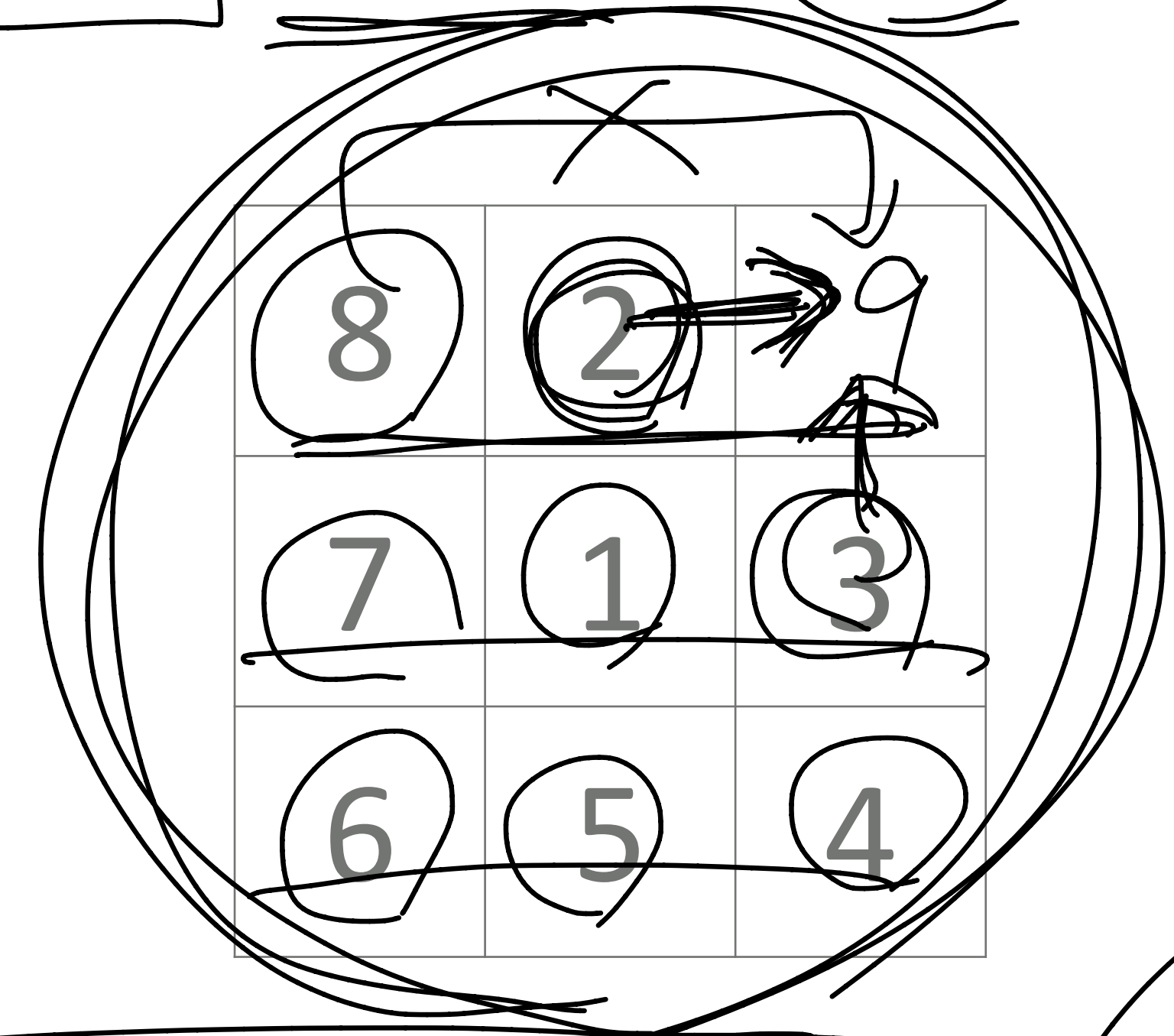
최소 연산

3x3

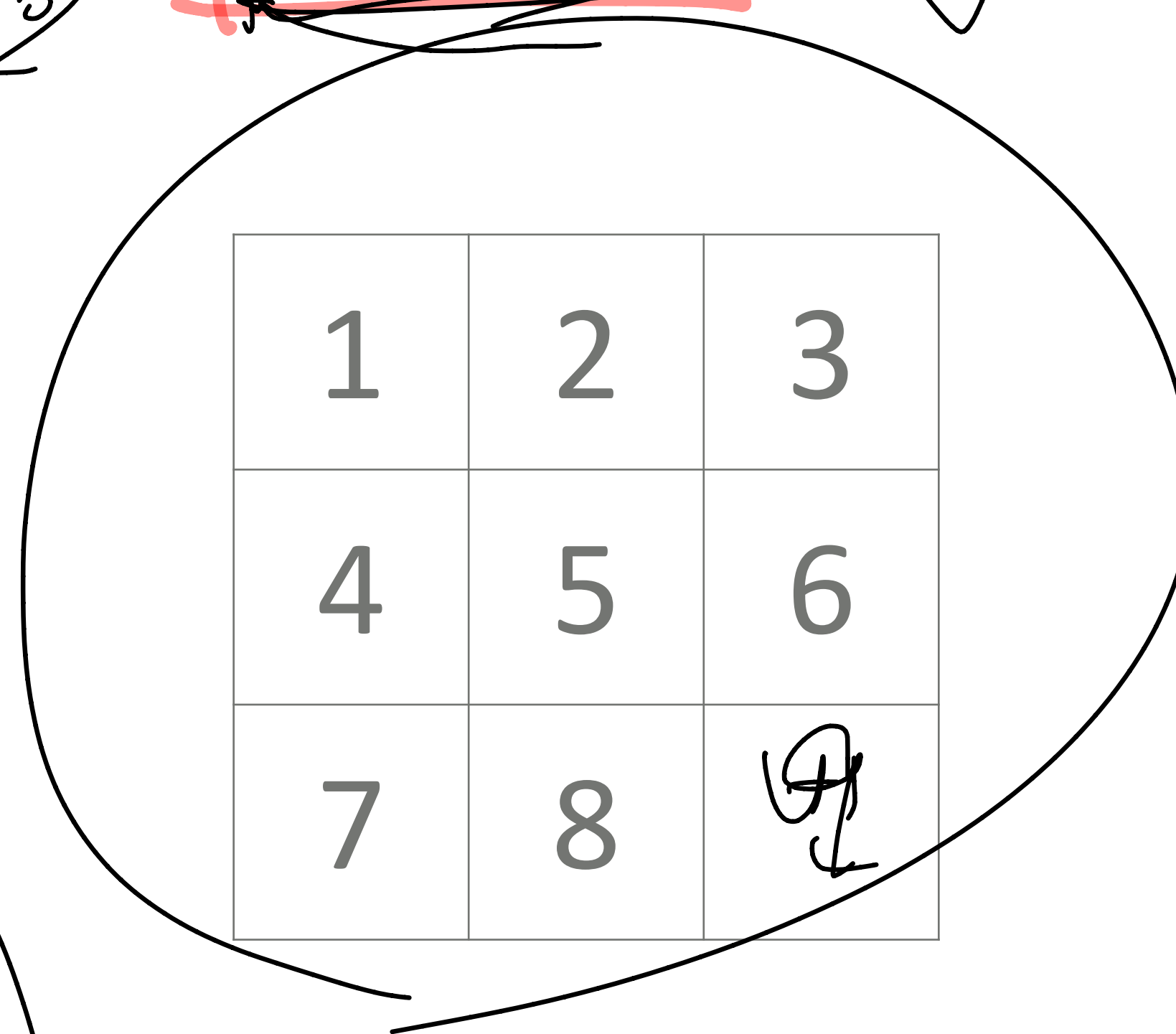
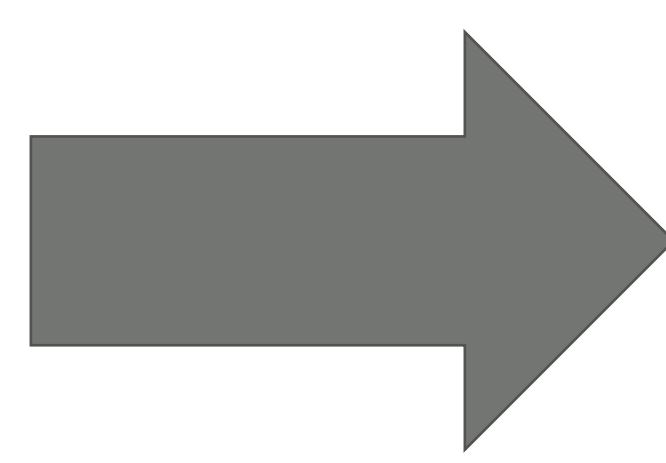
1~8

9! = 362880

2872
↓



1~9



829713654

1 2 3
4 5 6
8 7

Int

1,000,000,000

퍼즐

<https://www.acmicpc.net/problem/1525>

- 8퍼즐을 푸는 문제
- 총 퍼즐 상태의 개수는 $9! = 362,880$ 가지 이다

퍼즐

<https://www.acmicpc.net/problem/1525>

- 8퍼즐을 푸는 문제
- 총 퍼즐 상태의 개수는 $9! = 362,880$ 가지 이다
- 하지만, 상태를 나타내는 수가 9개이기 때문에 배열에 저장할 수는 없다

퍼즐

37

<https://www.acmicpc.net/problem/1525>

- 상태를 저장하는 방법
- 같은 수가 없기 때문에, 순열로 생각해서 몇 번째 순열인지를 저장하는 방법
 - 1727번 문제 응용
- map을 이용해서 저장하기
 - `map<vector<int>,int>`
 - `map<string,int>`
 - `map<int,int>`

AC [5]
문리역
6403

퍼즐

<https://www.acmicpc.net/problem/1525>

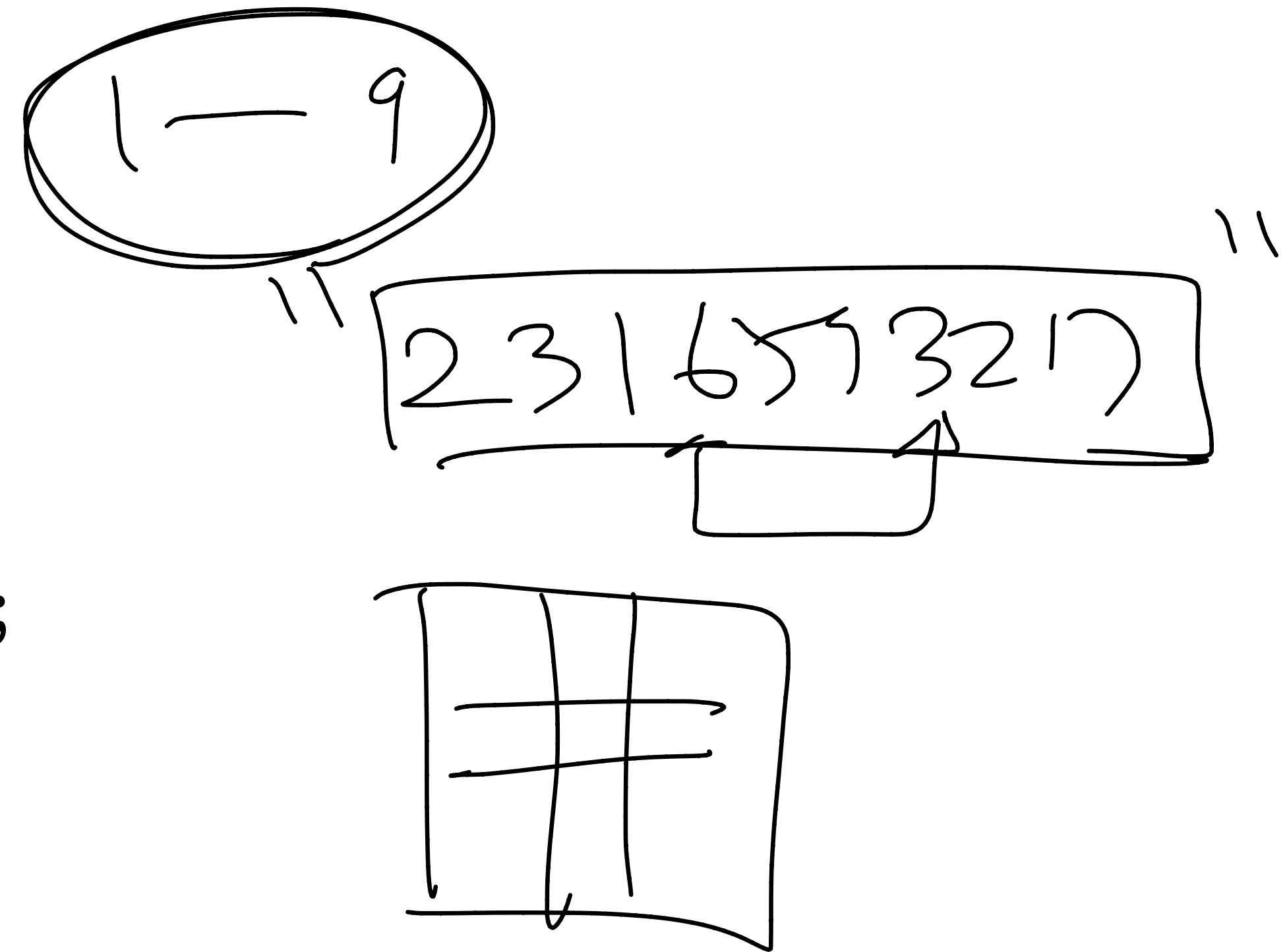
- 0을 9로 바꾸면, 항상 9자리 숫자가 나오기 때문에, 이를 이용해서 문제를 풀 수 있다

퍼즐

39

<https://www.acmicpc.net/problem/1525>

```
queue<int> q; q.push(start);  
map<int, int> d; d[start] = 0;  
while (!q.empty()) {  
    int now_num = q.front();  
    string now = to_string(now_num);  
    q.pop();  
    int z = now.find('9');  
    int x = z/3;  
    int y = z%3;  
    // 다음 페이지  
}
```



퍼즐

<https://www.acmicpc.net/problem/1525>

```
for (int k=0; k<4; k++) {
    int nx = x+dx[k];
    int ny = y+dy[k];
    if (nx >= 0 && nx < n && ny >= 0 && ny < n) {
        string next = now;
        swap(next[x*3+y], next[nx*3+ny]);
        int num = stoi(next);
        if (d.count(num) == 0) {
            d[num] = d[now_num] + 1;
            q.push(num);
        }
    }
}
```

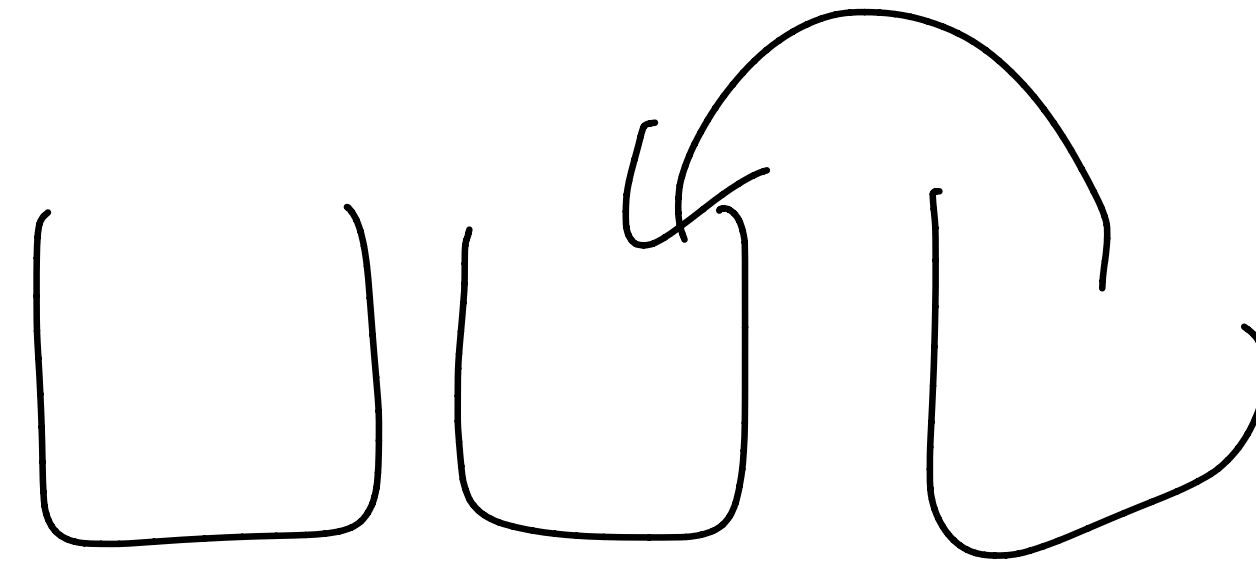

퍼즐

<https://www.acmicpc.net/problem/1525>

- 소스: <http://codeplus.codes/59e5364c39584bb8956504186ee7c717>

물통

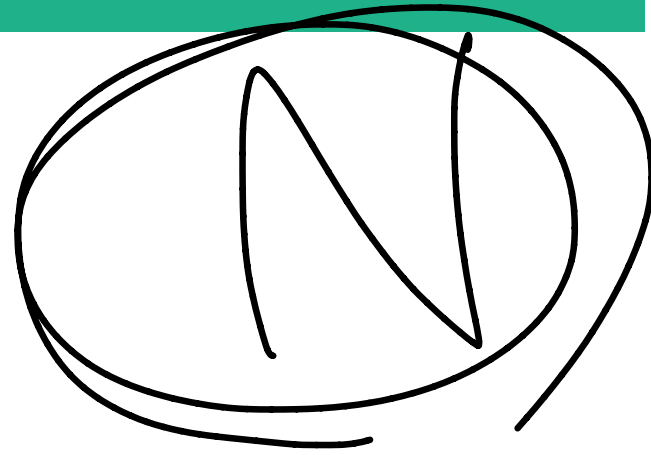
<https://www.acmicpc.net/problem/2251>



- 세 물통 A, B, C가 있을 때
- C만 가득차있다
- 어떤 물통에 들어있는 물을 다른 물통으로 쏟아 부을 수 있는데, 이 때에는 앞의 물통이 빌 때까지 붓거나, 뒤의 물통이 가득 찰 때까지 붓게 된다
- 이 과정에서 손실되는 물은 없다
- 이 때 A가 비어있을 때, C에 들어있을 수 있는 양을 모두 구하는 문제

$ans[i] = true$

물통



<https://www.acmicpc.net/problem/2251>

- 3차원 배열을 만들 필요는 없다
- 중간에 물이 손실되지 않기 때문에
- 첫 번째 물통, 두 번째 물통에 들어있는 물의 양만 알면 세 번째 물통에 들어있는 물의 양을 알 수 있다



<https://www.acmicpc.net/problem/2251>

```
queue<pair<int,int>> q;
q.push(make_pair(0, 0)); check[0][0] = true; ans[c] = true;
while (!q.empty()) {
    int x = q.front().first, y = q.front().second;
    int z = sum - x - y;
    q.pop();
    // x -> y
    // x -> z
    // y -> x
    // y -> z
    // z -> x
    // z -> y
}
```

<https://www.acmicpc.net/problem/2251>

```
// x -> y
ny += nx; nx = 0;
if (ny >= b) {
    nx = ny-b;
    ny = b;
}
if (!check[nx][ny]) {
    check[nx][ny] = true;
    q.push(make_pair(nx,ny));
    if (nx == 0) {
        ans[nz] = true;
    }
}
```

물통

<https://www.acmicpc.net/problem/2251>

- 소스: <http://codeplus.codes/cb03503c5f0c4ca087dbe87728b61068>

숨바꼭질 2

<https://www.acmicpc.net/problem/12851>

47

- 수빈이의 위치: N

- 동생의 위치: K

- 동생을 찾는 가장 빠른 시간을 구하는 문제, 그리고 그러한 방법의 개수도 구해야 한다

- 수빈이가 할 수 있는 행동 (위치: X)

1. 걷기: $X+1$ 또는 $X-1$ 로 이동 (1초)

2. 순간이동: $2*X$ 로 이동 (1초)

BFS

5 → 11

5 → 10 → 9 → 18 → 11

5 → 4 → 8 → 16 → 11

DP

숨바꼭질 2

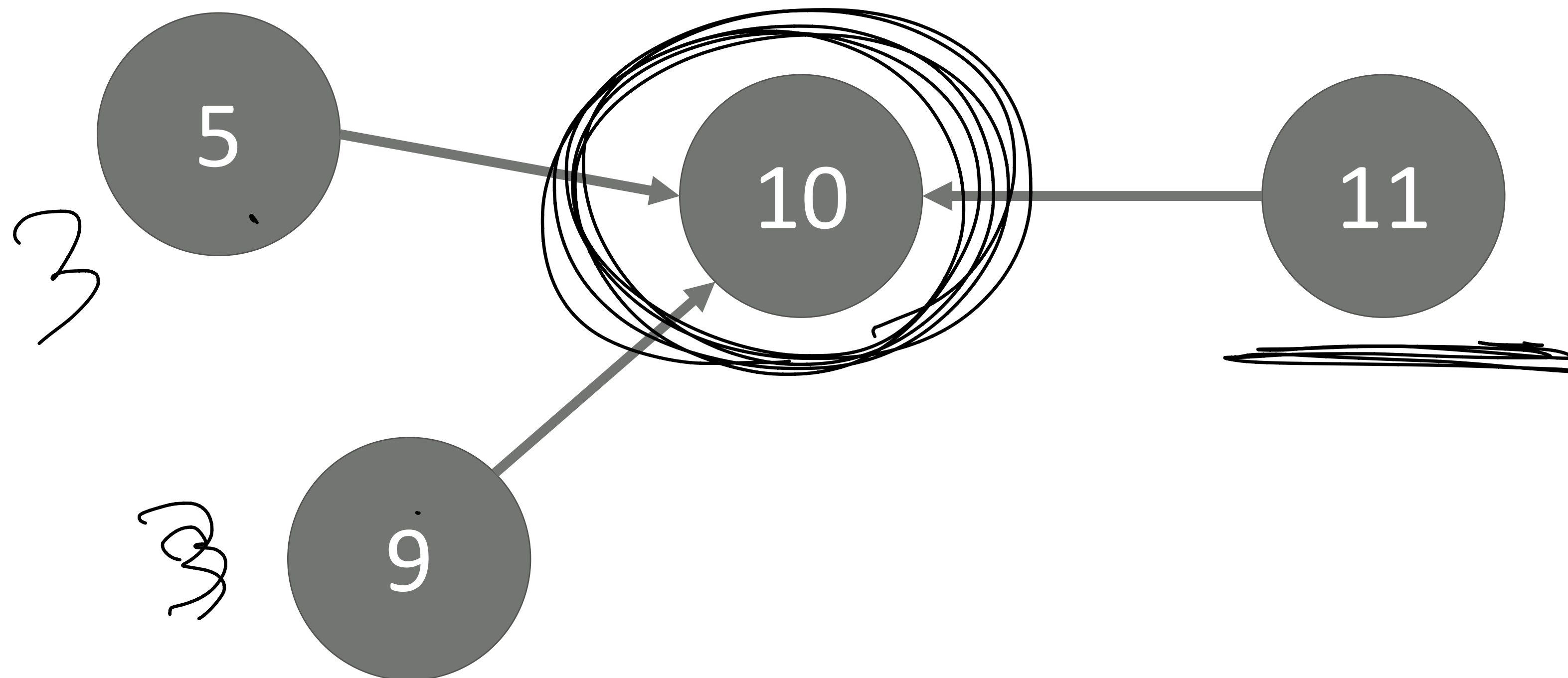
<https://www.acmicpc.net/problem/12851>

- 경우의 수는 다이나믹 프로그래밍으로 구할 수 있다
- cnt[i] = i 까지 가는 방법의 개수

숨바꼭질 2

<https://www.acmicpc.net/problem/12851>

- 10을 아직 방문하지 않았고
- 시작점에서 5와 9까지 가는 거리는 3, 11은 아직 방문하지 않았다고 가정하자

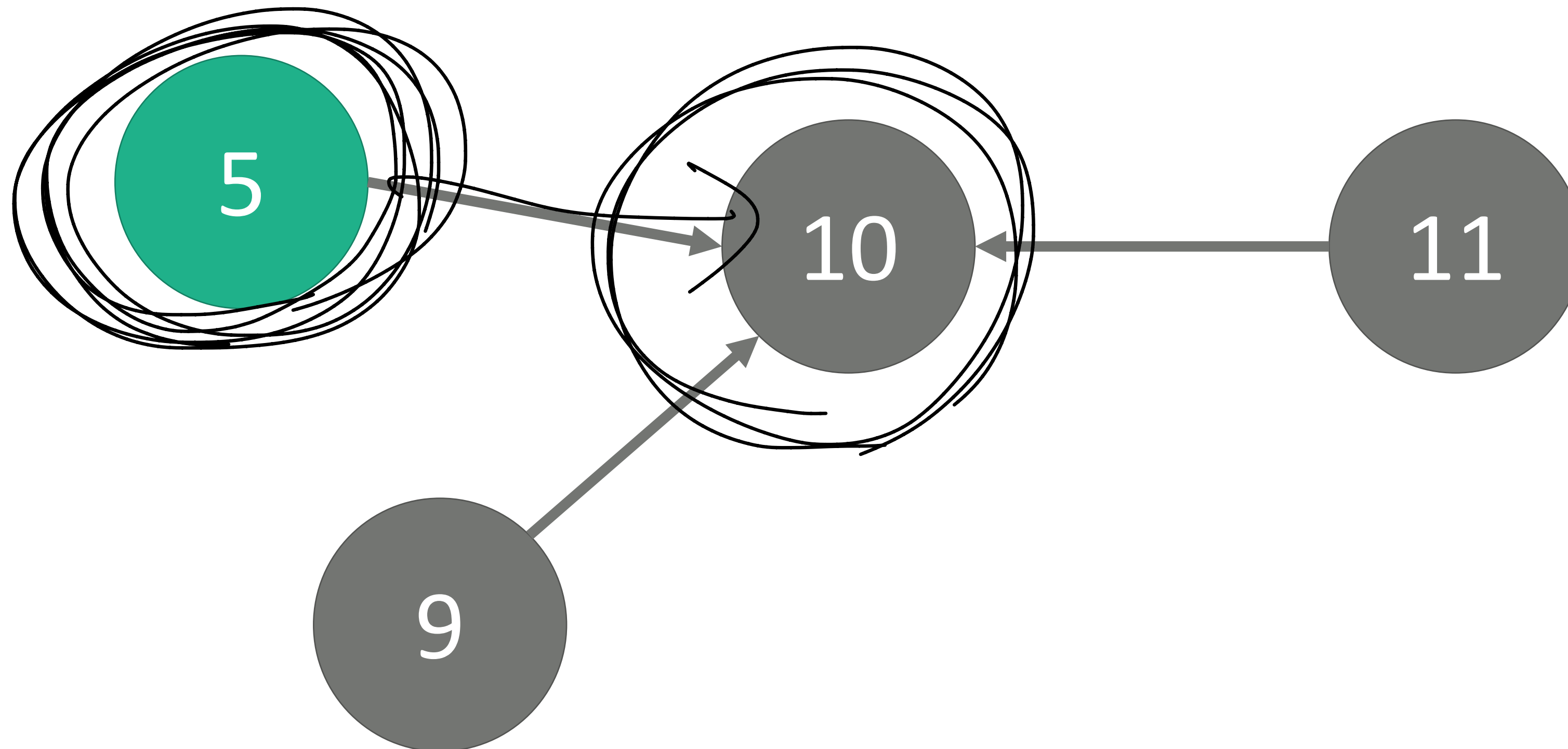


숨바꼭질 2

50

<https://www.acmicpc.net/problem/12851>

- 10은 아직 방문하지 않았기 때문에
- 10을 방문해야 한다.
- 이 때, $\text{cnt}[10] = \text{cnt}[5]$

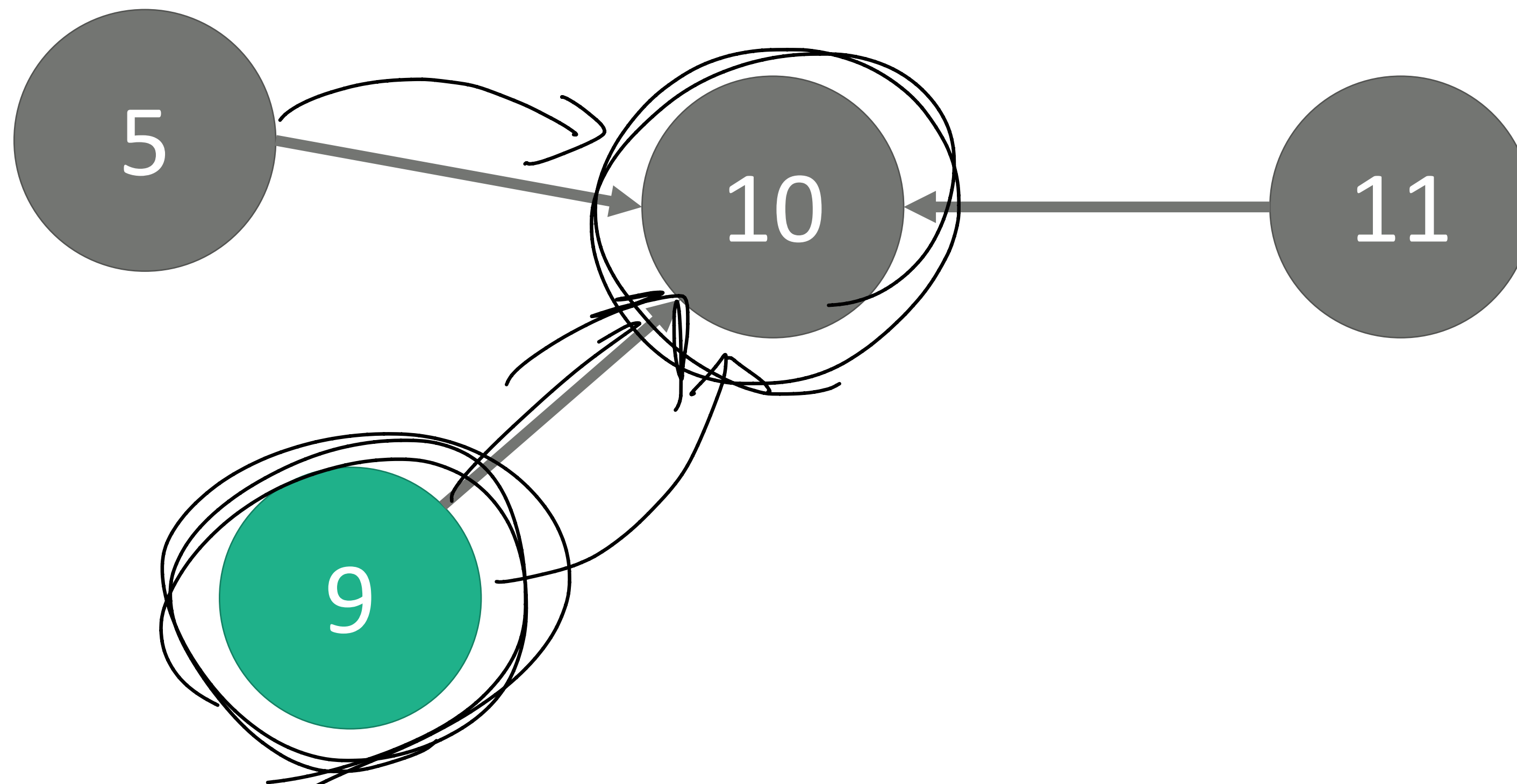


숨바꼭질 2

51

<https://www.acmicpc.net/problem/12851>

- 10은 이미 방문했기 때문에
- 10을 방문할 수는 없다. 하지만, 방법의 수는 증가해야 하기 때문에
- $\text{cnt}[10] += \text{cnt}[9]$

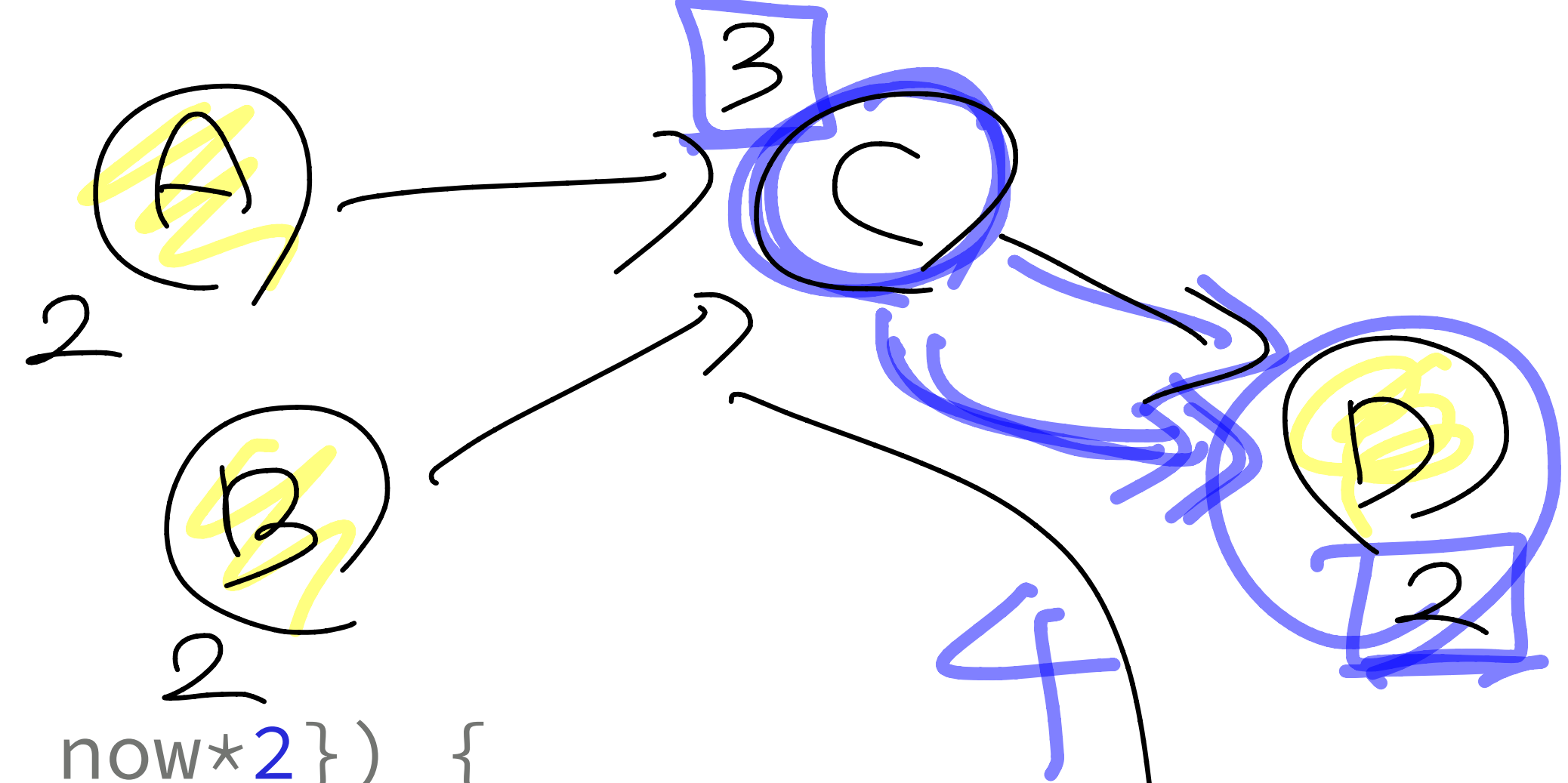


숨바꼭질 2

<https://www.acmicpc.net/problem/12851>

52

```
while (!q.empty()) {  
    int now = q.front(); q.pop();  
    for (int next : {now-1, now+1, now*2}) {  
        if (0 <= next && next <= MAX) {  
            if (check[next] == false) {  
                q.push(next); check[next] = true;  
                dist[next] = dist[now] + 1;  
                cnt[next] = cnt[now];  
            } else if (dist[next] == dist[now] + 1) {  
                cnt[next] += cnt[now];  
            }  
        }  
    }  
}
```



숨바꼭질 2

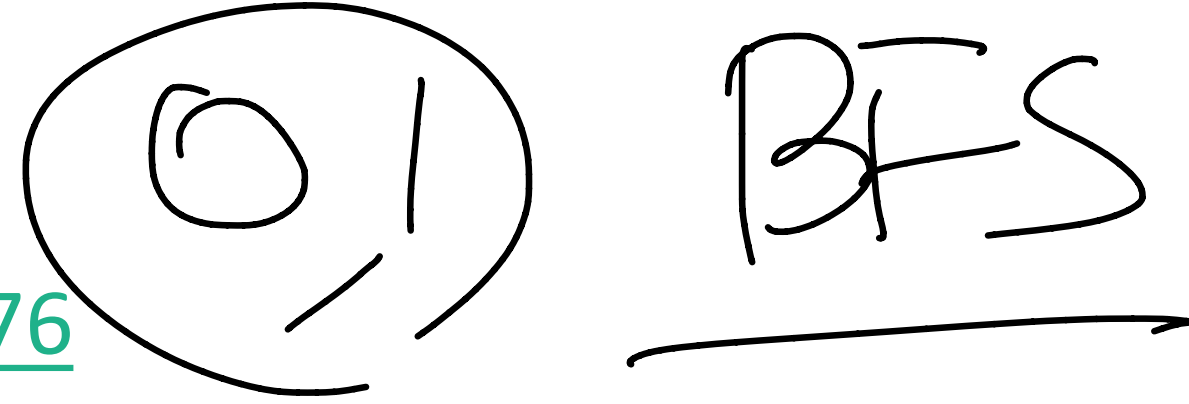
53

<https://www.acmicpc.net/problem/12851>

- 소스: <http://codeplus.codes/ea97d6b4260e4d4bbf87ff78f719835b>

탈옥

<https://www.acmicpc.net/problem/9376>



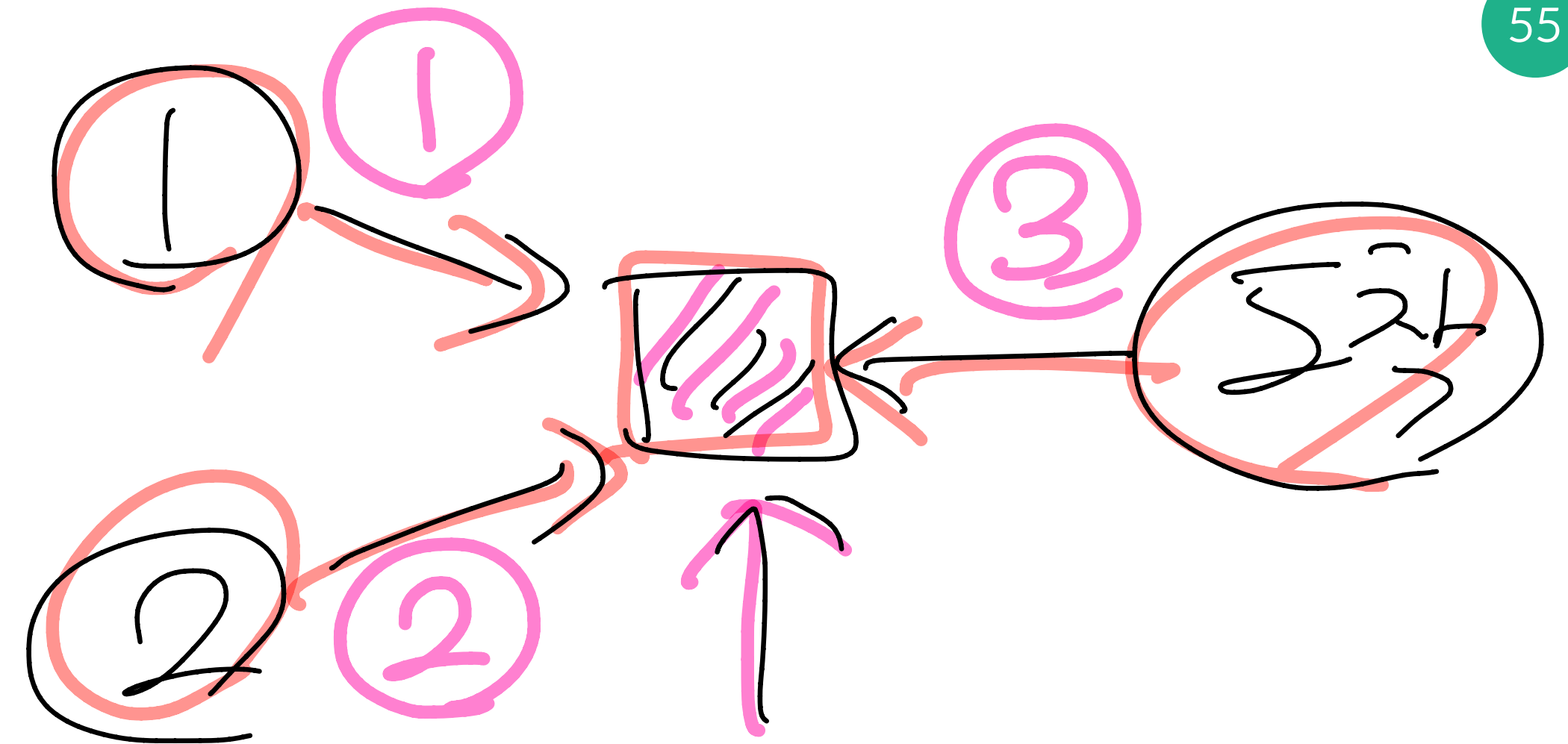
54

- 빈 칸, 벽, 문으로 이루어진 지도가 주어진다.
- 두 죄수가 탈옥하기 위해서 열어야 하는 문의 최소 개수를 구하는 문제

탈옥

<https://www.acmicpc.net/problem/9376>

- 두 지도를 상하좌우로 한 칸씩 확장하면
- 두 죄수의 탈옥 경로는
- 어딘가에서 만나서 함께 이동하는 꼴이 된다
- 따라서, 지도의 밖에서 BFS 1번, 각 죄수별로 1번씩 BFS를 수행한다.
- 그 다음, 정답을 합친다
- 이 때, 문에서 만나는 경우는 조심해야 한다



탈옥

연이어=하는 문의 최소

56

<https://www.acmicpc.net/problem/9376>

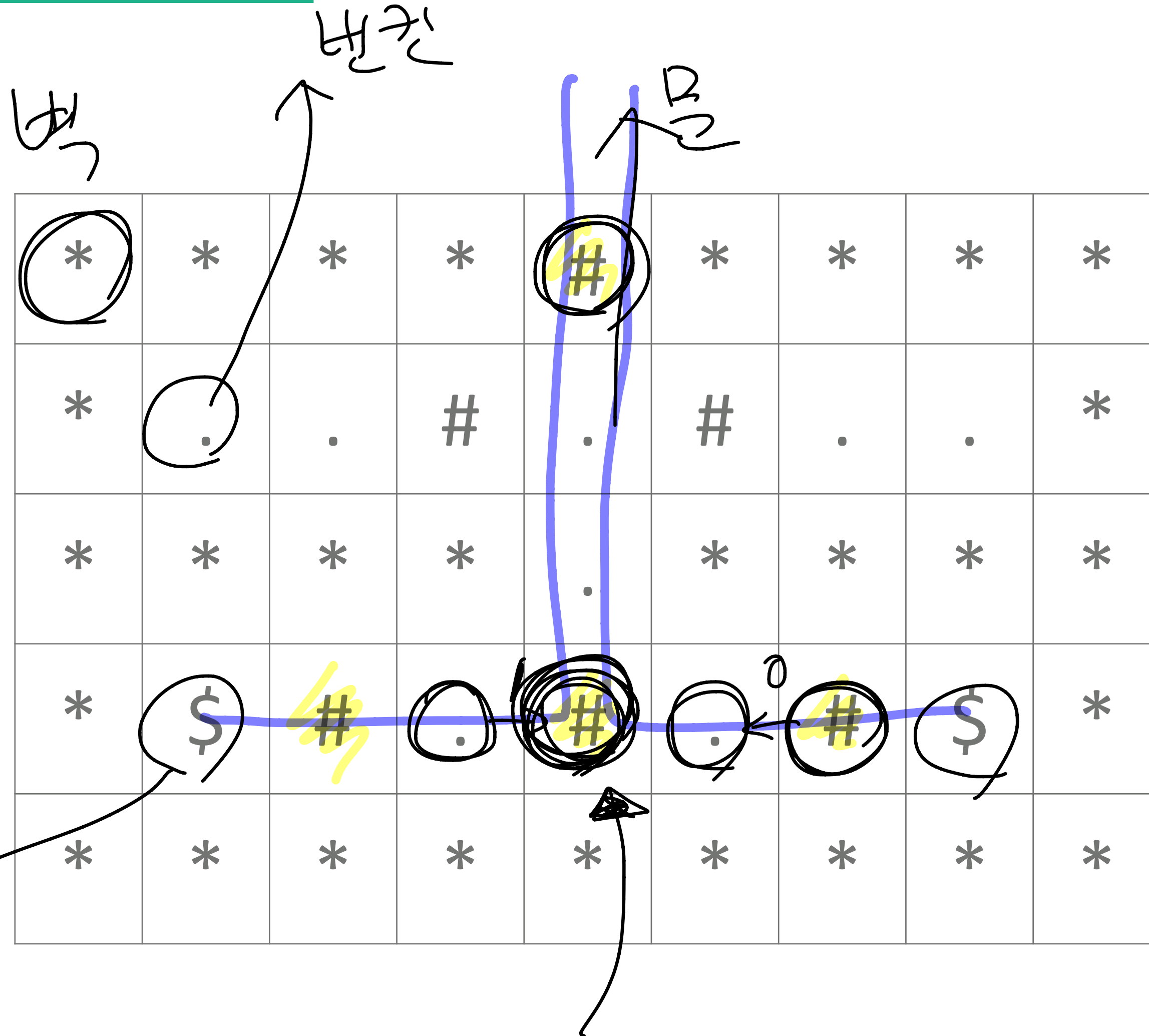
가장자리 0, 1

BFS

Time

Deque

최소



그리프

바퀴

12

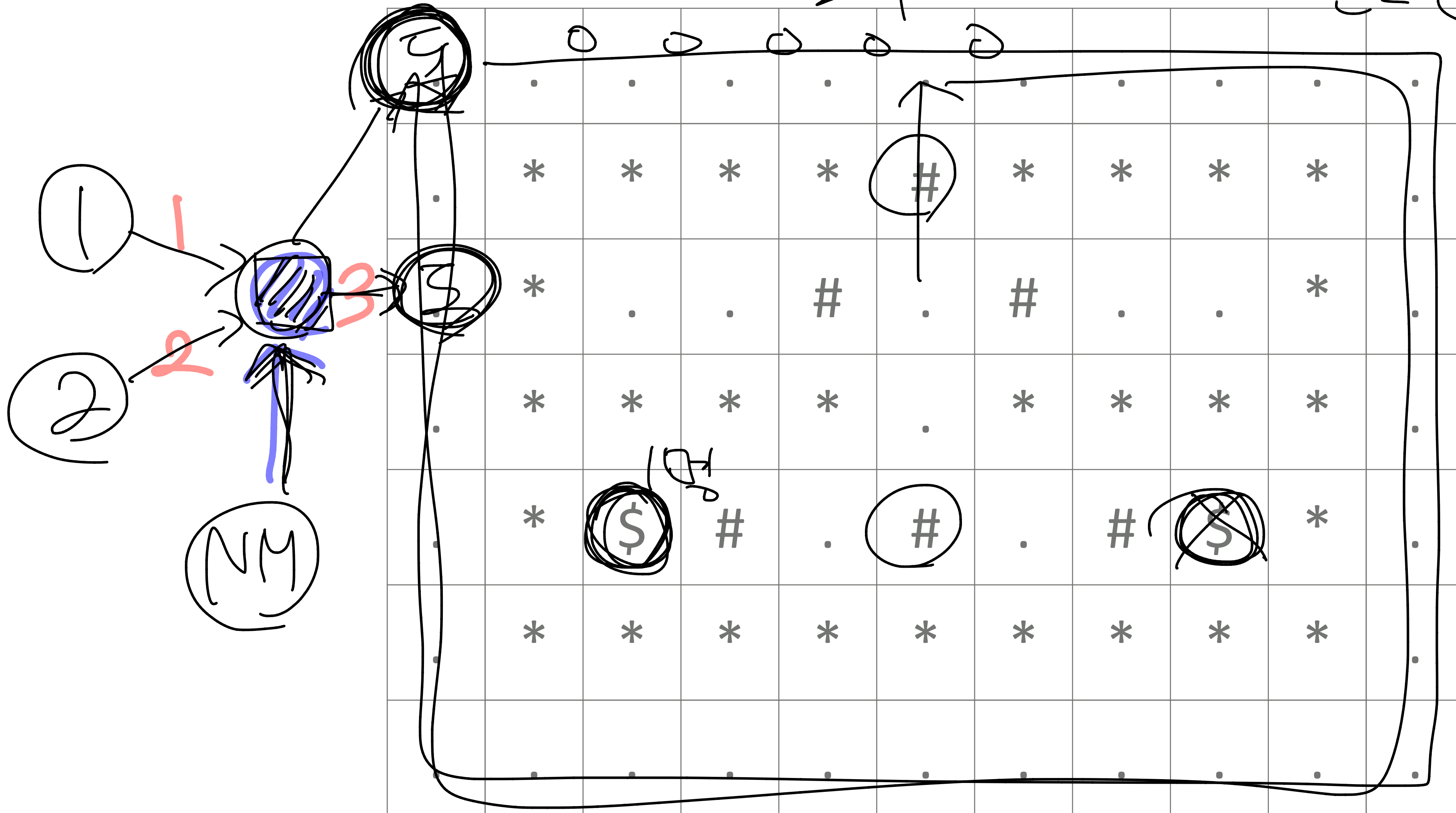
BFS

Pegs

BFS $\frac{2}{2}$ NM

NM \times NM

3



밖에서 부터

죄수 1부터

죄수 2부터

[illegible]

<https://www.acmicpc.net/problem/9376>

밖에서 부터

0	0	0	0	0	0	0	0	0	0	0
0	-	-	-	-	1	-	-	-	-	0
0	-	2	2	2	1	2	2	2	-	0
0	-	-	-	-	1	-	-	-	-	0
0	-	3	3	2	2	2	3	3	-	0
0	-	-	-	-	-	-	-	-	-	0
0	0	0	0	0	0	0	0	0	0	0

죄수 1부터

3	3	3	3	3	3	3	3	3	3	3
3	-	-	-	-	3	-	-	-	-	3
3	-	3	3	3	2	3	3	3	-	3
3	-	-	-	-	2	-	-	-	-	3
3	-	0	1	1	2	2	3	3	-	3
3	-	-	-	-	-	-	-	-	-	3
3	3	3	3	3	3	3	3	3	3	3

죄수 2부터

3	3	3	3	3	3	3	3	3	3	3
3	-	-	-	-	3	-	-	-	-	3
3	-	3	3	3	2	3	3	3	-	3
3	-	-	-	-	2	-	-	-	-	3
3	-	3	3	2	2	1	1	0	-	3
3	-	-	-	-	-	-	-	-	-	3
3	3	3	3	3	3	3	3	3	3	3

-2

탈옥

60

<https://www.acmicpc.net/problem/9376>

- 소스: <http://codeplus.codes/a973211b80e64f90a54eea322463a0b0>

열쇠

<https://www.acmicpc.net/problem/9328>

- 빌딩에서 문서를 훔치는 문제
- 지도에는 문과 열쇠가 있다
- 열쇠를 얻으면 문을 열 수 있다

열쇠

<https://www.acmicpc.net/problem/9328>

- BFS를 큐 27개로 수행해야 한다.
- 큐 1개: 일반적인 BFS
- 큐 26개: 문을 열기 위해 기다리는 큐

열쇠

<https://www.acmicpc.net/problem/9328>

- 큐: (1, 0)

1111111

01234567890123456

0 *****

1 **\$*

2 *B*A*~~2~~*~~3~~*X*Y*.X.

3 *y*x*a*~~p~~*\$*\$*\$*

4 *****

키: cz

열쇠

64

<https://www.acmicpc.net/problem/9328>

- 큐: (1, 1)

```

                                     1111111
01234567890123456
0  *****
1  .....**$*
2  *B*A*P*C**X*Y*.X.
3  *y*x*a*p**$*$**$*
4  *****
키: cz
```


열쇠

65

<https://www.acmicpc.net/problem/9328>

- 큐: (1, 2)
- 큐(B): (2, 1)

```

                                1111111
01234567890123456
0  *****
1  .....**$*
2  *B*A*P*C**X*Y*.X.
3  *y*x*a*p**$*$**$*
4  *****
키: cz
```

열쇠

66

<https://www.acmicpc.net/problem/9328>

- 큐: (1, 3)
- 큐(B): (2, 1)

```

                                11111111
01234567890123456
0  *****
1  .....**$*
2  *B*A*P*C**X*Y*.X.
3  *y*x*a*p**$*$**$*
4  *****
키: cz
```

열쇠

67

<https://www.acmicpc.net/problem/9328>

- 큐: (1, 4)
- 큐(A): (2, 3)
- 큐(B): (2, 1)

```

                                11111111
01234567890123456
0  *****
1  .....**$*
2  *B*A*P*C**X*Y*.X.
3  *y*x*a*p**$*$**$*
4  *****
키: cz
```

열쇠

68

<https://www.acmicpc.net/problem/9328>

- 큐: (1, 5)
- 큐(A): (2, 3)
- 큐(B): (2, 1)

```

                                1111111
01234567890123456
0  *****
1  .....**$*
2  *B*A*P*C**X*Y*.X.
3  *y*x*a*p**$*$**$*
4  *****
키: cz
```

열쇠

69

<https://www.acmicpc.net/problem/9328>

- 큐: (1, 6)
- 큐(A): (2, 3)
- 큐(B): (2, 1)
- 큐(P): (2, 5)

```

                                1111111
01234567890123456
0  *****
1  .....**$*
2  *B*A*P*C**X*Y*.X.
3  *y*x*a*p**$*$**$*
4  *****
키: cz
```

열쇠

70

<https://www.acmicpc.net/problem/9328>

- 큐: (1, 7)
- 큐(A): (2, 3)
- 큐(B): (2, 1)
- 큐(P): (2, 5)

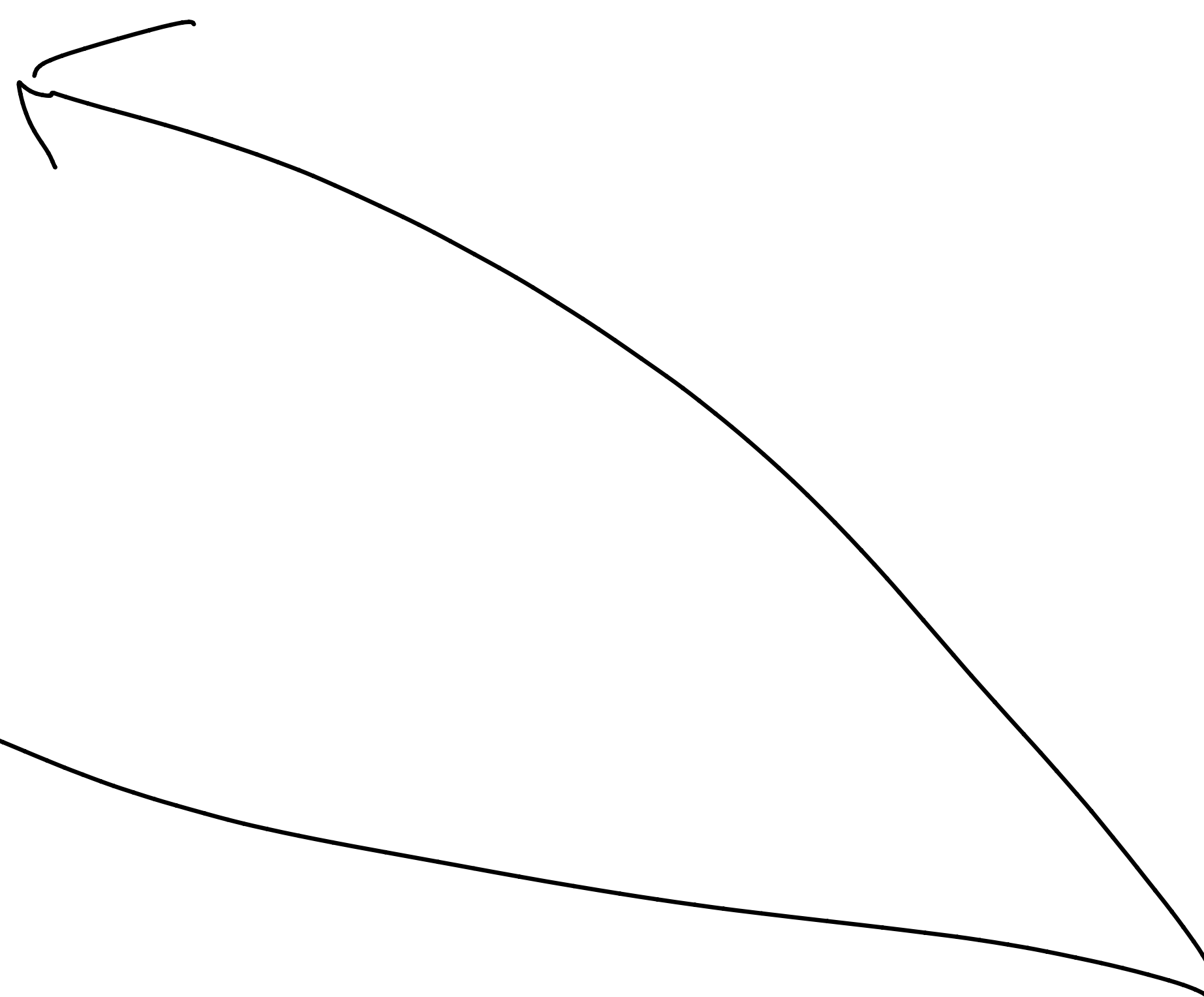
```

                                1111111
01234567890123456
0  *****
1  .....**$*
2  *B*A*P*C**X*Y*.X.
3  *y*x*a*p**$*$*$*
4  *****
키: cz
```

열쇠

71

<https://www.acmicpc.net/problem/9328>

- 큐: (2, 7), (1, 8)
 - 큐(A): (2, 3)
 - 큐(B): (2, 1)
 - 큐(P): (2, 5)
- 

```

                                11111111
01234567890123456
0  *****
1  .....**$*
2  *B*A*P*C**X*Y*.X.
3  *y*x*a*(p)**$*$*$*
4  *****
키: cz
```

열쇠

72

<https://www.acmicpc.net/problem/9328>

- 큐: (1, 8), (3, 7), (2, 5)
- 큐(A): (2, 3)
- 큐(B): (2, 1)

```

                                1111111
01234567890123456
0  *****
1  .....**$*
2  *B*A*P*C**X*Y*.X.
3  *y*x*a*p**$*$**$*
4  *****
키: czp
```


열쇠

73

<https://www.acmicpc.net/problem/9328>

- 큐: (3, 7), (2, 5), (1, 9)
- 큐(A): (2, 3)
- 큐(B): (2, 1)

```

                                1111111
01234567890123456
0  *****
1  .....**$*
2  *B*A*P*C**X*Y*.X.
3  *y*x*a*p**$*$*$*
4  *****
키: czp
```

열쇠

74

<https://www.acmicpc.net/problem/9328>

- 큐: (2, 5), (1, 9), (3, 5), (2, 3)
- 큐(B): (2, 1)

```

                                1111111
01234567890123456
0  *****
1  .....**$*
2  *B*A*P*C**X*Y*.X.
3  *y*x*a*p**$*$**$*
4  *****
키: czpa
```

열쇠

<https://www.acmicpc.net/problem/9328>

- 소스: <http://codeplus.codes/a3184baaf981444f9e45e89bfccd4bba>

로봇 청소기

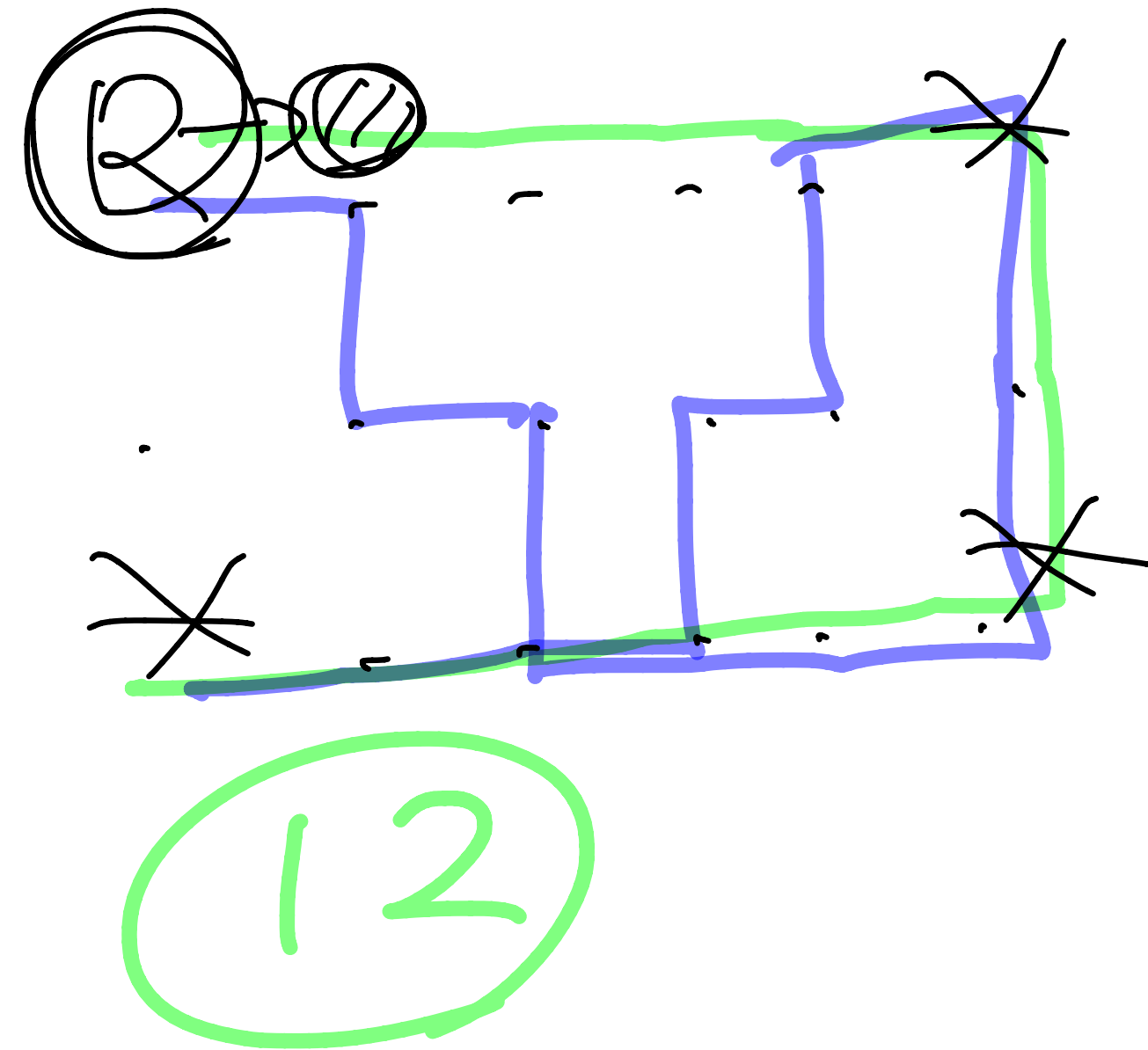
<https://www.acmicpc.net/problem/4991>

(u, v)

BFS
칸

76

- 직사각형 모양의 방이 주어졌을 때
- 모든 더러운 칸을 깨끗한 칸으로 바꾸기 위해 필요한 최소 이동 횟수를 구하는 문제
- 너비, 높이 ≤ 20 , 더러운 칸의 개수 ≤ 10



10!

로봇 청소기

<https://www.acmicpc.net/problem/4991>

- 최소로 이동해야 한다.
- 직사각형 모양의 지도에서 임의의 두 칸 사이의 최소 이동 거리는 BFS로 구할 수 있다

로봇 청소기

<https://www.acmicpc.net/problem/4991>

- 더러운 칸의 개수가 10개 이하이기 때문에
- 모든 순열을 구하고 거리를 계산해서 해결한다.

로봇 청소기

<https://www.acmicpc.net/problem/4991>

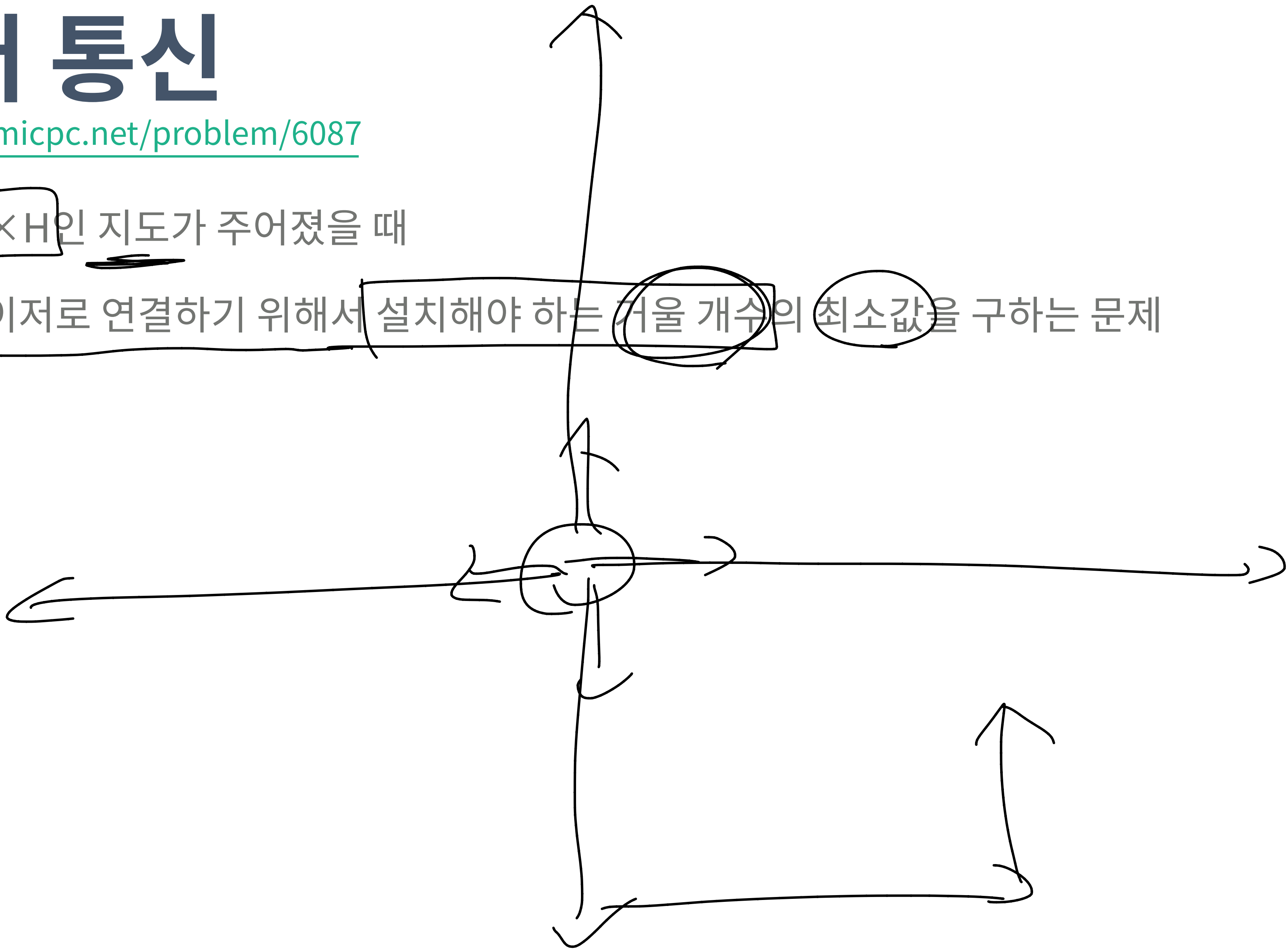
- 소스: <http://codeplus.codes/062c02dda6824a5ea83edaa398dcc6ac>

레이저 통신

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

80

- 크기가 $W \times H$ 인 지도가 주어졌을 때
- 두 C 를 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.

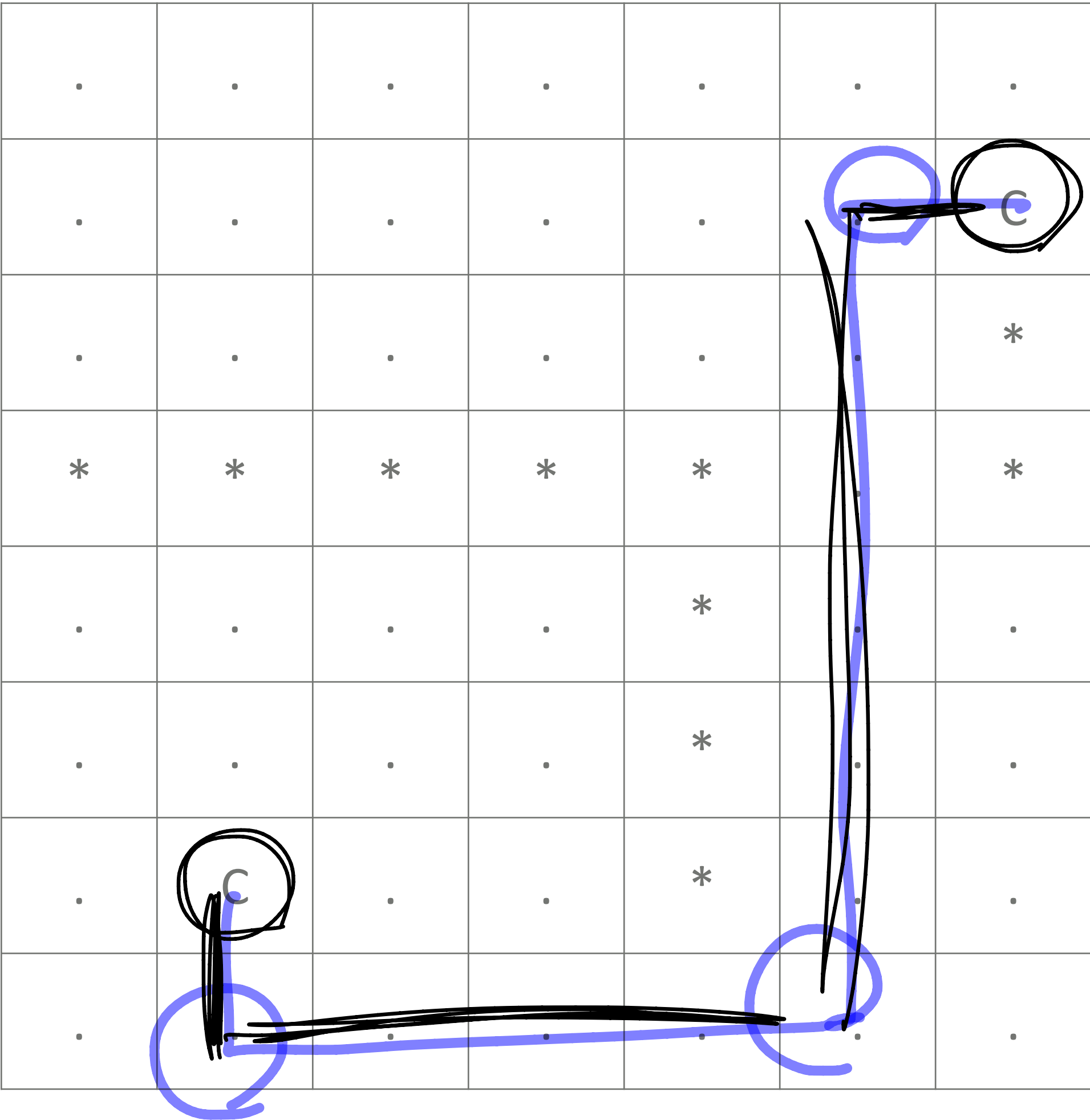
레이저 통신

<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>

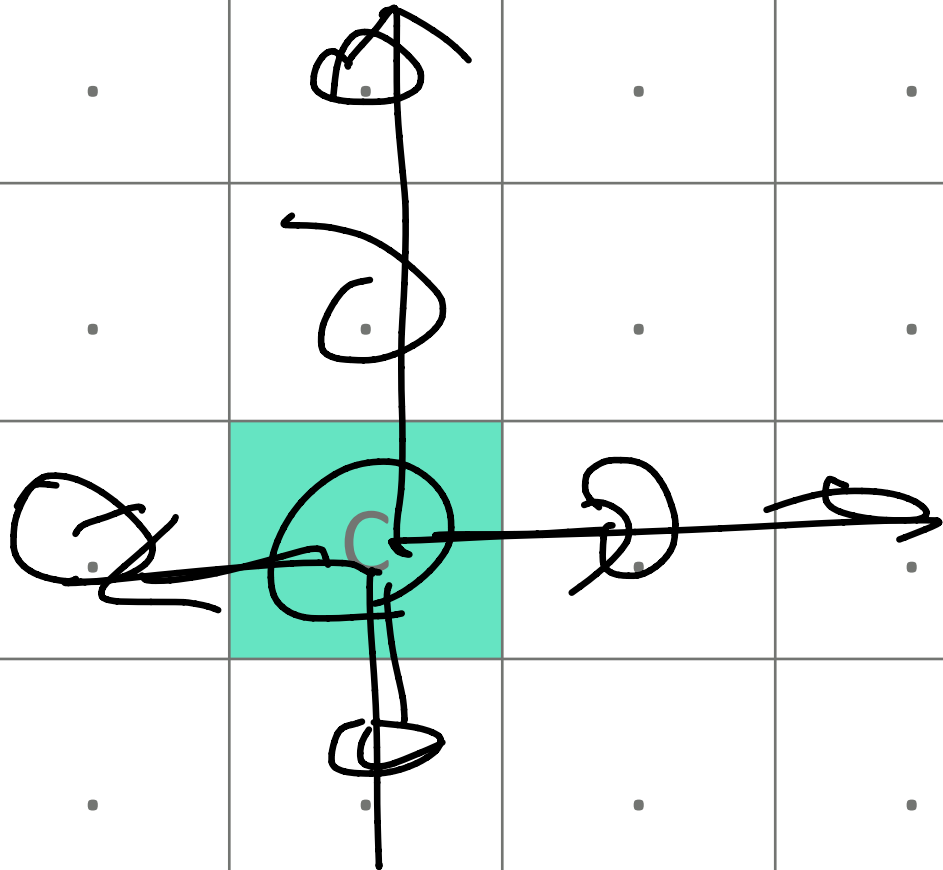


-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1
-1	0	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1

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/56d9cfd1d53847ffadd5e03f42e29ac6>

$$(A \times B) \bmod C = (A \bmod C \times B \bmod C) \bmod C$$

88

3 4 2 (N) ⊕

$$3 \times 10 + 4 \times 10 + 2$$
$$((3 \times 10) + 4) \times 10 + 2$$

% N

0과 1

<https://www.acmicpc.net/problem/8111>

- 자연수 N 이 주어졌을 때 N 의 배수 중에서 다음 조건을 만족하는 수를 찾는 문제 ($N \leq 20,000$)

1. 0과 1로만 이루어져 있다

2. 1이 적어도 하나 있다

3. 수의 길이가 100 이하이다

4. 수가 0으로 시작하지 않는다



0과 1

<https://www.acmicpc.net/problem/8111>

- 0과 1로만 이루어져 있으면서
- 길이가 1인 수: 1
- 길이가 2인 수: 10, 11
- 길이가 3인 수: 100, 101, 110, 111
- 길이가 4인 수: 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111
- 길이가 k 인 수는 총 2^k 개가 존재한다.

0과 1

<https://www.acmicpc.net/problem/8111>

- N의 배수를 구하는 것이기 때문에
- 실제로 그 수가 무엇인지 아는 것 보다는 그 수를 N으로 나눈 나머지가 몇 인지 아는 것이 중요

0과 1

<https://www.acmicpc.net/problem/8111>

- 0과 1로만 이루어져 있으면서
- 길이가 1인 수: 1 ($= 1\%17$)
- 길이가 2인 수: 10 ($= (1*10+0)\%17 = 10$), 11 ($= (1*10+1)\%17 = 11$)
- 길이가 3인 수: 100 ($= (10*10+0)\%17 = 15$), 101 ($= (10*10+1)\%17 = 16$), 110 ($= (11*10+0)\%17 = 8$), 111 ($= (11*10+1)\%17 = 9$)
- 길이가 4인 수: 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111

- 0과 1로 이루어져 있는 수 중에서 N 으로 나눈 나머지는 총 N 개 존재한다.

0 ~ $N-1$

0과 1

<https://www.acmicpc.net/problem/8111>

- 소스: <http://codeplus.codes/fa32aa582a8c44b99b9674a35fb027ad>

점프 게임

<https://www.acmicpc.net/problem/15558>

- 오른쪽 그림과 같은 지도가 있다 ($N \leq 100,000$)
- 유저가 할 수 있는 행동은 아래 3가지 중 하나이다
- 한 칸 위로, 한 칸 아래로, 옆 칸으로 (+k만큼 이동)
- i초에 i번 칸이 사라진다.
- N번 칸을 넘어갈 수 있는지 구하는 문제



N
N-1
⋮
⋮
⋮
3
2
1

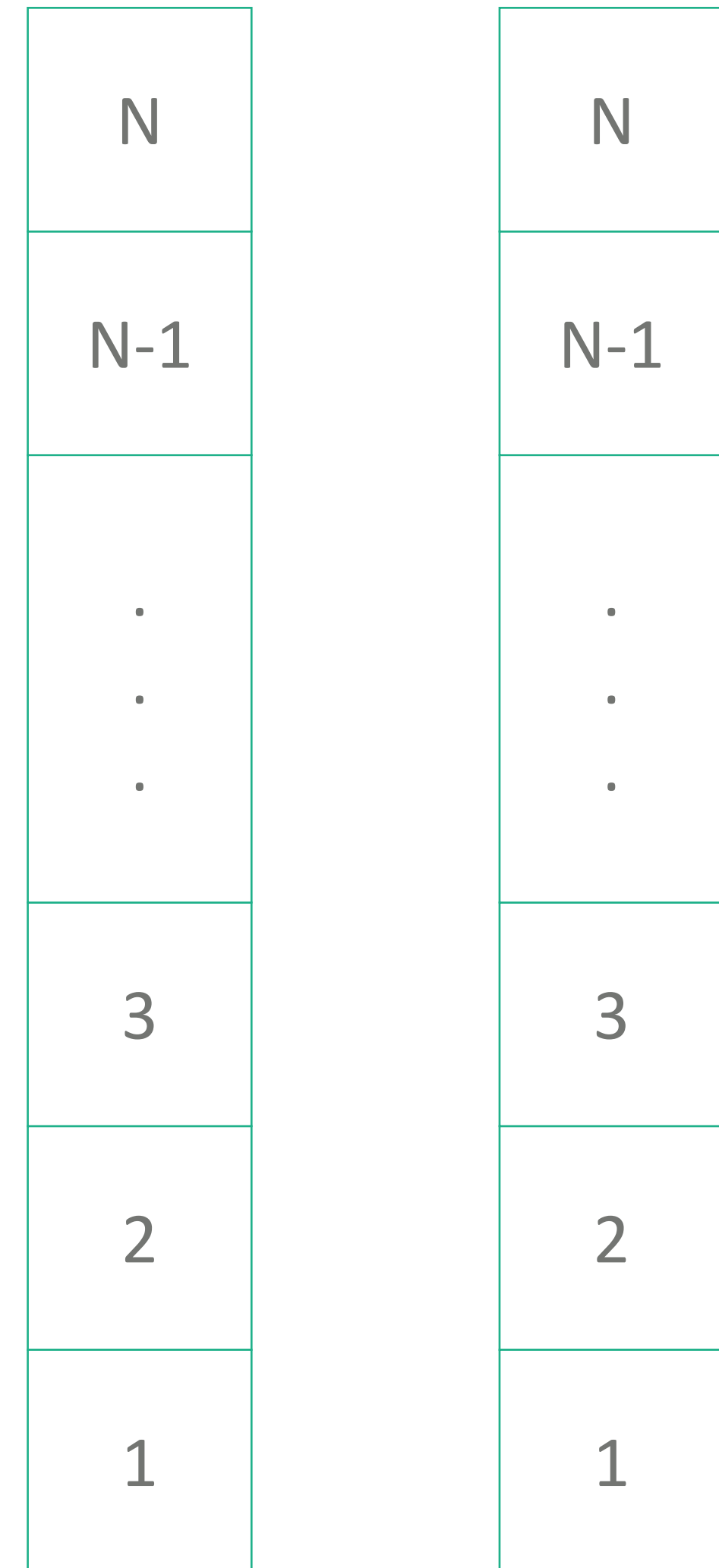
N
N-1
⋮
⋮
⋮
3
2
1

점프 게임

95

<https://www.acmicpc.net/problem/15558>

- 만약, 칸이 사라지는 조건이 없으면, BFS로 해결할 수 있다.



점프 게임

<https://www.acmicpc.net/problem/15558>

- BFS는 어떤 칸을 방문하는 최단 거리를 구하게 되는데
- i 번 칸을 방문한 초 $\geq i$ 이면 방문할 수 있는 것이다.

N
N-1
.
.
.
3
2
1

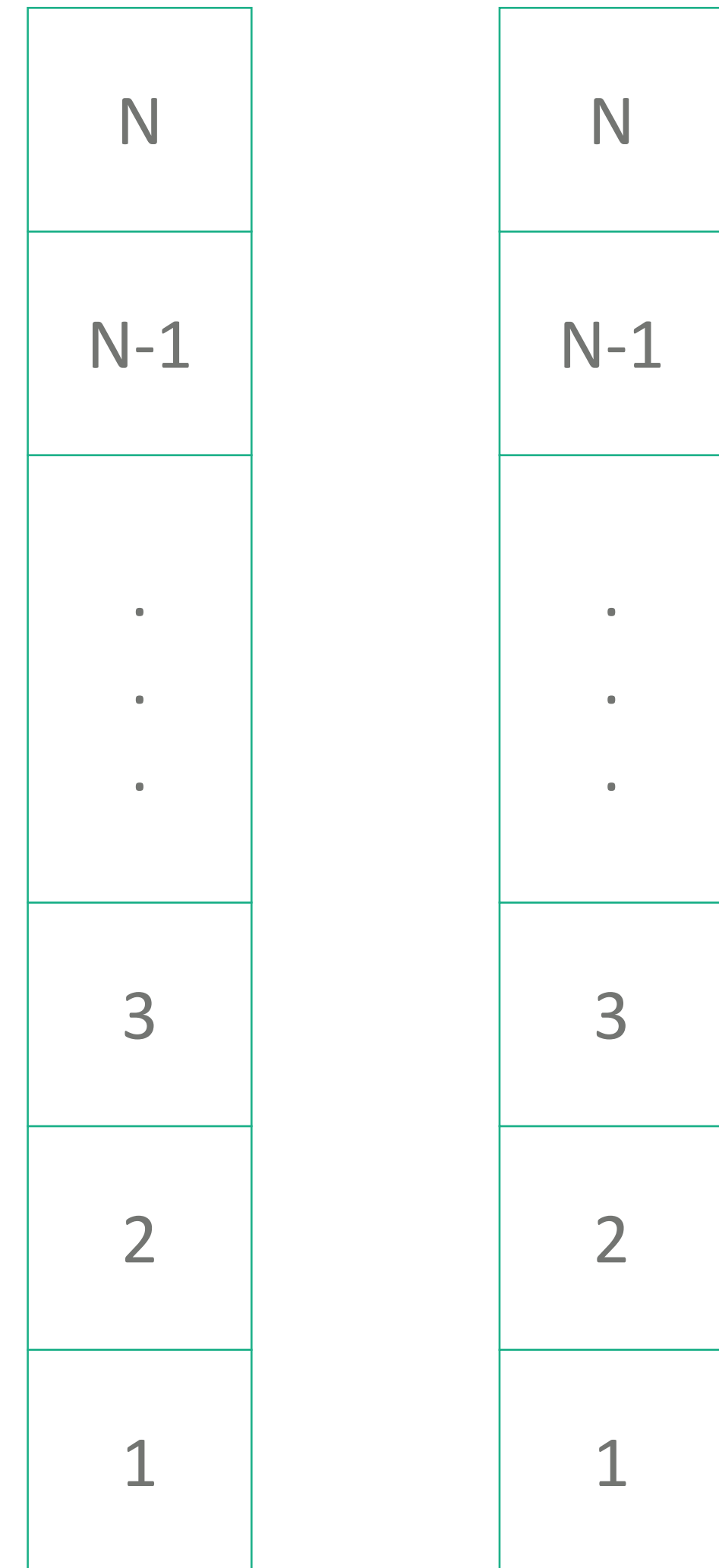
N
N-1
.
.
.
3
2
1

점프 게임

97

<https://www.acmicpc.net/problem/15558>

- 소스: <http://codeplus.codes/6f2617e81b5a49d9bcacbfced20bb5b0>



끝

코드 플러스

<https://code.plus>

- 슬라이드에 포함된 소스 코드를 보려면 "정보 수정 > 백준 온라인 저지 연동"을 통해 연동한 다음, "백준 온라인 저지"에 로그인해야 합니다.
- 강의 내용에 대한 질문은 코드 플러스의 "질문 게시판"에서 할 수 있습니다.
- 문제와 소스 코드는 슬라이드에 첨부된 링크를 통해서 볼 수 있으며, "백준 온라인 저지"에서 서비스됩니다.
- 슬라이드와 동영상 강의는 코드 플러스 사이트를 통해서만 볼 수 있으며, 동영상 강의의 녹화와 다운로드, 배포와 유통은 저작권법에 의해서 금지되어 있습니다.
- 다른 경로로 이 슬라이드나 동영상 강의를 본 경우에는 codeplus@startlink.io 로 이메일 보내주세요.
- 강의 내용, 동영상 강의, 슬라이드, 첨부되어 있는 소스 코드의 저작권은 스타트링크와 최백준에게 있습니다.