

BFS 2

최백준 choi@startlink.io

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 ($=(1 \times 10 + 0)\%17 = 10$), 11 ($=(1 \times 10 + 1)\%17 = 11$)
- 길이가 3인 수: 100 ($=(10 \times 10 + 0)\%17 = 15$), 101 ($=(10 \times 10 + 1)\%17 = 16$), 110 ($=(11 \times 10 + 0)\%17 = 8$), 111 ($=(11 \times 10 + 1)\%17 = 9$)
- 길이가 4인 수: 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111
- 0과 1로 이루어져 있는 수 중에서 N으로 나눈 나머지는 총 N개 존재한다.

0과 1

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

- 소스: <http://codeplus.codes/ab5b9be07a2942d6aaa0d404a305281c>

숨바꼭질 5

<https://www.acmicpc.net/problem/17071>

- 수빈이는 N에 있고, 동생은 K에 있다. ($0 \leq N, K \leq 500,000$)
- 수빈이가 동생을 찾을 수 있는 가장 빠른 시간을 구하는 문제
- 수빈이의 가능한 이동 1초 후에 $X \rightarrow 2X, X+1, X-1$ 중 하나
- 동생의 이동 $K \rightarrow K+1 \rightarrow K+1+2 \rightarrow K+1+2+3 \rightarrow \dots$
- 0보다 작은 좌표, 50보다 큰 좌표로 이동은 불가능, 정수 좌표에서만 찾을 수 있다.
- $N = 5, K = 17$ 인 경우 2초
- $N = 17, K = 5$ 인 경우 4초
- $N = 1, K = 10$ 인 경우 6초

숨바꼭질 5

<https://www.acmicpc.net/problem/17071>

- i 초 후의 동생의 위치를 알 수 있다.
- 동생이 이동할 때마다 BFS를 이용해 가장 빠른 시간을 구해볼 수 있다.

숨바꼭질 5

<https://www.acmicpc.net/problem/17071>

- 소스: <http://codeplus.codes/92cb9856a60d468e9741b2d35d5c130b>
- 가능한 동생의 위치는 $\sqrt{500,000}$ 이다.
- BFS의 시간 복잡도는 $O(500,000)$ 이기 때문에, $O(500,000\sqrt{500,000})$ 이라 시간이 매우 많이 걸린다.

숨바꼭질 5

<https://www.acmicpc.net/problem/17071>

- 수빈이의 가능한 이동 1초 후에 $X \rightarrow 2X, X+1, X-1$ 중 하나
- $X \rightarrow X+1 \rightarrow X$ 의 이동이 가능하다.
- 즉, 수빈이가 한 위치에 도착했다면, 2초마다 같은 위치로 이동할 수 있다.
- 홀수 시간에 어떤 칸에 도착했고, 동생이 홀수 시간만에 그 위치로 왔다면, 찾을 수 있다.
- 짝수 시간에 어떤 칸에 도착했고, 동생이 짝수 시간만에 그 위치로 왔다면, 찾을 수 있다.

숨바꼭질 5

<https://www.acmicpc.net/problem/17071>

- BFS를 이용하는데, 어떤 정점에 홀수 시간에 도착한 경우, 짝수 시간에 도착한 경우로 나누어서 최소 시간을 구해야 한다.

숨바꼭질 5

<https://www.acmicpc.net/problem/17071>

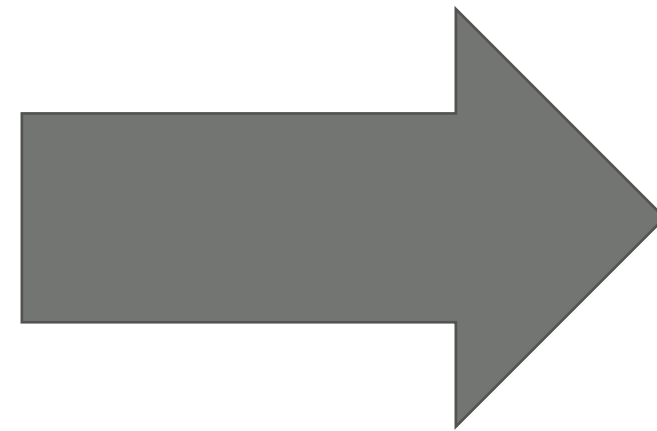
- 소스: <http://codeplus.codes/f761cd66d3b741d7a9871b09110721ba>

퍼즐

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

- 8퍼즐을 푸는 문제

8	2	
7	1	3
6	5	4



1	2	3
4	5	6
7	8	

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

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

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

퍼즐

<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>

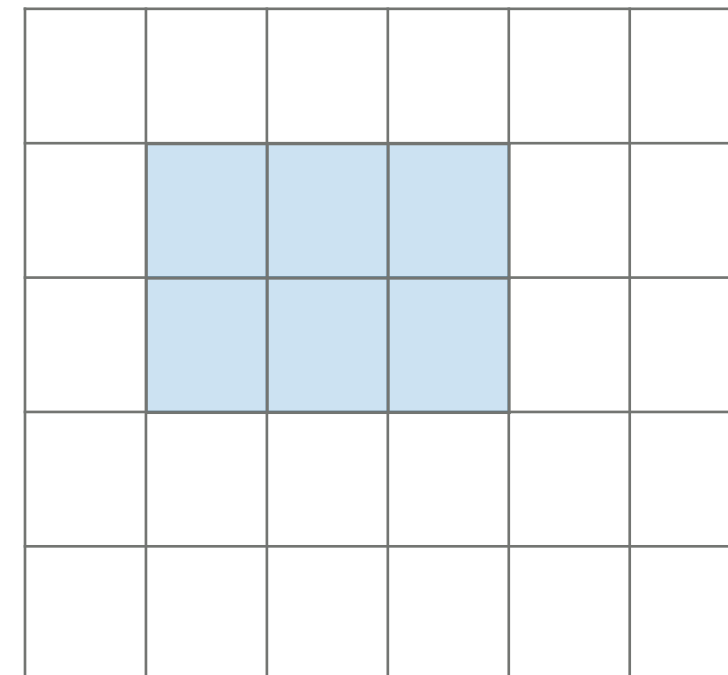
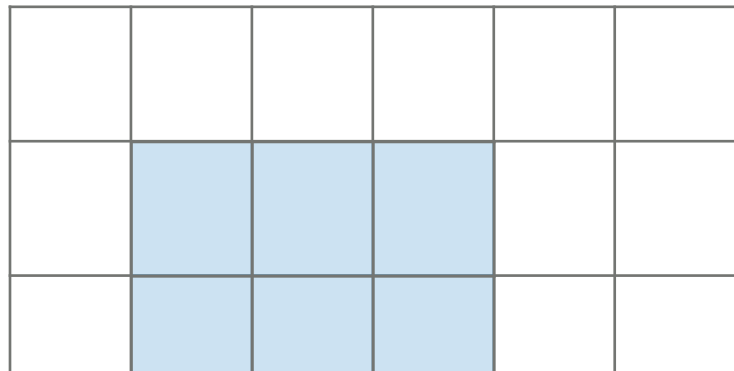
```
for (int k=0; k<4; k++) {
    int nx = x+dx[k];
    int ny = y+dy[k];
    if (nx >= 0 && nx < n && ny >= 0 && ny < n) {
        string next = now;
        swap(next[x*3+y], next[nx*3+ny]);
        int num = stoi(next);
        if (d.count(num) == 0) {
            d[num] = d[now_num] + 1;
       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

퍼즐

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

- 소스: <http://codeplus.codes/3e02ac9404d64eca8428f0c0581d9964>

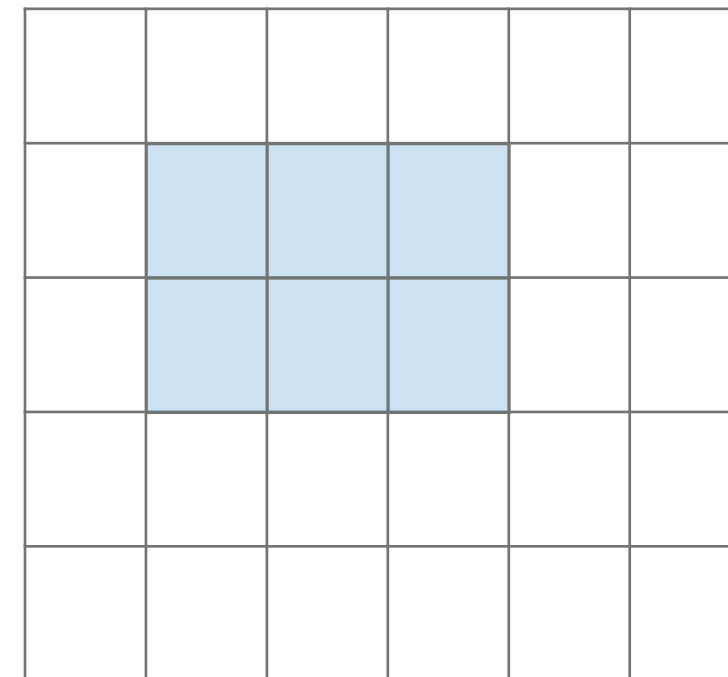
- 크기가 $N \times M$ 인 격자판에 크기가 $H \times W$ 인 직사각형이 놓여져 있다.
- 직사각형을 (S_r, S_c) 에서 (F_r, F_c) 로 옮기는 문제 (가장 왼쪽 칸 기준)
- 이동은 상하좌우 4방향이 가능
- 일부 칸은 이동할 수 없는 칸
- $2 \leq N, M \leq 1,000$



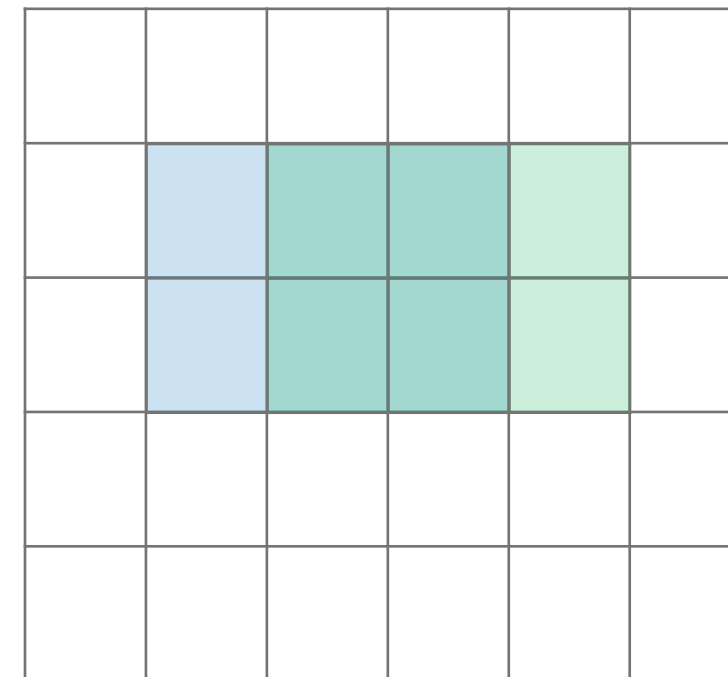
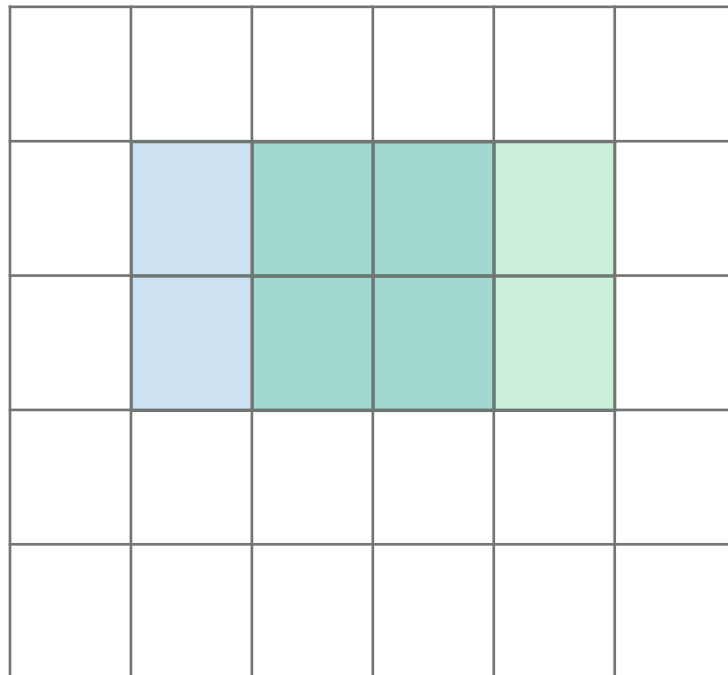
직사각형 탈출

<https://www.acmicpc.net/problem/16973>

- 직사각형의 정보를 저장하기 위해서 직사각형의 모든 칸을 저장할 필요는 없다.
- 가장 왼쪽 위칸과 직사각형의 크기를 알고 있으면, 직사각형을 만들 수 있다.
- 따라서, 가장 왼쪽 위칸의 정보만 이용하면 된다.



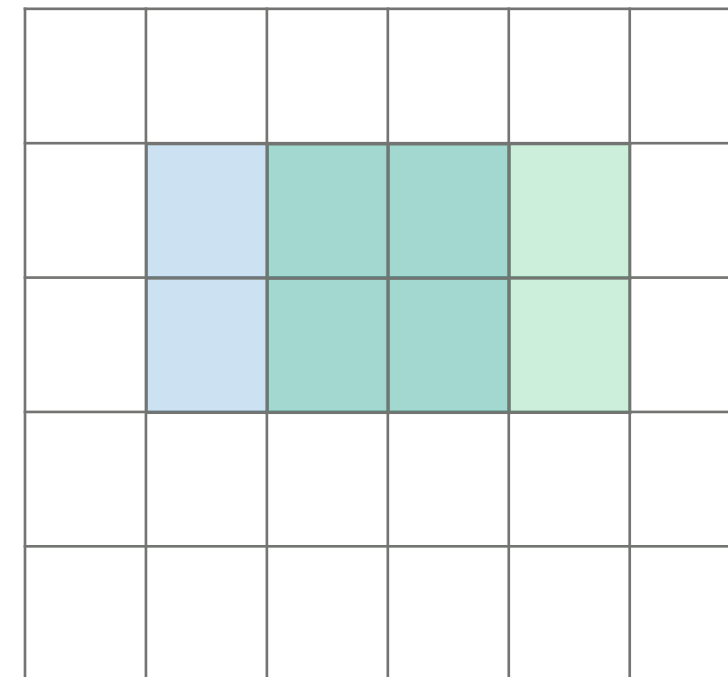
- 1×1 인 직사각형이라면, $O(NM)$ 이 걸린다.
- 크기가 $H \times W$ 이기 때문에, 4방향 중 하나를 검사하는데 걸리는 시간이 $O(HW)$ 이다.
- 따라서, $O(NMHW)$ 에 해결할 수 있다.
- $2 \leq N, M \leq 1,000$
- $1 \leq H \leq N$
- $1 \leq W \leq M$
- 너무 오랜 시간이 걸린다.



직사각형 탈출

<https://www.acmicpc.net/problem/16973>

- 1×1 인 직사각형이라면, $O(NM)$ 이 걸린다.
- 이동하기 전과 다른 칸의 개수는 W 개 또는 H 개 이기 때문에, 4방향 중 하나를 검사하는데 걸리는 시간이 $O(W)$ 또는 $O(H)$ 이다.
- 따라서, $O(NMH + NMW)$ 에 해결할 수 있다.
- $2 \leq N, M \leq 1,000$
- $1 \leq H \leq N$
- $1 \leq W \leq M$
- 꽤 오랜 시간이 걸린다.



직사각형 탈출

<https://www.acmicpc.net/problem/16973>

- 가장 왼쪽 칸을 기준으로 $O(1)$ 만에 어떤 방향으로 이동할 수 있는지 없는지 알아낼 수 있다.

직사각형 탈출

<https://www.acmicpc.net/problem/16973>

- 이동하려고 하는 직사각형의 합이 0인 경우에만 이동할 수 있다.
- 부분 직사각형의 합은 누적합을 이용해 $O(1)$ 만에 구할 수 있다.

직사각형 탈출

<https://www.acmicpc.net/problem/16973>

- 소스: <http://codeplus.codes/2085f9812ef64e2b86dab04502de3f0e>

배달

<https://www.acmicpc.net/problem/1175>

- N행 M열의 격자판에서
- S에서 C로 이동하는 최소 이동 횟수를 구하는 문제
- 이 때, C의 개수는 2개이고, 계속해서 방향을 바꿔야 한다.

배달

<https://www.acmicpc.net/problem/1175>

- 총 상태의 개수
- 칸의 좌표, 현재 칸에 들어온 방향, 배달 완료한 선물의 개수
- $50 \times 50 \times 4 \times 4$
- 상태 $(r, c, k, s) = (r, c)$ 에 k 방향으로 들어옴. 완료한 선물의 상태: s

배달

<https://www.acmicpc.net/problem/1175>

- 소스: <http://codeplus.codes/16e9dc719503409ea94c7c8323d2547e>

체스판 여행 1

<https://www.acmicpc.net/problem/16959>

- 크기가 $N \times N$ 인 체스판이 있고, 각 칸에는 1부터 N^2 까지의 정수가 한 번씩 적혀있다.
- $1 \rightarrow 2 \rightarrow \dots \rightarrow N^2$ 까지 이동하려고 한다. $3 \leq N \leq 10$
- 중간에 다른 칸을 방문할 수도 있고, 같은 칸을 여러 번 방문하는 것도 가능하다.
- 1에 나이트, 비숍, 룯 중 하나를 놓고 시작한다.
- 1초 동안 할 수 있는 행동: 말을 이동시키거나, 다른 말로 바꾸는 것
- 필요한 시간의 최솟값을 구하는 문제

체스판 여행 1

<https://www.acmicpc.net/problem/16959>

- 나이트로 시작

1	9	3
8	6	7
4	2	5

- $(3, 2) \rightarrow (1, 3) \rightarrow (3, 2) \rightarrow$ 룯으로 바꿈 $\rightarrow (3, 1) \rightarrow (3, 3) \rightarrow (3, 2) \rightarrow (2, 2) \rightarrow (2, 3) \rightarrow (2, 1) \rightarrow (1, 1) \rightarrow (1, 2)$

체스판 여행 1

<https://www.acmicpc.net/problem/16959>

- 크기가 $N \times N$ 인 체스판이 있고, 각 칸에는 1부터 N^2 까지의 정수가 한 번씩 적혀있다.
- $1 \rightarrow 2 \rightarrow \dots \rightarrow N^2$ 까지 이동하려고 한다. $3 \leq N \leq 10$
- 중간에 다른 칸을 방문할 수도 있고, **같은 칸을 여러 번 방문하는 것도 가능하다.**
- $(1 \rightarrow 2), (2 \rightarrow 3), \dots, (N^2-1 \rightarrow N^2)$ 으로 쪼개서 문제를 풀 수 있다.
- 같은 칸을 $1 \rightarrow 2$ 에서 방문하고, $2 \rightarrow 3$ 에서도 방문했을 수 있다.
- 하지만, $X \rightarrow X+1$ 로 이동할 때, 같은 칸을 여러 번 방문하는 것은 절대로 최소가 아니다.

체스판 여행 1

<https://www.acmicpc.net/problem/16959>

- BFS를 이용할 수 있다.
- (r, c, num, piece)
 - (r, c): 말이 놓여져 있는 위치
 - num: 현재 몇 번까지 방문을 마쳤는지 ($\text{num} \rightarrow \text{num}+1$)로 가는 중이라는 의미
 - piece: 말의 종류 (0: 나이트, 1: 룯, 2: 비숍)

체스판 여행 1

<https://www.acmicpc.net/problem/16959>

- BFS를 이용할 수 있다.
- (r, c, num, piece)
 - (r, c): 말이 놓여져 있는 위치
 - num: 현재 몇 번까지 방문을 마쳤는지 ($\text{num} \rightarrow \text{num}+1$)로 가는 중이라는 의미
 - piece: 말의 종류 (0: 나이트, 1: 룯, 2: 비숍)
- 크게 2가지의 이동이 가능하다.
 - 다른 말로 교체한다.
 - 이동한다.

체스판 여행 1

35

<https://www.acmicpc.net/problem/16959>

- 소스: <http://codeplus.codes/d96d68fce60d46faaec76e332d31845b>

체스판 여행 2

<https://www.acmicpc.net/problem/16952>

- 크기가 $N \times N$ 인 체스판이 있고, 각 칸에는 1부터 N^2 까지의 정수가 한 번씩 적혀있다.
- $1 \rightarrow 2 \rightarrow \dots \rightarrow N^2$ 까지 이동하려고 한다. $3 \leq N \leq 10$
- 중간에 다른 칸을 방문할 수도 있고, 같은 칸을 여러 번 방문하는 것도 가능하다.
- 1에 나이트, 비숍, 룯을 놓고 시작한다.
- 1초 동안 할 수 있는 행동: 말을 이동시키거나, 다른 말로 바꾸는 것
- 필요한 시간의 최소값을 구하는 문제
- 이때, 말을 교체한 횟수도 최소로 해야 한다.

체스판 여행 2

<https://www.acmicpc.net/problem/16952>

- 체스판 여행 1과 비슷하게 해결할 수 있다.
- 거리를 저장할 배열에 두 개의 값을 저장해야 한다.

체스판 여행 2

<https://www.acmicpc.net/problem/16952>

- 소스: <http://codeplus.codes/ce38c205f4834ed7aa4d39d6c77d0b2d>

숨바꼭질 2

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

- 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)

숨바꼭질 2

40

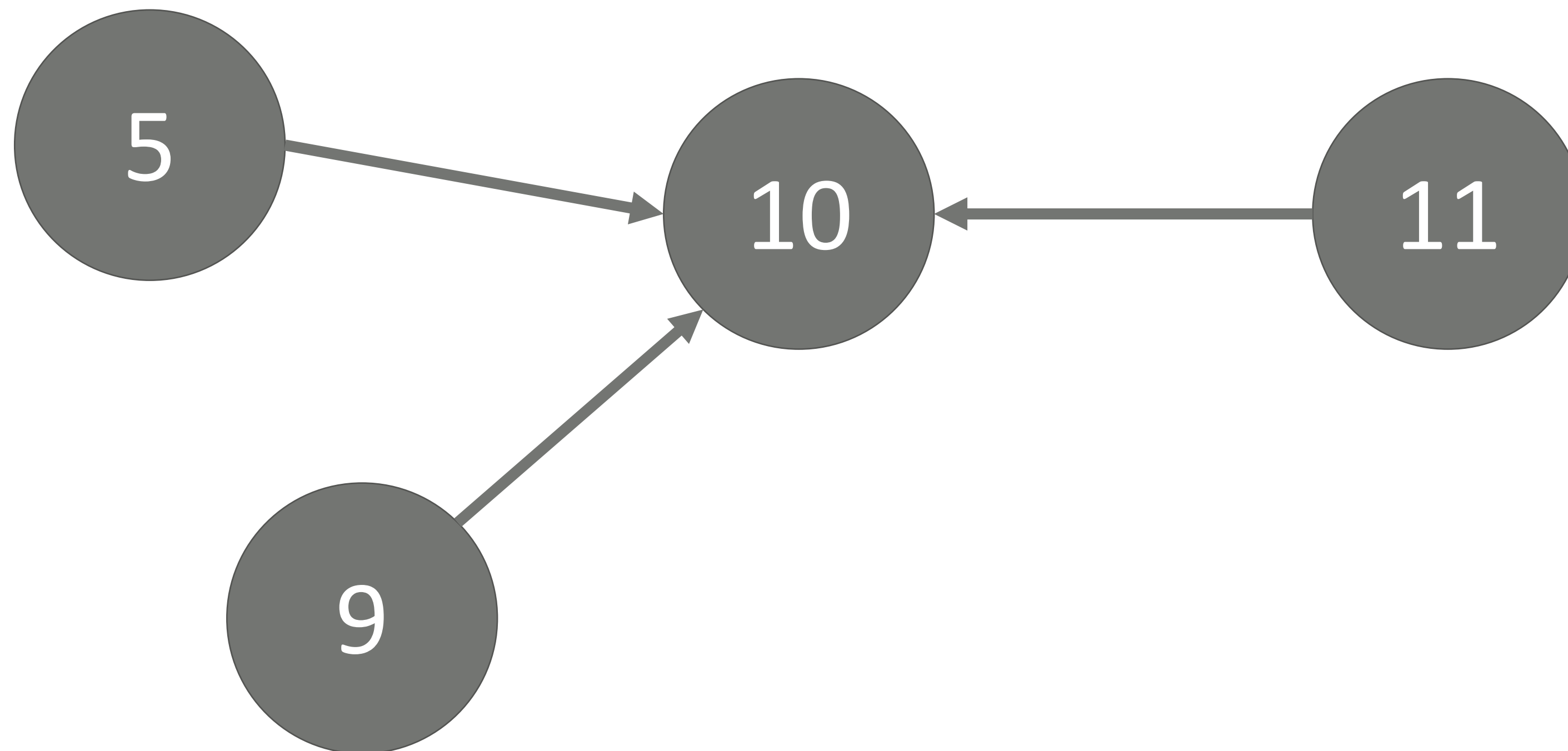
<https://www.acmicpc.net/problem/12851>

- 경우의 수는 다이나믹 프로그래밍으로 구할 수 있다
- $\text{cnt}[i] = i$ 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

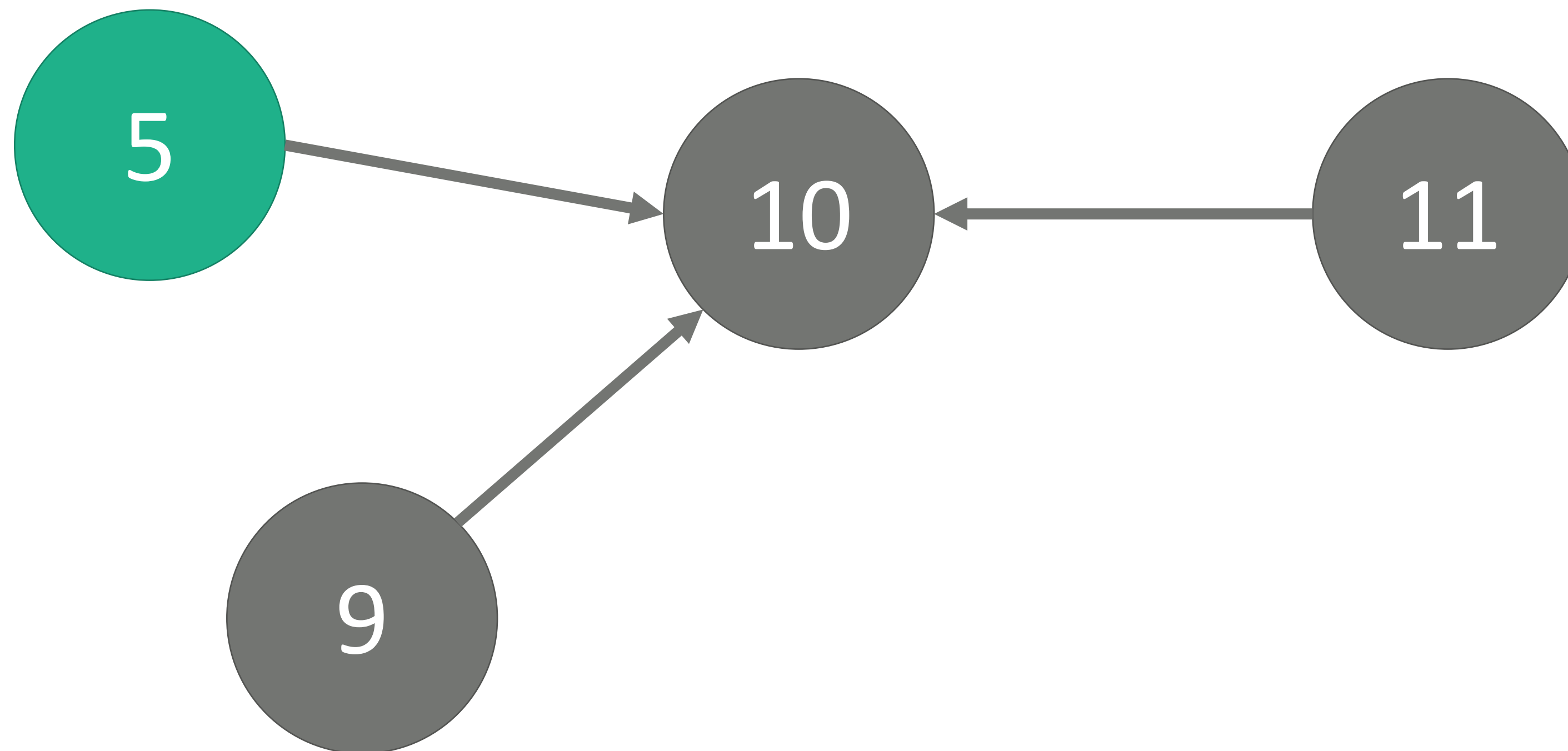


숨바꼭질 2

42

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

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

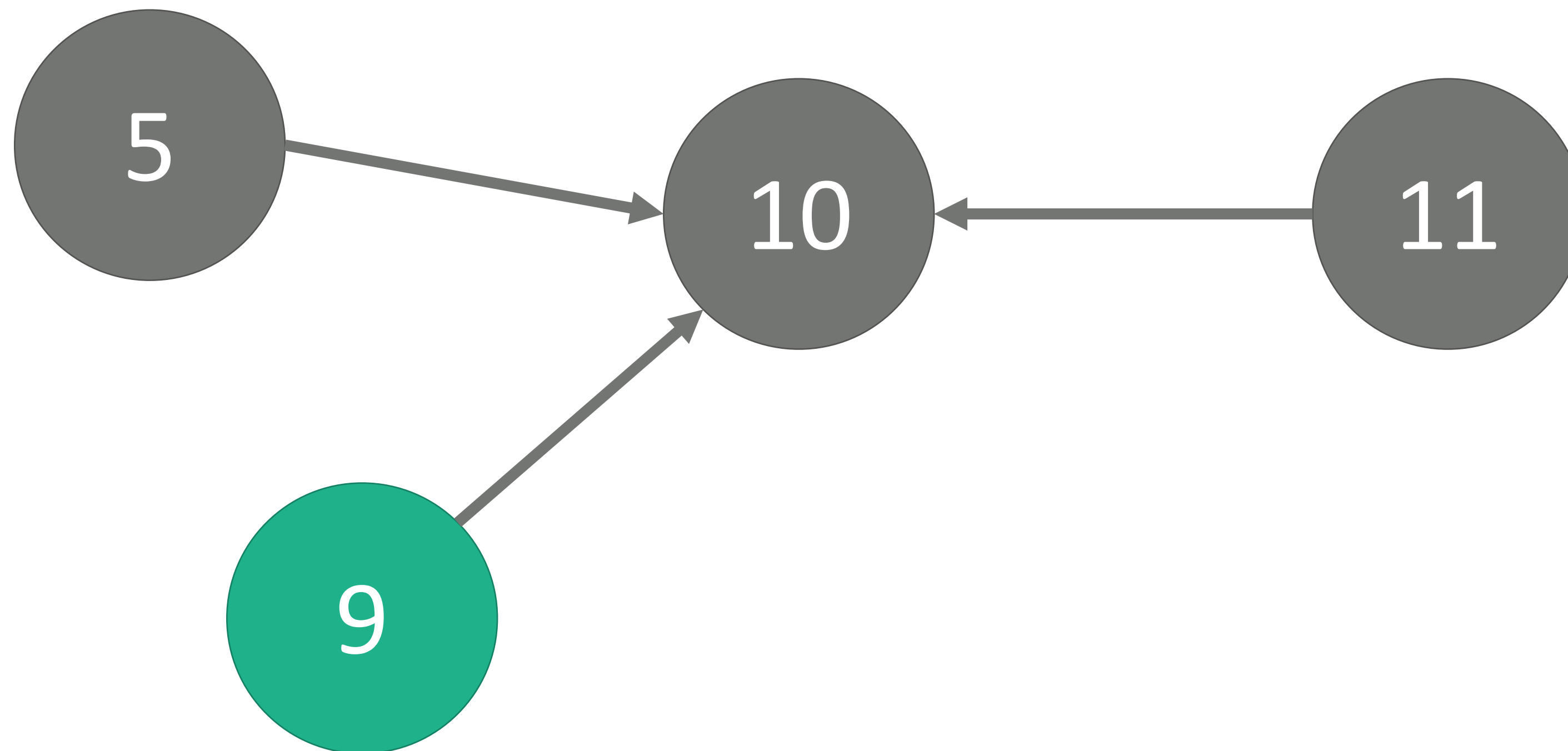


숨바꼭질 2

43

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

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



숨바꼭질 2

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

```
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

45

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

- 소스: <http://codeplus.codes/88736bb1083f4c9b84df726352956d33>

백조의 호수

<https://www.acmicpc.net/problem/3197>

- 두 마리의 백조가 호수에 살고 있다
- 두 마리는 호수를 덮고 있는 얼음 때문에 만날 수 없다
- 매일 물과 접촉한 얼음은 녹는다 (가로, 세로)
- 몇 일이 지나야 백조가 만날 수 있을까?

백조의 호수

<https://www.acmicpc.net/problem/3197>

```
...XXXXXX..XX.XXX  ...XXXX.....XX  ...XX.....
...XXXXXXXXXX.XXX  ...XXXX..X.....  ...X.....
...XXXXXXXXXXXXX..  ...XXX..XXXX....  ...X...X....
..XXXXX.LXXXXXX..  ...XXX...XXXX....  ...X.....XX....
.XXXXXX..XXXXXX..  ..XXXX...XXXX....  ...XX.....XX....
XXXXXXXXX...XXXX..  ..XXXX.....XX....  ...X.....
..XXXXX...XXX....  ...XX...X.....
...XXXXX.XXXL...  ...XX...X.....
```

시작

첫째 날

둘째 날

백조의 호수

<https://www.acmicpc.net/problem/3197>

- 물의 퍼짐과 백조의 이동을 BFS로 진행할 수 있다
- 각각의 날에 대해서
- 물을 먼저 이동시키고, 그 다음에 백조를 이동시키면 된다

백조의 호수

<https://www.acmicpc.net/problem/3197>

- 소스: <http://codeplus.codes/0aab51f4601d47afa523e05d76f24f59>

열쇠

50

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

열쇠

53

<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

<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

55

<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

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

- 큐: (1, 4)
- 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

57

<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

58

<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

59

<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

60

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

- 큐: (2, 7), (1, 8)
- 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

61

<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

62

<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

63

<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

64

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

- 소스: <http://codeplus.codes/0a3c023f3e0547f88f893c2c2fdd51bb>

확장 게임

<https://www.acmicpc.net/problem/16920>

- 크기가 $N \times M$ 인 지도와 플레이어 P 명이 있다. 지도의 각 칸은 빈 칸 또는 벽
- 각 플레이어는 지도 위에 성을 하나 이상 가지고 있는데, 성을 확장할 수 있다.
- 각 턴마다 자신의 성과 거리가 $S[i]$ 이하인 모든 칸으로 성을 확장한다.
- 최종 성의 개수를 구하는 문제

확장 게임

66

<https://www.acmicpc.net/problem/16920>

3 3 2

1 1

1..

...

..2

• 답: 6 3

확장 게임

67

<https://www.acmicpc.net/problem/16920>

4 4 2

1 1

1...

....

....

...2

• 답: 10 6

확장 게임

68

<https://www.acmicpc.net/problem/16920>

4 4 2

1 1

1..1

....

....

...2

• 답: 11 6

확장 게임

69

<https://www.acmicpc.net/problem/16920>

4 4 2

2 1

1..1

....

....

...2

• 답: 14 2

확장 게임

70

<https://www.acmicpc.net/problem/16920>

- BFS를 이용해서 해결할 수 있다.
- 큐에 시작점을 여러개 넣으면 된다.
- 플레이어어 별로 BFS를 이용한다.
- 라운드 별로 BFS를 처음부터 다시 할 필요는 없다.

확장 게임

<https://www.acmicpc.net/problem/16920>

- 소스: <http://codeplus.codes/cf2e2dd25e144d47aaa572fd2e283f70>

구슬 탈출 4

72

<https://www.acmicpc.net/problem/15653>

- 보드의 상태가 주어졌을 때, 최소 몇 번 만에 빨간 구슬을 구멍을 통해 빼낼 수 있는지 구하는 문제
- 만약, 어떻게 움직여도 빨간 구슬을 구멍을 통해 빼낼 수 없으면 -1을 출력

구슬 탈출 4

73

<https://www.acmicpc.net/problem/15653>

- 보드의 상태를 변수로 나타내보자

구슬 탈출 4

<https://www.acmicpc.net/problem/15653>

- 보드에는 빈 칸, 벽, 빨간 구슬, 파란 구슬, 구멍이 있다.
- 가능한 방법은 구슬을 이동시키는 것이다.

구슬 탈출 4

<https://www.acmicpc.net/problem/15653>

- 보드에는 빈 칸, 벽, 빨간 구슬, 파란 구슬, 구멍이 있다.
- 가능한 방법은 구슬을 이동시키는 것이다.
- 빈 칸, 벽, 구멍은 어떻게 이동시켜도 변하지 않는다.
- 빨간 구슬과 파란 구슬만 변한다.
- 따라서, 구슬의 위치가 문제의 상태가 된다.

구슬 탈출 4

76

<https://www.acmicpc.net/problem/15653>

- 상태: (rx, ry, bx, by)
- 빨간 구슬의 위치가 (rx, ry) 이고, 파란 구슬의 위치가 (bx, by)
- 총 가능한 상태의 개수: $(NM)^2$

구슬 탈출 4

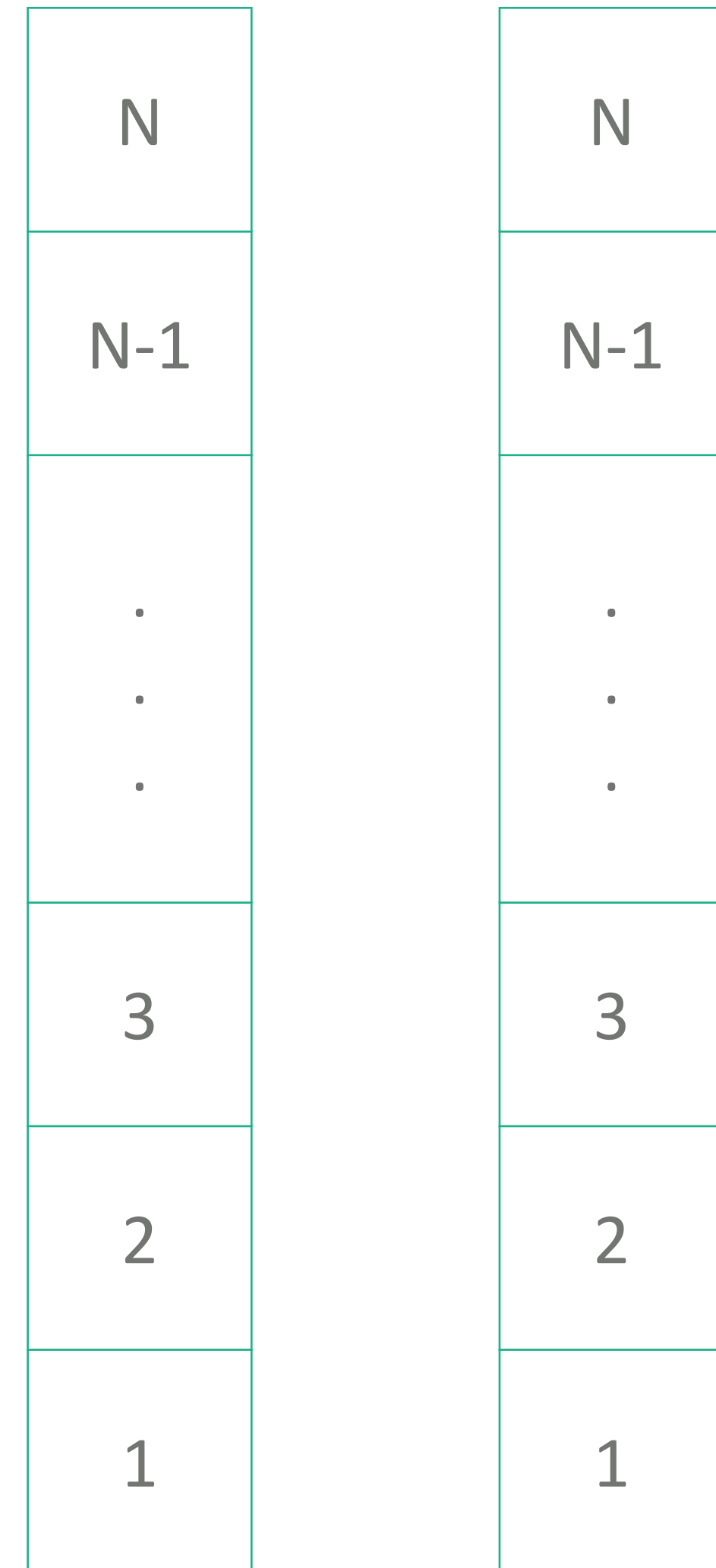
<https://www.acmicpc.net/problem/15653>

- 소스: <http://codeplus.codes/2abc7bcd093404c8278506493060232>

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

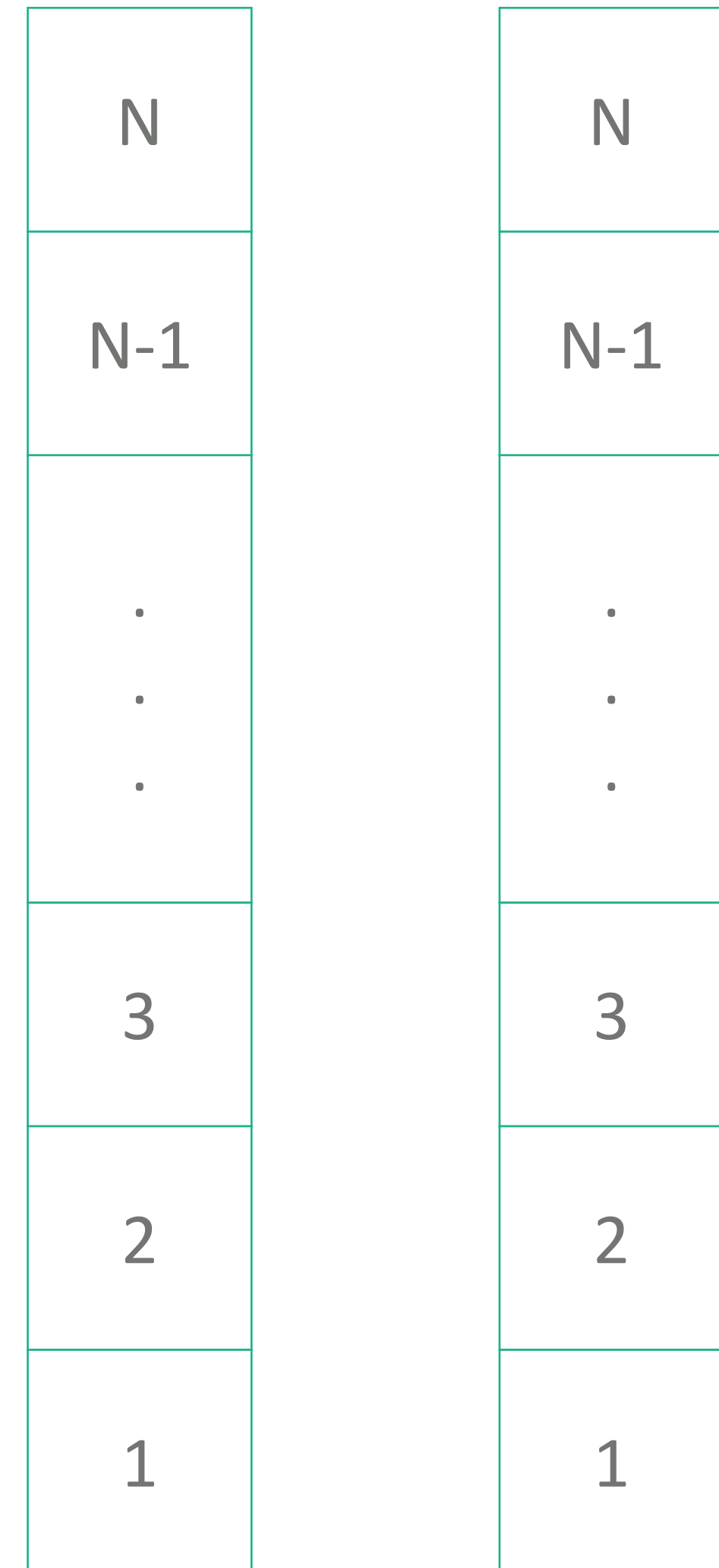


점프 게임

79

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

- 만약, 칸이 사라지는 조건이 없으면, BFS로 해결할 수 있다.



점프 게임

80

<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

81

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

- 소스: <http://codeplus.codes/4ecce30d648d4f75b7c68b4650a62ba9>

