

재귀
비트마스크
순열

Brute Force
Backtracking
BFS

완전 탐색 1

최백준 choi@startlink.io

완전 탐색

완전 탐색

Exhaustive Search

- 가능한 모든 경우의 수를 만들어보고 탐색하는 방법
- 가능한 모든 경우의 수를 알아야 한다.

그냥 다 해보기

그냥 다 해보기

Exhaustive Search

5

- 가능한 모든 경우의 수를 만들어보고 탐색하는 방법
- 가능한 모든 경우의 수를 알아야 한다.

날짜 계산

<https://www.acmicpc.net/problem/1476>

- 준규가 사는 나라는 E S M이라는 연도를 사용한다.

- $1 \leq E \leq 15, 1 \leq S \leq 28, 1 \leq M \leq 19$

- 1년 = 1 1 1

- 2년 = 2 2 2

- ...

- 15년 = 15 15 15

- 16년 = 1 16 16

- E S M이 주어졌을 때, 이게 몇 년인지 구하는 문제

1	2	1	1
18	3	18	18
19	4	19	19
20	5	20	1

$$15 \times 28 \times 19 = 7980$$

날짜 계산

7

<https://www.acmicpc.net/problem/1476>

- 가능한 경우의 수
- $15 \times 28 \times 19$
- 모든 경우를 for loop을 이용해서 해보면 된다

~~15~~ 15 → a
~~28~~ 28 → b
~~19~~ 19 → c

$O(1)$

날짜 계산

<https://www.acmicpc.net/problem/1476>

- C++: <https://gist.github.com/Baekjoon/b6333b742b3042619dbb>
- Java: <https://gist.github.com/Baekjoon/9c1b1a4e7245cb8858879234fa96fe4a>

리모컨

<https://www.acmicpc.net/problem/1107>

- TV 채널을 리모컨을 이용해 바꾸는 문제
- 버튼: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -
- 일부 숫자 버튼은 망가져있다
- 현재 보고 있는 채널: 100
- 수빈이가 이동하려고 하는 채널: N
- 이 때, 리모컨 버튼을 누르는 횟수를 최소로 하는 문제

5750 5749 +
 5751 -

① 숫자 +- ... 숫자 +-
+- +- - 숫자 -

② +- 변경
++ +- +-

→ ① 숫자 변경 0 (현재 채널 고정)
→ ② +- -2 누른2

리모컨

<https://www.acmicpc.net/problem/1107>

- 숫자 버튼을 이용해 채널 C로 이동한 다음
- 거기서 +나 -버튼을 몇 번 눌러야하는지 계산을 해본다
- 가능한 M의 개수: 500,000개
- +나 -를 누르는 횟수 계산은 뱌셈으로 한 번에 구할 수 있다

리모컨

<https://www.acmicpc.net/problem/1107>

1. 이동할 채널 C 를 정한다
2. C 에 포함되어있는 숫자 중에 망가진 버튼이 있는지 확인한다
3. 망가진 버튼이 없다면 $|C-N|$ 을 계산해 +나 - 버튼을 총 몇 번 눌러야 하는지를 계산한다

리모컨

<https://www.acmicpc.net/problem/1107>

1. 이동할 채널 C를 정한다
2. C에 포함되어있는 숫자 중에 망가진 버튼이 있는지 확인한다
3. 망가진 버튼이 없다면 $|C-N|$ 을 계산해 +나 - 버튼을 총 몇 번 눌러야 하는지를 계산한다

```
for (int i=0; i<=500000; i++) {  
    int c = i;  
}
```

0 $\xrightarrow{+}$ 50000 $\xleftarrow{-}$ 50000

1) 370 $\rightarrow /10$
 $\%10=0$
2) 37 $\rightarrow /10$
 $\%10=7$
3) 3 $\rightarrow /10$
 $\%10=3$
4) 0 $\rightarrow /10$
 $\%10=0$

리모컨

13

<https://www.acmicpc.net/problem/1107>

$$0 \leq C \leq 50만$$

1. 이동할 채널 C를 정한다
2. C에 포함되어있는 숫자 중에 망가진 버튼이 있는지 확인한다
3. 망가진 버튼이 없다면 $|C-N|$ 을 계산해 +나 - 버튼을 총 몇 번 눌러야 하는지를 계산한다

`bool broken[10];` // 버튼이 망가져 있으면 `true`, 아니면 `false`

```
bool possible(int c) {  
    while (c > 0) {  
        if (broken[c % 10]) return false;  
        c /= 10;  
    }  
    return true;  
}
```

C가 0 인데
0 번이 망가진 경우

리모컨

<https://www.acmicpc.net/problem/1107>

- C에 포함되어있는 숫자 중에 망가진 버튼이 있는지 확인한다
- possible(0) 은 항상 true를 리턴한다.

```
bool broken[10]; // 버튼이 망가져 있으면 true, 아니면 false
bool possible(int c) {
    while (c > 0) {
        if (broken[c % 10]) return false;
        c /= 10;
    }
    return true;
}
```

리모컨

<https://www.acmicpc.net/problem/1107>

- 0인 경우를 처리하는 코드

```
if (c == 0) {  
    if (broken[0]) {  
        return false;  
    } else {  
        return true;  
    }  
}
```

리모컨

<https://www.acmicpc.net/problem/1107>

- possible을 불가능하면 0, 가능하면 버튼을 눌러야 하는 횟수를 리턴하게 변경

```
int possible(int c) {  
    if (c == 0) {  
        return broken[0] ? 0 : 1;  
    }  
    int len = 0;  
    while (c > 0) {  
        if (broken[c % 10]) return 0;  
        len += 1;  
        c /= 10;  
    }  
    return len;  
}
```


리모컨

<https://www.acmicpc.net/problem/1107>

- 가장 처음에 보고 있는 채널은 100이기 때문에
- 초기값을 100에서 숫자 버튼을 누르지 않고 이동하는 횟수로 지정

101번

|N-100|

리모컨

<https://www.acmicpc.net/problem/1107>

- 가장 처음에 보고 있는 채널은 100이기 때문에
- 초기값을 100에서 숫자 버튼을 누르지 않고 이동하는 횟수로 지정

채널 변경 횟수

문제는

사용 가능 : 1, 5

$0 \leq N \leq 5000$

155555 → 500000 ← 511111

리모컨

<https://www.acmicpc.net/problem/1107>

- C++: <https://gist.github.com/Baekjoon/2c32984ad5c42b333c38>

N중 for문

N중 for문

for

- N개 중에 일부를 선택해야 하는 경우에 많이 사용한다
- 재귀 호출이나 비트마스크를 사용하면 더 간결하고 보기 쉬운 코드를 작성할 수 있기 때문에, 사용할 일이 거의 없다.

1, 2, 3 더하기

<https://www.acmicpc.net/problem/9095>

- 정수 n 을 1, 2, 3의 조합으로 나타내는 방법의 수를 구하는 문제

$$n \leq 10$$

- $n = 4$
- $1+1+1+1$
- $1+1+2$
- $1+2+1$
- $2+1+1$
- $2+2$
- $1+3$
- $3+1$

$$\underbrace{0 + 0 + 0 + \dots + 0 = n}_{10개}$$

1, 2, 3 더하기

<https://www.acmicpc.net/problem/9095>

23

- N이 10보다 작거나 같기 때문에
- 최대 10개 이하로 표현 가능
- $1+1+1+1+1+1+1+1+1+1$
- 10중 for문!

1, 2, 3 더하기

<https://www.acmicpc.net/problem/9095>

```
for (int l1=1; l1<=3; l1++) {  
    if (l1 == n) ans += 1;  
    for (int l2=1; l2<=3; l2++) {  
        if (l1+l2 == n) ans += 1;  
        ... 생략  
        for (int l0=1; l0<=3; l0++) {  
            if (l1+l2+l3+l4+l5+l6+l7+l8+l9+l0 == n) {  
                ans += 1;  
            }  
        }  
    }  
}
```


1, 2, 3 더하기

25

<https://www.acmicpc.net/problem/9095>

- C++: <https://gist.github.com/Baekjoon/281372b0f3900d333101>
- Java: <https://gist.github.com/Baekjoon/77c4ceb5a6da881e0bd4>

순열 사용하기

//

$$\text{DFS 순열} : O(N \cdot N!)$$

$$N = 10$$

$$10 - 10!$$

$$= 36,288,000$$

$$\boxed{N \leq 10} \quad N!$$

팩토리얼

Factorial

- $3! = 6$
- $4! = 24$
- $5! = 120$
- $6! = 720$
- $7! = 5,040$
- $8! = 40,320$
- $9! = 362,880$
- $10! = 3,628,800$
- $11! = 39,916,800$
- $12! = 479,001,600$
- $13! = 6,227,020,800$

차이를 최대로

<https://www.acmicpc.net/problem/10819>

- 수 N 개가 주어졌을 때 ($3 \leq N \leq 8$)
- $|A[0] - A[1]| + |A[1] - A[2]| + \dots + |A[N-2] - A[N-1]|$
- 를 최대로 하는 문제

$N!$

$8! = 40320$

차이를 최대로

<https://www.acmicpc.net/problem/10819>

- $N! = 8! = 40320$
- 모든 경우를 다해봐도 된다.
- 수를 `next_permutation`을 이용해 모든 경우를 다 해본다

차이를 최대로

<https://www.acmicpc.net/problem/10819>

```
do {  
    int temp = calculate(a);  
    ans = max(ans, temp);  
} while(next_permutation(a.begin(), a.end()));
```

차이를 최대로

<https://www.acmicpc.net/problem/10819>

- C++: <https://gist.github.com/Baekjoon/fb602ec4b6778757d717>
- Java: <https://gist.github.com/Baekjoon/1b03fb9b88d7de8cd959>

외판원 순회 2

<https://www.acmicpc.net/problem/10971>

$N!$

- 영어로 Travelling Salesman Problem (TSP)
- 1번부터 N번까지 번호가 매겨져있는 도시가 있다
- 한 도시에서 시작해 N개의 모든 도시를 거쳐 다시 원래 도시로 돌아오려고 한다 (한 번 갔던 도시로는 다시 갈 수 없다)
- 이 때, 가장 적은 비용을 구하는 문제

- $W[i][j] = i \rightarrow j$ 비용

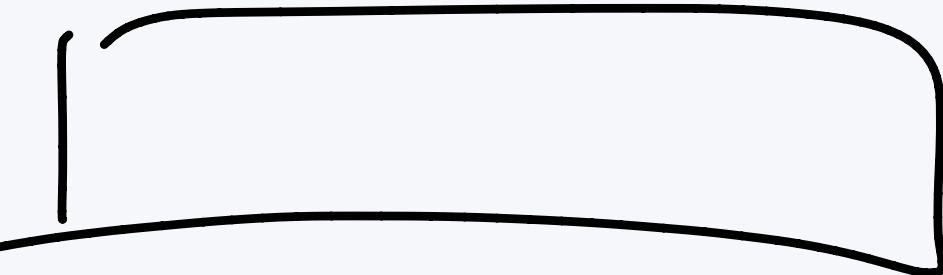


$W[i][j]$

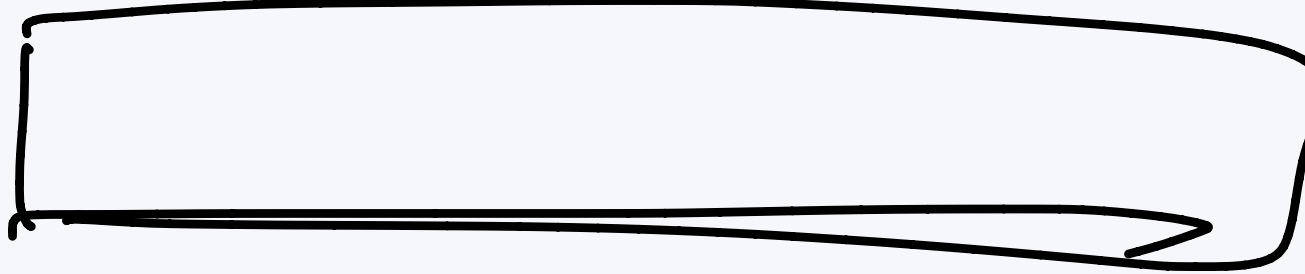
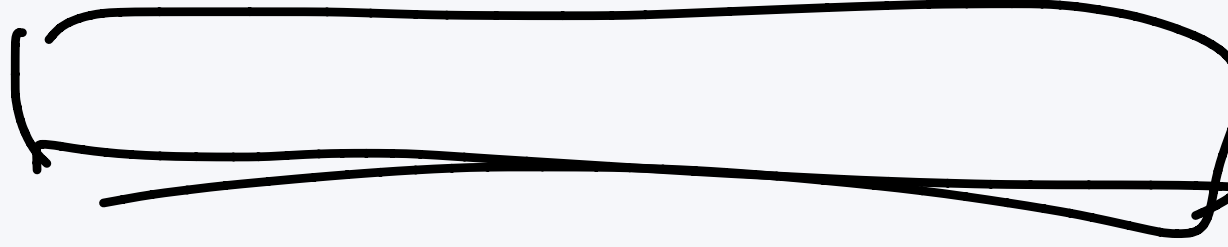

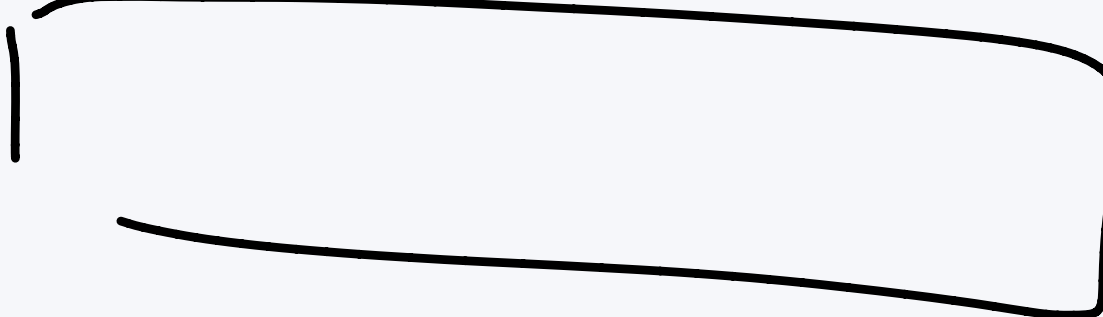
$i \Rightarrow j$ 비용

외판원 순회 2

<https://www.acmicpc.net/problem/10971>

- $2 \leq N \leq 10$
- $N! = 10! = 3628800$
- 모든 경우를 다해봐도 시간 안에 나온다

1 2 
1 3 
⋮
1 N 

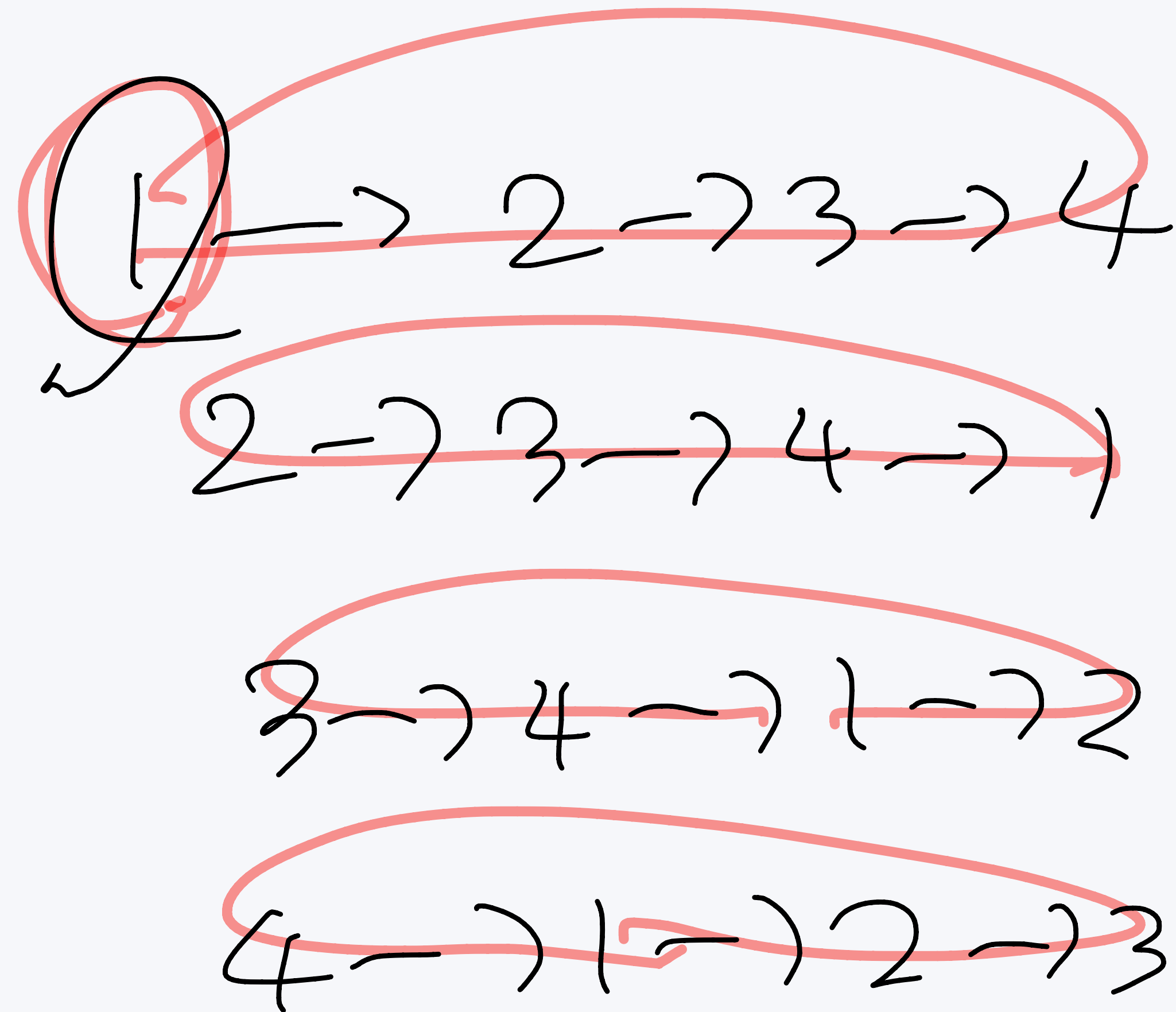
1 
2 
3 
4 

외판원 순회 2

34

<https://www.acmicpc.net/problem/10971>

- $2 \leq N \leq 10$
- $N! = 10! = 3628800$
- 모든 경우를 다해봐도 시간 안에 나온다
- 모든 경우 = $N!$
 - 비용 계산 = N
- 시간복잡도: $O(N * N!)$



외판원 순회 2

<https://www.acmicpc.net/problem/10971>

$w[i][j] = 0$

$i \rightarrow j$

35

```
do {  
    if (d[0] == 1) break;  
    bool ok = true;  
    int sum = 0;  
    for (int i=0; i<n-1; i++) {  
        if (w[d[i]][d[i+1]] == 0) ok = false;  
        else sum += w[d[i]][d[i+1]];  
    }  
    if (ok && w[d[n-1]][d[0]] != 0) {  
        sum += w[d[n-1]][d[0]];  
        if (ans > sum) ans = sum;  
    }  
} while (next_permutation(d.begin(), d.end()));
```

시간 복잡도: $O(NN!)$

↓

① $(N - (N-1)!) = 1$

$O(N!)$

+1

②

외판원 순회 2

<https://www.acmicpc.net/problem/10971>

$N \leq 16$

36

- $O(N \cdot N!)$
- C++: <https://gist.github.com/Baekjoon/a62f0b1263752c8d1a75>
- Java: <https://gist.github.com/Baekjoon/a5450f44bc19da72f9ac>
- $O(N!)$
- C++: <https://gist.github.com/Baekjoon/3eeee9003b22cffb2a76>
- C++ 2: <https://gist.github.com/Baekjoon/45c47a211c3be61e054a>
- Java: <https://gist.github.com/Baekjoon/88bfb6c2e54bb399beb2>

16!

<https://www.acmicpc.net/problem/6603>

조합

- 배열에 1, 1, 2, 2, 2를 넣고 next_permutation을 수행하면 어떻게 될까?

1 1 2 2 2

1 2 1 2 2

1 2 2 1 2

$N \geq k$

$\frac{N!}{(N-k)!}$

$\frac{1}{k}$

로또

<https://www.acmicpc.net/problem/6603>

- 1 1 2 2 2
- 1 2 1 2 2
- 1 2 2 1 2
- 1 2 2 2 1
- 2 1 1 2 2
- 2 1 2 1 2
- 2 1 2 2 1
- 2 2 1 1 2
- 2 2 1 2 1
- 2 2 2 1 1

<https://www.acmicpc.net/problem/6603>

- 0을 K-6개, 1을 6개를 넣은 다음에 next_permutation 를 수행하면 조합 모든 조합을 구할 수 있다

<https://www.acmicpc.net/problem/6603>

- C++: <https://gist.github.com/Baekjoon/b8578f48f40ed91d3d083527df181ee8>

BFS

큐 사용하기

- ① 상태의 개수가 탐색가능
- ② 상태 \Rightarrow 상태
가중치가 1
- ③ 최소 비용을 구하는 문제

모든 가중치가 1 이면 최단 거리 알고리즘이

BFS

숨바꼭질

<https://www.acmicpc.net/problem/1697>

- 수빈이의 위치: N
- 동생의 위치: K
- 동생을 찾는 가장 빠른 시간을 구하는 문제
- 수빈이가 할 수 있는 행동 (위치: X)

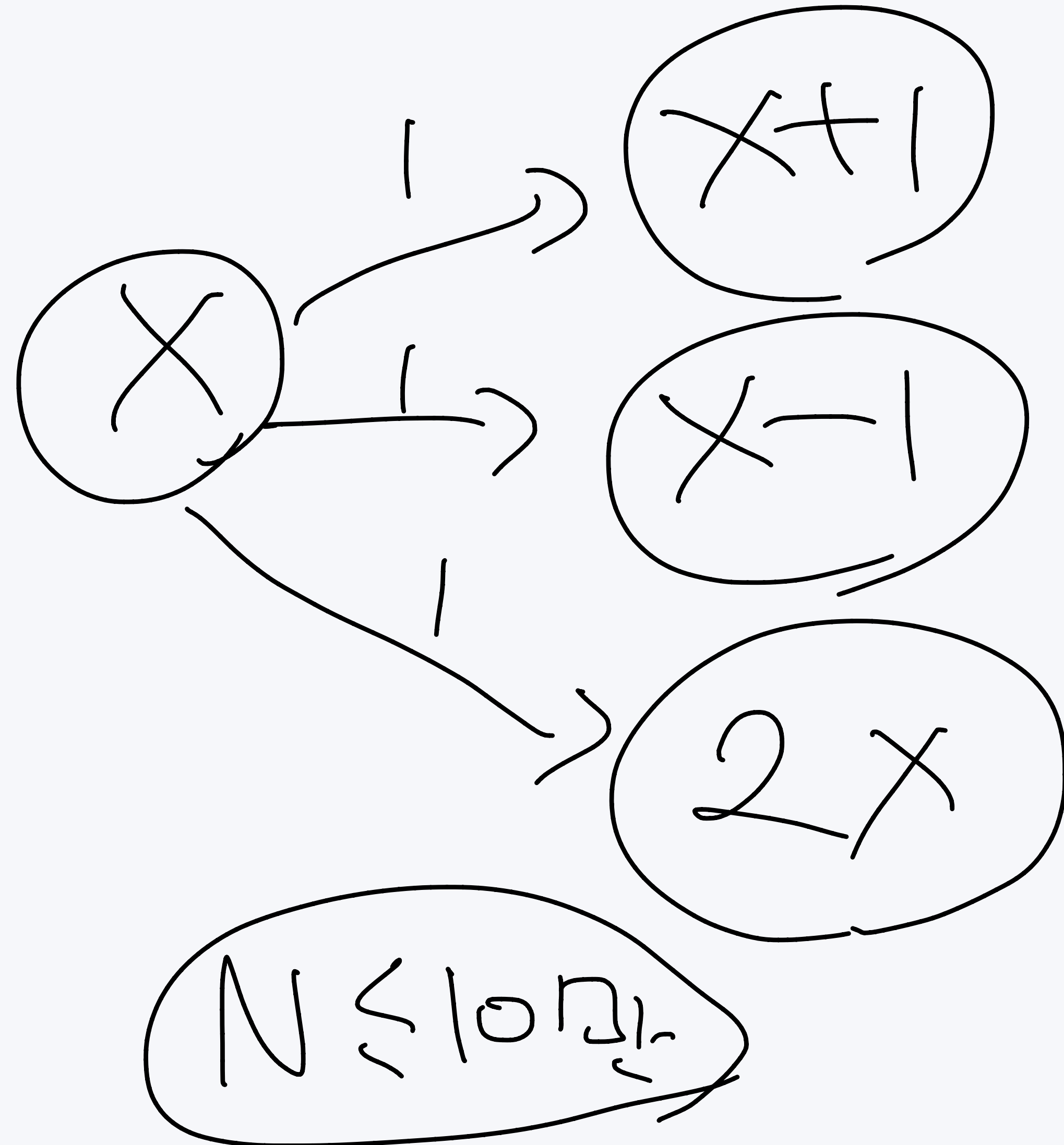
1. 걷기: $X+1$ 또는 $X-1$ 로 이동
2. 순간이동: $2 \times X$ 로 이동

1초

[차원 과포워(으)]

~~(V+E)~~

42



숨바꼭질

43

<https://www.acmicpc.net/problem/1697>

- 수빈이의 위치: 5
- 동생의 위치: 17
- 5-10-9-18-17 로 4초만에 동생을 찾을 수 있다.

숨바꼭질

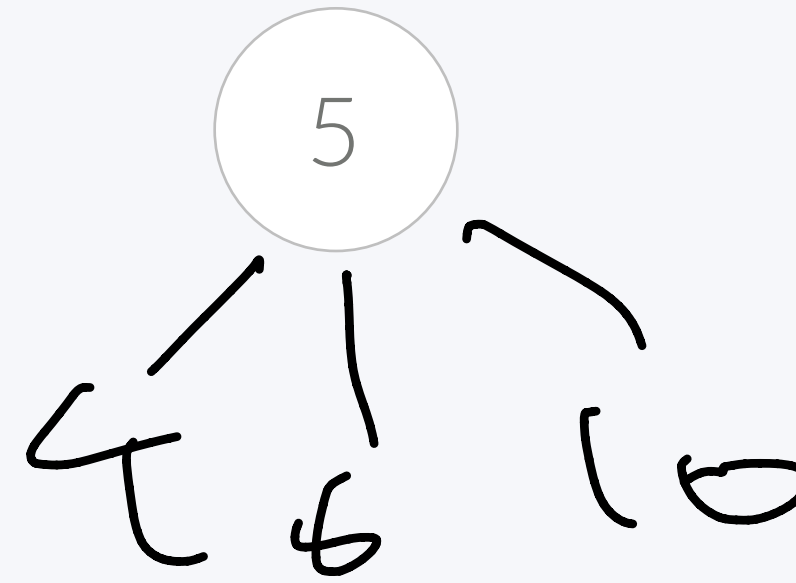
<https://www.acmicpc.net/problem/1697>

- 큐에 수빈이의 위치를 넣어가면서 이동시킨다
- 한 번 방문한 곳은 다시 방문하지 않는 것이 좋기 때문에, 따로 배열에 체크하면서 방문

숨바꼭질

<https://www.acmicpc.net/problem/1697>

- 가장 처음
- Queue: 5

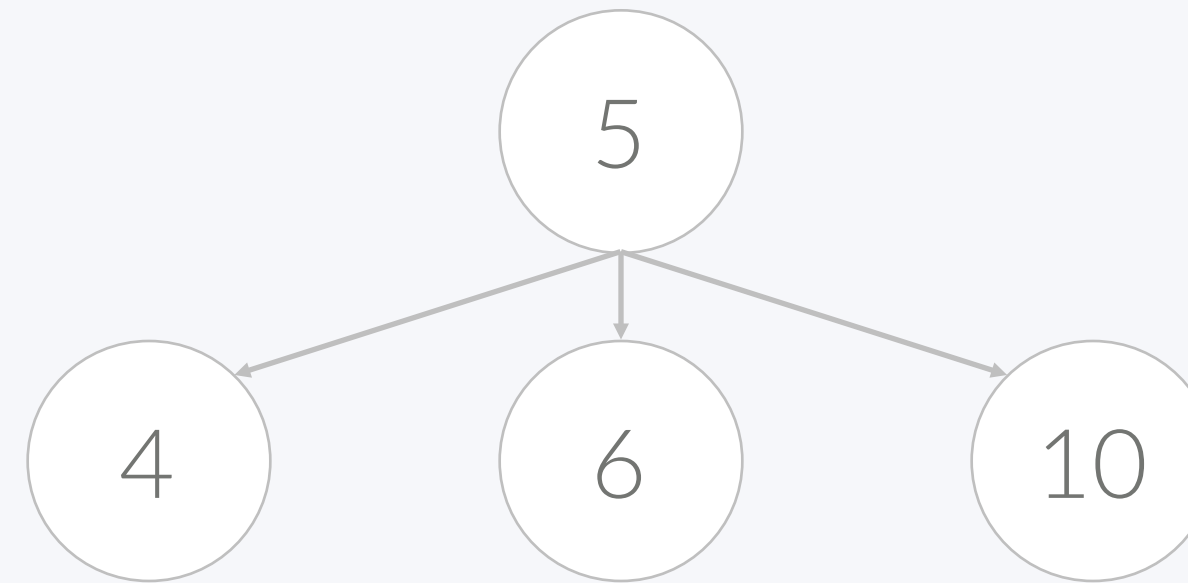


$$\begin{array}{l} x \rightarrow x+1 \\ \quad x-1 \\ \quad \quad 2x \end{array}$$

숨바꼭질

<https://www.acmicpc.net/problem/1697>

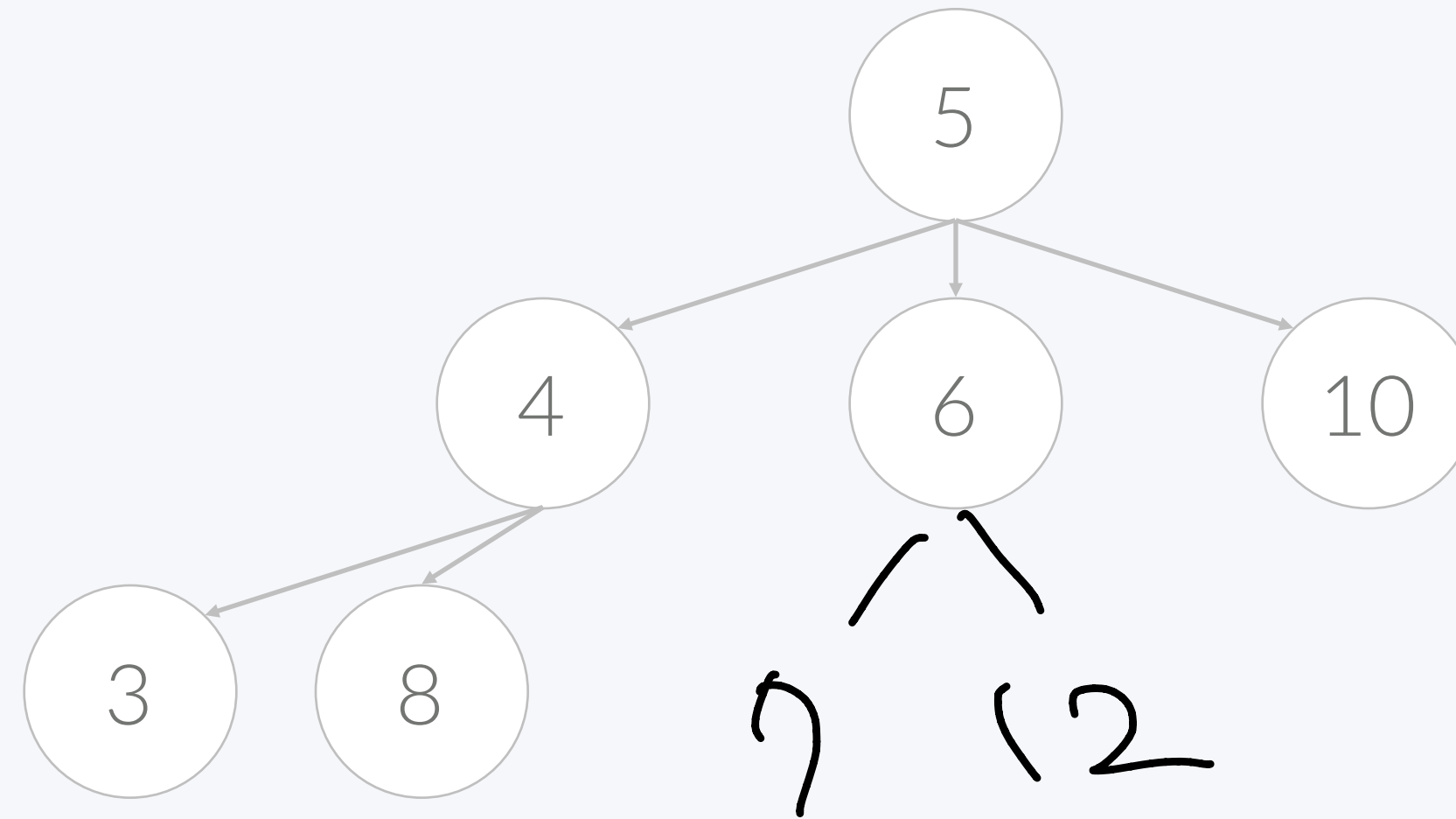
- 5에서 이동
- Queue: 5 4 6 10



숨바꼭질

<https://www.acmicpc.net/problem/1697>

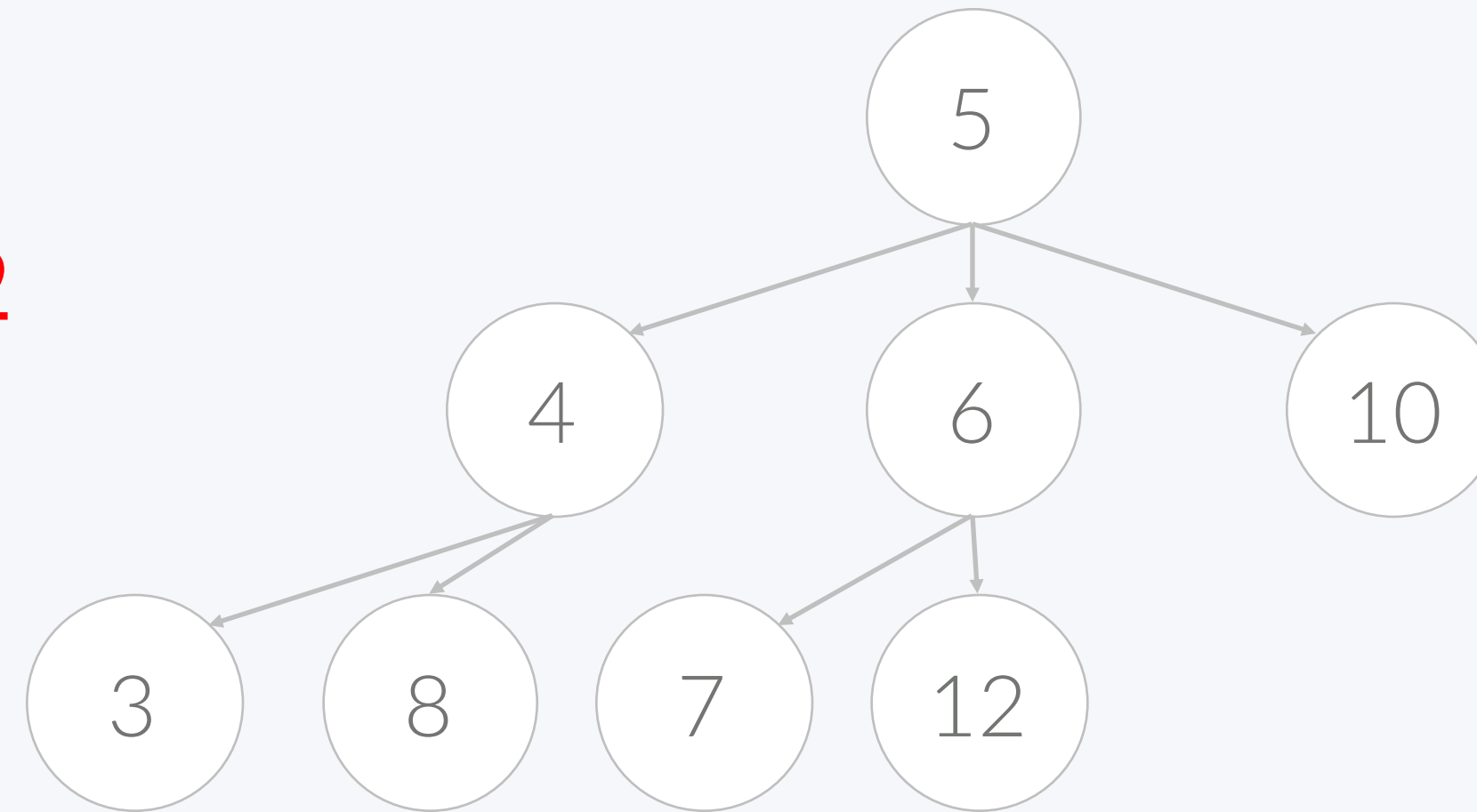
- 4에서 이동
- Queue: 5 4 6 10 3 8



숨바꼭질

<https://www.acmicpc.net/problem/1697>

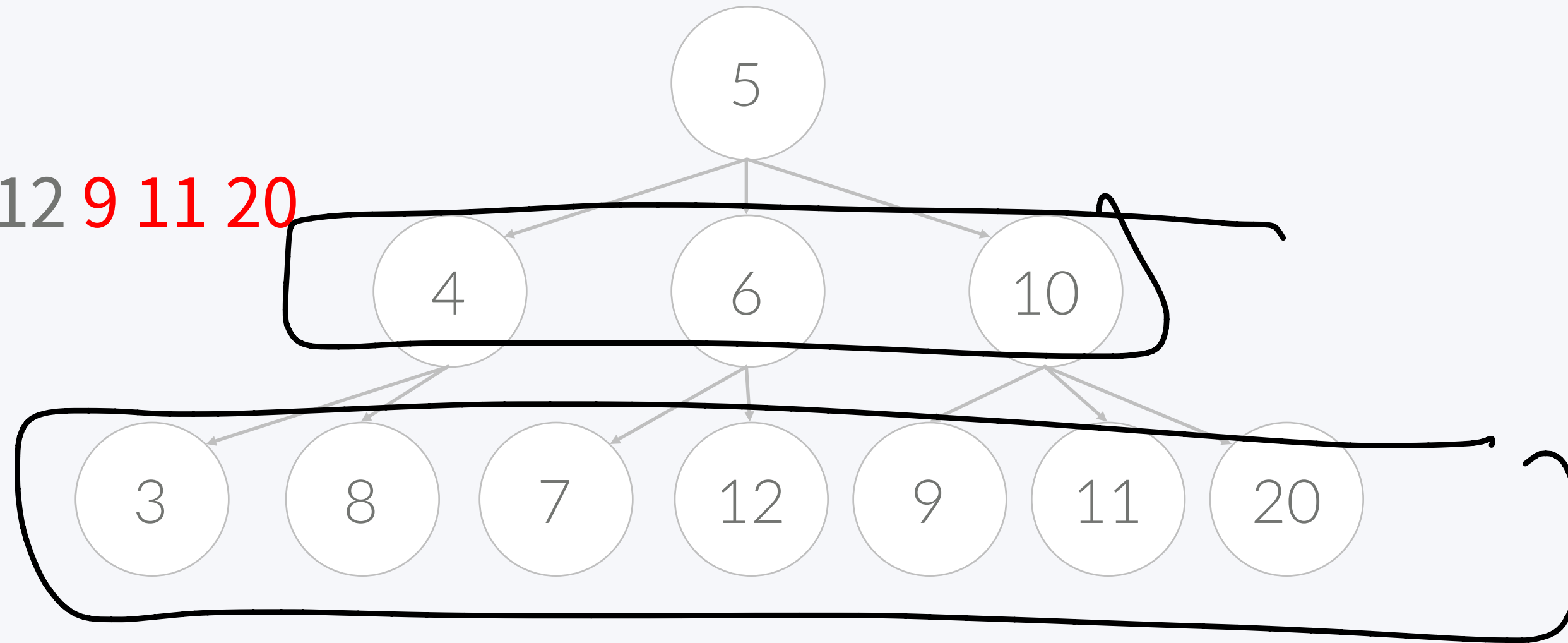
- 6에서 이동
- Queue: 5 4 6 10 3 8 7 12



숨바꼭질

<https://www.acmicpc.net/problem/1697>

- 10에서 이동
- Queue: 5 4 6 10 3 8 7 12 9 11 20

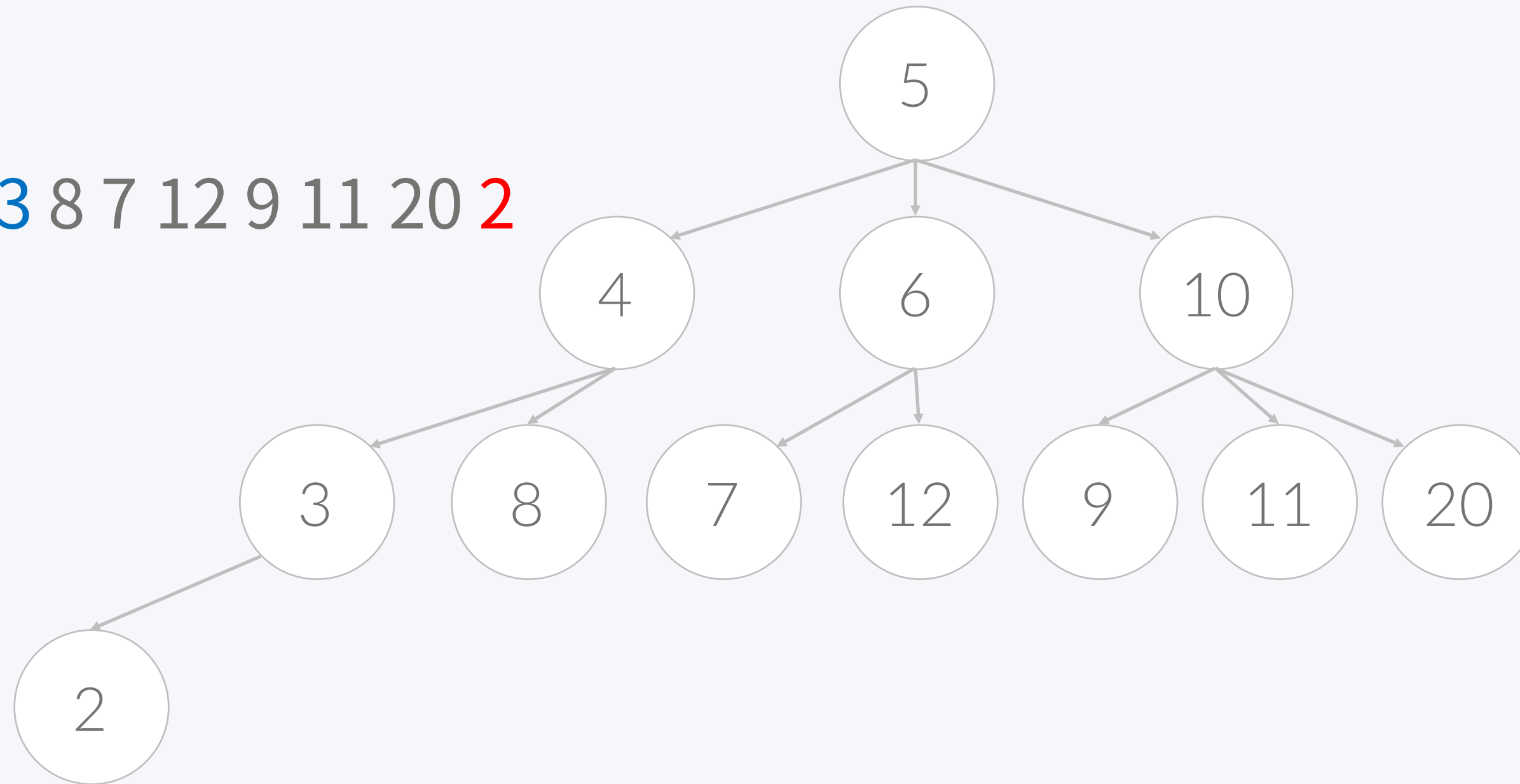


숨바꼭질

50

<https://www.acmicpc.net/problem/1697>

- 3에서 이동
- Queue: 5 4 6 10 3 8 7 12 9 11 20 2



숨바꼭질

<https://www.acmicpc.net/problem/1697>

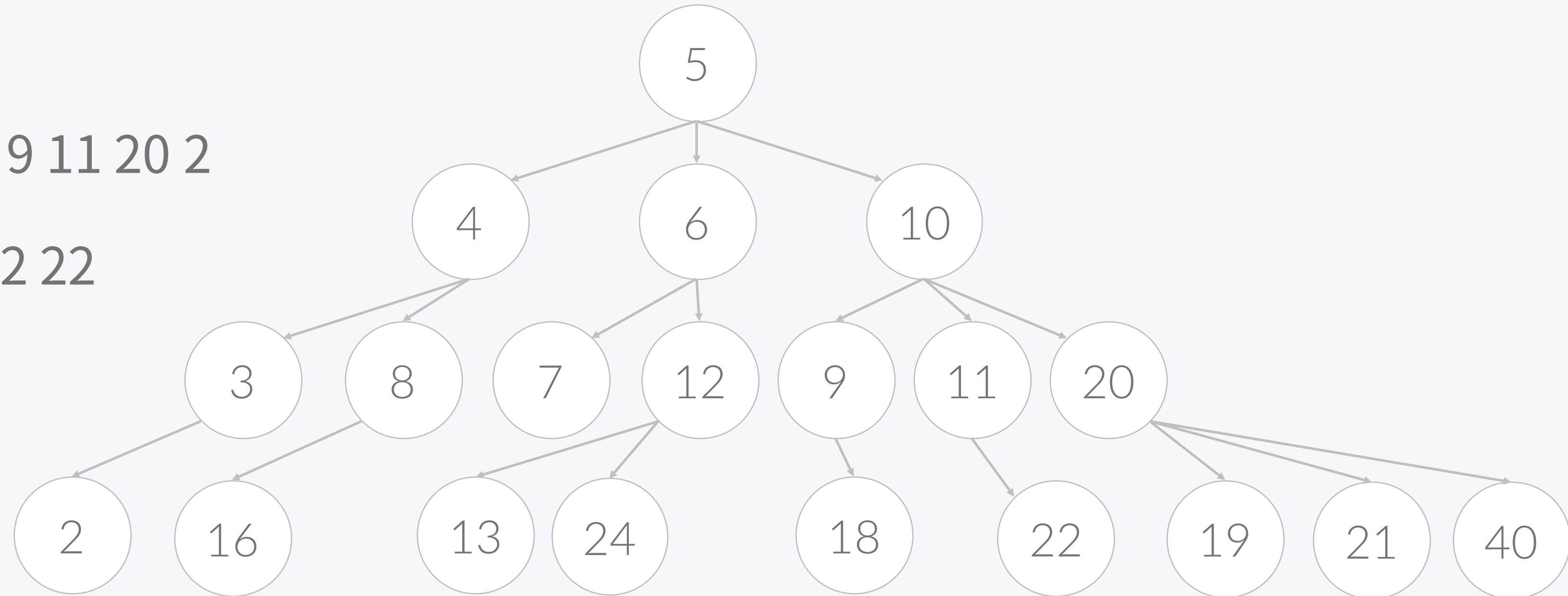
- 이런식으로...

숨바꼭질

52

<https://www.acmicpc.net/problem/1697>

- Queue:
- 5 4 6 10 3 8 7 12 9 11 20 2
- 16 13 15 28 18 12 22
- 19 21 40



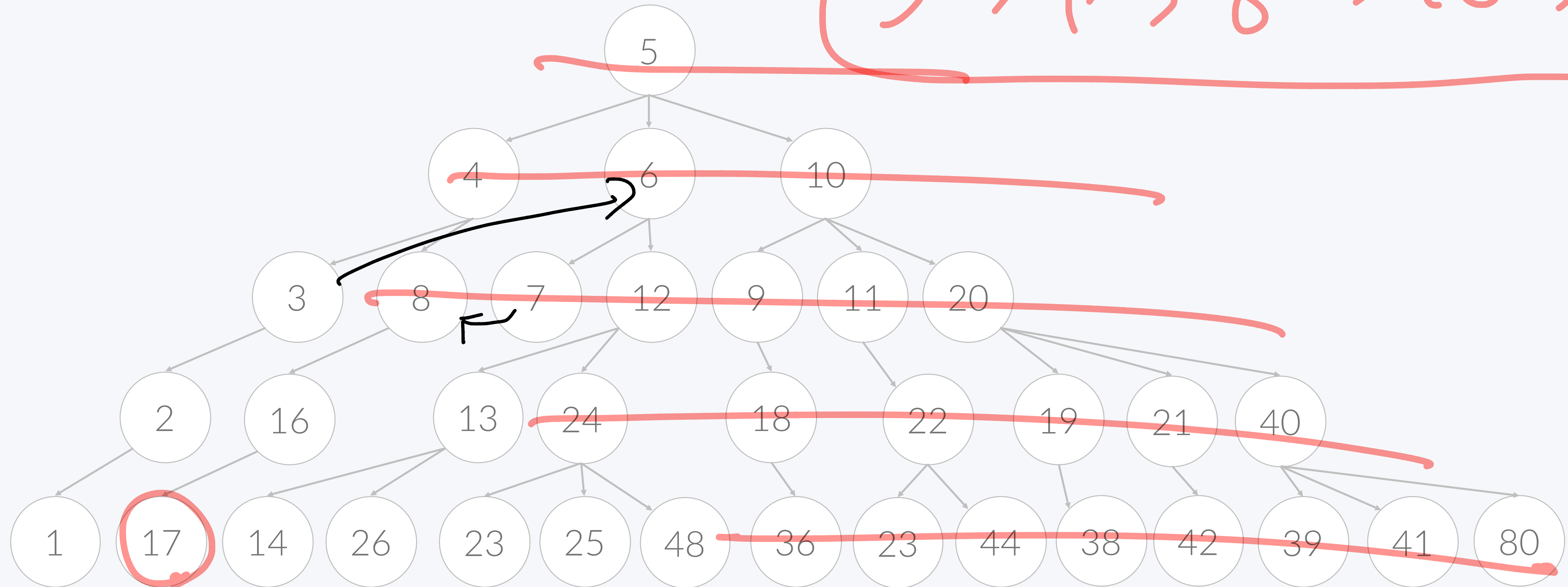
숨바꼭질

<https://www.acmicpc.net/problem/1697>

53

•

5 \Rightarrow 17
5 \rightarrow 4 \rightarrow 8 \rightarrow 16 \rightarrow 17

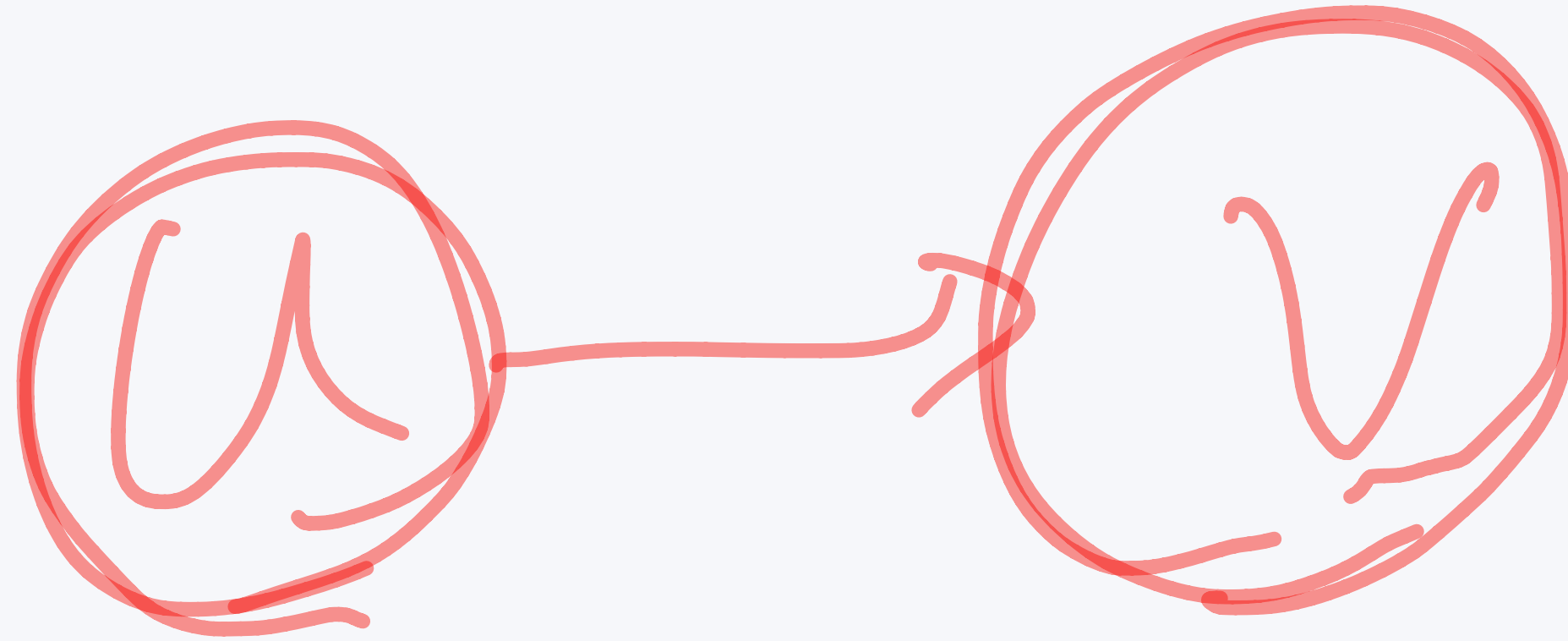


숨바꼭질

<https://www.acmicpc.net/problem/1697>

- $check[i]$ = i 를 방문했는지
- $dist[i]$ = i 를 몇 번만에 방문했는지

시작 \Rightarrow 거리
 $dist$



$$dist[v] = dist[u] + 1$$

숨바꼭질

<https://www.acmicpc.net/problem/1697>

55

dist

```
check[n] = true;
dist[n] = 0;
queue<int> q;
q.push(n);
```

```
while (!q.empty()) {
    int now = q.front();
    q.pop();
```

```
    if (now-1 >= 0) {
        if (check[now-1] == false) {
            q.push(now-1);
            check[now-1] = true;
            dist[now-1] = dist[now] + 1;
        }
    }
}
```

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

```
    if (now*2 < MAX) {
        if (check[now*2] == false) {
            q.push(now*2);
            check[now*2] = true;
            dist[now*2] = dist[now] + 1;
        }
    }
```

숨바꼭질

<https://www.acmicpc.net/problem/1697>

- $Check[i]$ = i 를 방문했는지
- $D[i]$ = i 를 몇 번만에 방문했는지



① $check[v] = \text{false}$

② $check[v] = \text{true}$ 22
 $dist[v] = dist[u] + 1$

숨바꼭질

<https://www.acmicpc.net/problem/1697>

- C++: <https://gist.github.com/Baekjoon/5b8924d3aec661746358>
- Java: <https://gist.github.com/Baekjoon/e1abcf6deb6796c1282a>

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

소수 경로

59

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

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

소수 경로

60

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

- 이 문제도 숨바꼭질 문제와 비슷하게
- Queue를 이용해서 풀 수 있다

소수 경로

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

```
while (!q.empty()) {
    int now = q.front(); q.pop();
    for (int i=0; i<4; i++) {
        for (int j=0; j<=9; j++) {
            int next = change(now, i, j);
            if (next != -1) {
                if (prime[next] && c[next] == false) {
                    q.push(next);
                    d[next] = d[now] + 1;
                    c[next] = true;
                }
            }
        }
    }
}
```

소수 경로

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

- C++: <https://gist.github.com/Baekjoon/374bc605f358e0a2cafd>
- Java: <https://gist.github.com/Baekjoon/d815d8eae0b586537a02>

DSLR

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

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

• D: $N \rightarrow 2 \times N$ 1234 → 2468

• S: $N \rightarrow N - 1$ 1234 → 1233

• L: 한 자리씩 왼쪽으로

1234 → 234)

• R: 한 자리씩 오른쪽으로

1234 → 4123

1234 → 8246
2468

DIR

N → 2N
N → N-1

0000 ~ 9999

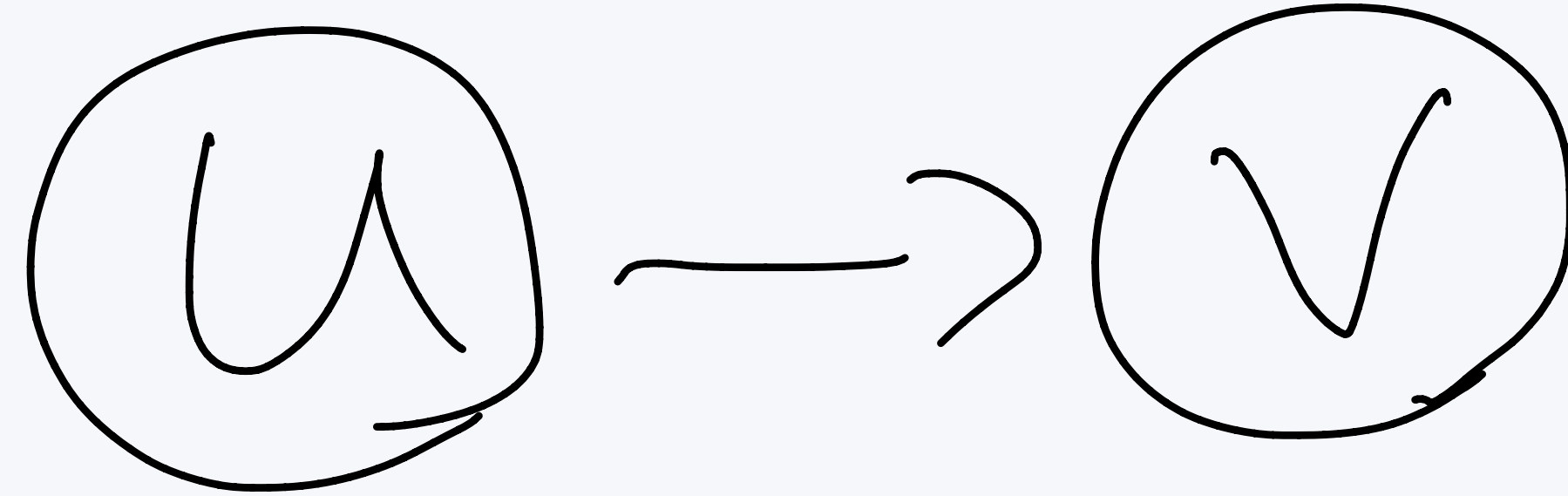
만가!

DSL R

64

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

- 앞의 두 문제와 동일하게 풀지만
- 이 문제는 최소값을 구해야 하는건 맞지만
- 어떠한 과정을 거쳐야 하는지를 구해야 한다
- 배열 두 개를 더 이용해서 어떤 과정을 거쳤는지를 저장해야 한다
- $from[i] = i$ 를 어떤 수에서 만들었는지
- $how[i] = i$ 를 어떻게 만들었는지



$$dist[V] = dist[U] + 1$$

$$from[V] = U;$$
$$how[V] = 'D', 'S', 'L', 'R'$$

DSLR

<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';  
}
```

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 R

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

1234 → 2341

67

```
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>

[234] → 4123

68

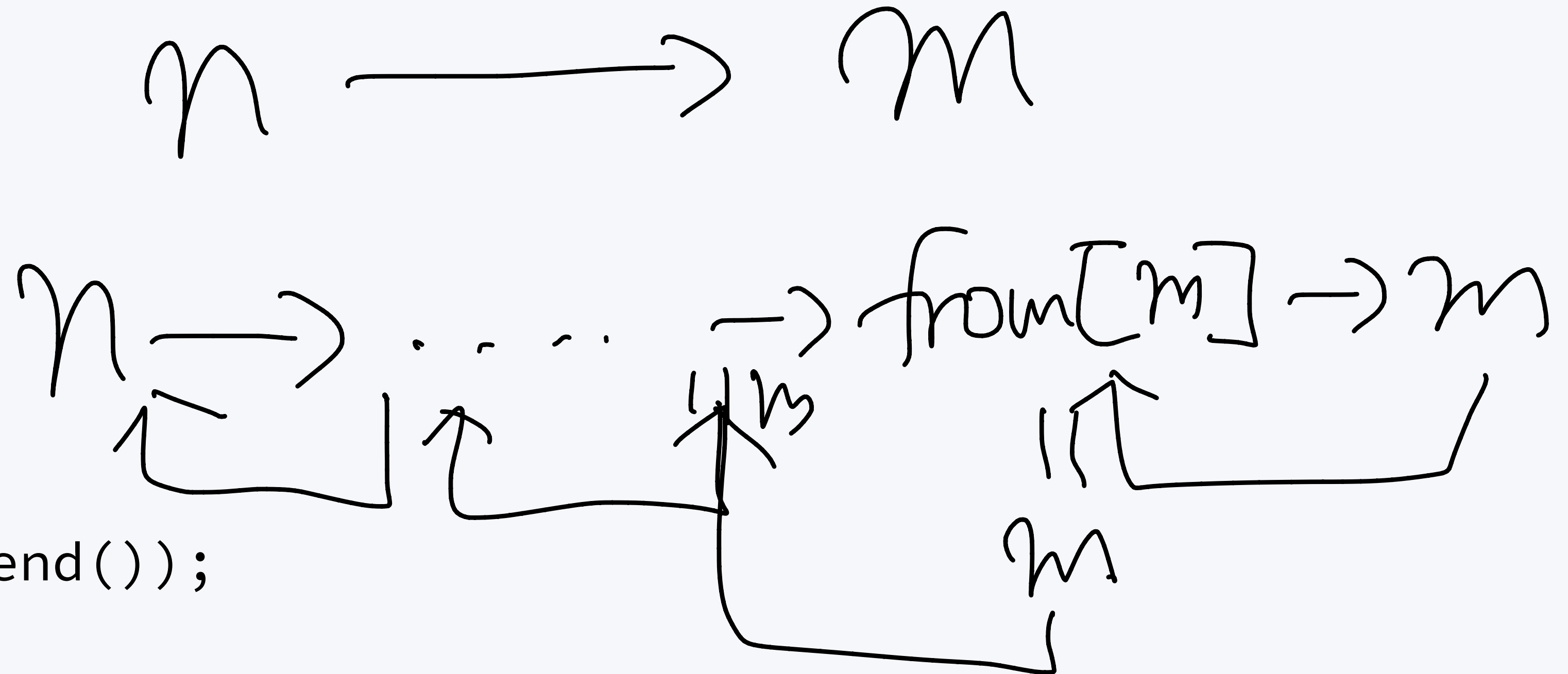
```
next = (now/10) + (now%10)*1000;  
if (check[next] == false) {  
    q.push(next);  
    check[next] = true;  
    dist[next] = dist[now]+1;  
    from[next] = now;  
    how[next] = 'R';  
}
```

DSLR

69

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

```
string ans = "";  
while (m != n) {  
    ans += how[m];  
    m = from[m];  
}  
reverse(ans.begin(), ans.end());  
cout << ans << '\\n';
```

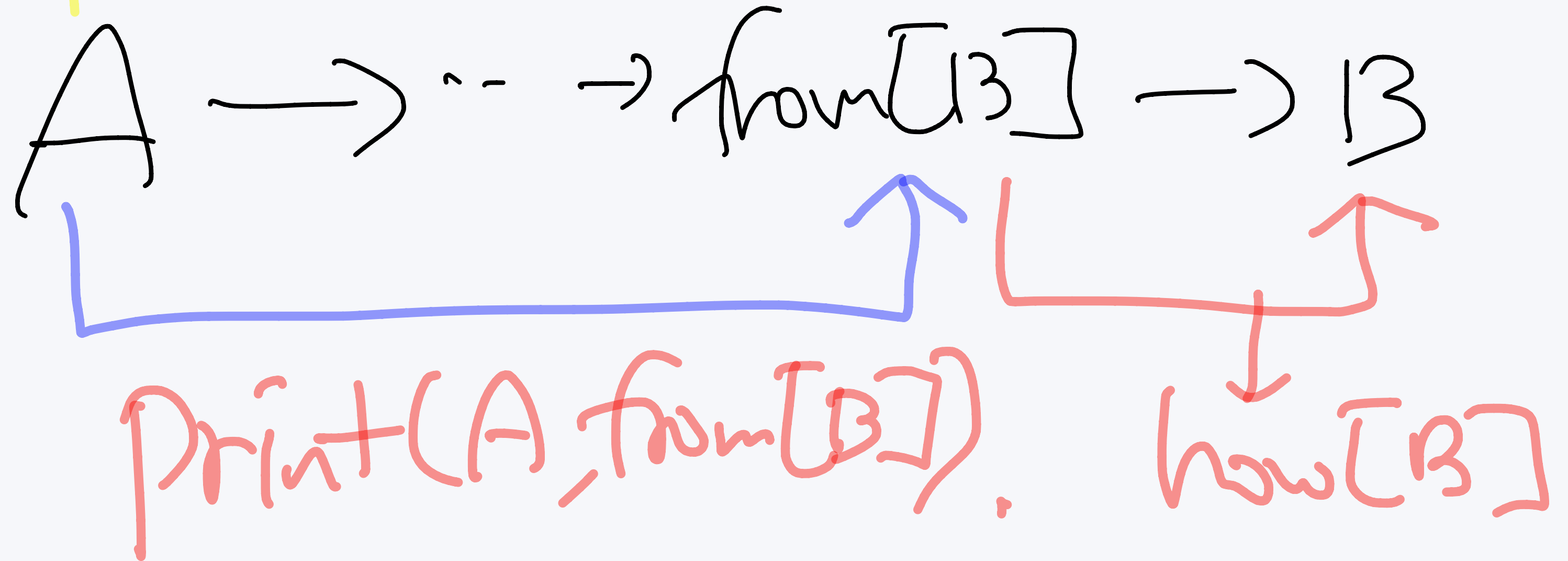


DSL

70

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

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



DSL R

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

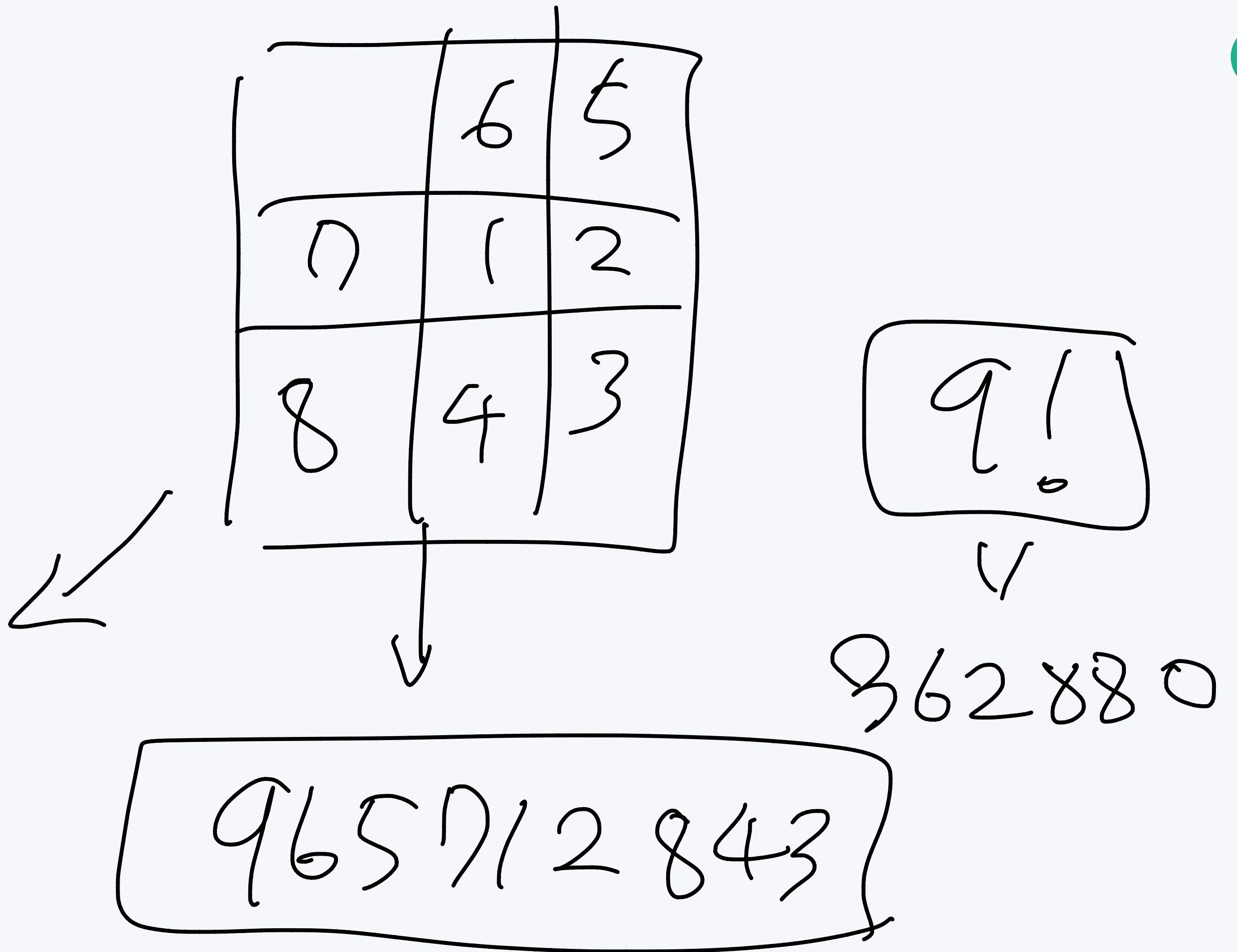
- C++: <https://gist.github.com/Baekjoon/05b6e7a9c0bf6d4742ab>
- C++: <https://gist.github.com/Baekjoon/82ed59e4713f5286001ddcbba644849b>
- Java: <https://gist.github.com/Baekjoon/8c41844868ecb0c58a44>

퍼즐

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

- 8퍼즐을 푸는 문제
- 배열로 상태를 저장할 수가 없다

1	2	3
4	5	6
7	8	



퍼즐

73

<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>`

퍼즐

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

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

퍼즐

75

<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>

- C++: <https://gist.github.com/Baekjoon/6fa3fdc760b4ffc95d75>
- Java: <https://gist.github.com/Baekjoon/1b496dcf92f128468aca>

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

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;
    }
}
```

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

- C++: <https://gist.github.com/Baekjoon/f1f802661214359bd2ff>
- C++: <https://gist.github.com/Baekjoon/68ea2da7363fc088972a>
- Java: <https://gist.github.com/Baekjoon/5c7aba583dd3bc298b5d>

더 풀어볼 문제

더 풀어볼 문제

- 스타트링크: <https://www.acmicpc.net/problem/5014>

재귀호출 사용하기

재귀함수 사용하기

Recursion

85

- 재귀함수를 잘 설계해야 한다

1, 2, 3 더하기

<https://www.acmicpc.net/problem/9095>

- 정수 n 을 1, 2, 3의 조합으로 나타내는 방법의 수를 구하는 문제
- $n = 4$
- $1+1+1+1$
- $1+1+2$
- $1+2+1$
- $2+1+1$
- $2+2$
- $1+3$
- $3+1$

1, 2, 3 더하기

<https://www.acmicpc.net/problem/9095>

- `go(count, sum, goal)`
- 숫자 count개로 합 sum을 만드는 경우의 수

① 불가능한 경우

$$\text{sum} > \text{goal}$$

② 정답을 찾은 경우

$$\text{sum} == \text{goal}.$$

③ 다른 호출

1이 오는 경우

`go(count + 1, sum + 1, goal)`

2가 오는 경우

`go(count + 1, sum + 2, goal)`

3이 오는 경우

`go(count + 1, sum + 3, goal)`

1, 2, 3 더하기

<https://www.acmicpc.net/problem/9095>

- `go(count, sum, goal)`
- 숫자 count개로 합 sum을 만드는 경우의 수
- 불가능한 경우
 - ~~sum < goal~~
 - `sum > goal`
- 가능한 경우
 - `sum == goal`

1, 2, 3 더하기

<https://www.acmicpc.net/problem/9095>

- `go(count, sum, goal)`
- 숫자 count개로 합 sum을 만드는 경우의 수
- 다음 경우
 - 1을 사용하는 경우
 - `go(count+1, sum+1, goal)`
 - 2를 사용하는 경우
 - `go(count+1, sum+2, goal)`
 - 3을 사용하는 경우
 - `go(count+1, sum+3, goal)`

1, 2, 3 더하기

<https://www.acmicpc.net/problem/9095>

```
int go(int count, int sum, int goal) {  
if (count > 10) return 0;  
    if (sum > goal) return 0;  
    if (sum == goal) return 1;  
    int now = 0;  
    for (int i=1; i<=3; i++) {  
        now += go(count+1, sum+i, goal);  
    }  
    return now;  
}
```

시간복잡도

(3^N)

1, 2, 3 더하기

91

<https://www.acmicpc.net/problem/9095>

- C++: <https://gist.github.com/Baekjoon/3235f76fe44c1ad17648>
- Java: <https://gist.github.com/Baekjoon/bdeba307e9e6d1e80fc7>

암호 만들기

<https://www.acmicpc.net/problem/1759>

- 암호는 서로 다른 L 개의 알파벳 소문자들로 구성되며 최소 한 개의 모음과 최소 두 개의 자음으로 구성되어 있다
- 암호를 이루는 알파벳이 암호에서 증가하는 순서로 배열되었어야 한다
- 암호로 사용할 수 있는 문자의 종류는 C 가지
- 가능성 있는 암호를 모두 구하는 문제

암호 만들기

<https://www.acmicpc.net/problem/1759>

- $L = 4, C = 6$

- 사용 가능한 알파벳: a t c i s w

- 가능한 암호

- acis

- acit

- aciw

- acst

- acsw

- actw

- aist

암호의 길이: 4

alpha

- aism

- aitm

- astm

- cism

- cism

- citm

- istm



암호 만들기

<https://www.acmicpc.net/problem/1759>

94

- `go(n, alpha, password, i)`

- `n`: 만들어야 하는 암호의 길이
- `alpha`: 사용할 수 있는 알파벳
- `password`: 현재까지 만든 암호

- `i`: 사용할지 말지 결정해야 하는 알파벳의 인덱스

① 정답을 찾은 경우

$n == \text{password.length}()$

② 불가능한 경우

= 사용할 수 있는 알파벳이

없음

$i \geq \text{alpha.size}()$

③ 다음 경우

사용한다 $\text{go}(n, \text{alpha}, \text{password} + \text{alpha}[i], i+1)$

사용하지 않는다 $\text{go}(n, \text{alpha}, \text{password}, i+1)$

암호 만들기

<https://www.acmicpc.net/problem/1759>

- `go(n, alpha, password, i)`
 - `n`: 만들어야 하는 암호의 길이
 - `alpha`: 사용할 수 있는 알파벳
 - `password`: 현재까지 만든 암호
 - `i`: 사용할지 말지 결정해야 하는 알파벳의 인덱스
- 언제 답인지 아닌지 확인해야 하나?
 - `n == password.length()`
- 다음
 - `i`번째 알파벳을 사용하는 경우
 - `i`번째 알파벳을 사용하지 않는 경우

암호 만들기

<https://www.acmicpc.net/problem/1759>

- 다음
 - i번째 알파벳을 사용하는 경우
 - `go(n, alpha, password+alpha[i], i+1)`
 - i번째 알파벳을 사용하지 않는 경우
 - `go(n, alpha, password, i+1)`

암호 만들기

<https://www.acmicpc.net/problem/1759>

```
void go(int n, vector<char> &alpha, string password, int i) {  
    if (password.length() == n) {  
        if (check(password)) {  
            cout << password << '\n';  
        }  
        return;  
    }  
    if (i >= alpha.size()) return;  
    go(n, alpha, password+alpha[i], i+1);  
    go(n, alpha, password, i+1);  
}
```

재귀함수

재귀 호출

암호 만들기

98

<https://www.acmicpc.net/problem/1759>

```
bool check(string &password) {  
    int ja = 0;  
    int mo = 0;  
    for (char x : password) {  
        if (x == 'a' || x == 'e' || x == 'i' || x == 'o' || x ==  
'u') {  
            mo += 1;  
        } else {  
            ja += 1;  
        }  
    }  
    return ja >= 2 && mo >= 1;  
}
```

Vowel
Co

암호 만들기

<https://www.acmicpc.net/problem/1759>

- C++: <https://gist.github.com/Baekjoon/dff42ddf0ae028f6b7f1>
- Java: <https://gist.github.com/Baekjoon/e92cfec2c020cd62b8ef>

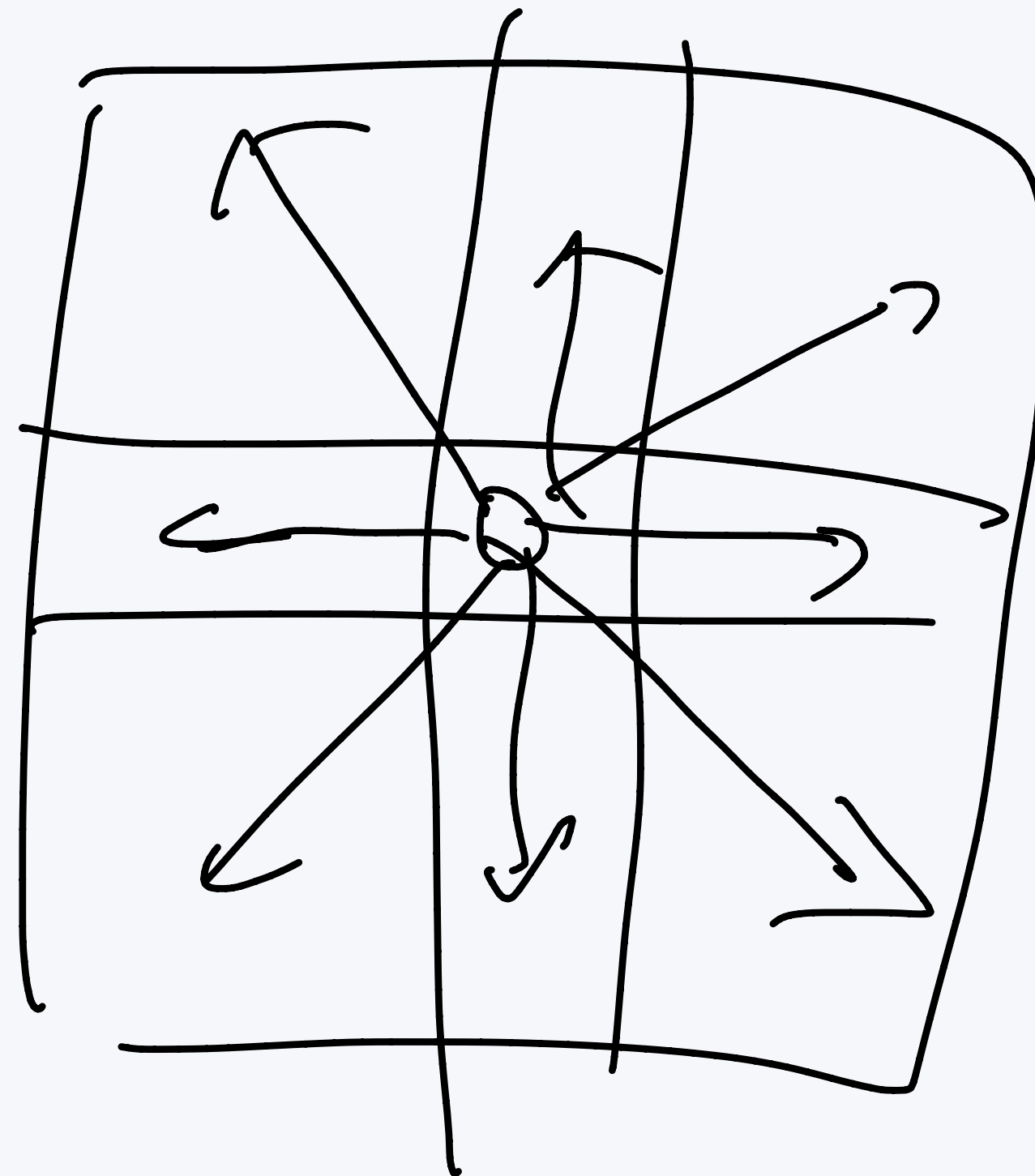
N-Queen

<https://www.acmicpc.net/problem/9663>

크윈

100

- $N \times N$ 크기의 체스판 위에 Queen을 N 개 놓는 방법의 수를 구하는 문제



1행 : 1개

1열 : 1개

↖ : 1개

↘ : 1개

N-Queen

101

<https://www.acmicpc.net/problem/9663>

- `calc(row)`: row 행에 퀸을 어디에 놓을지 결정해야 함

N-Queen

<https://www.acmicpc.net/problem/9663>

- calc(row): row 행에 퀸을 어디에 놓을지 결정해야 함

```
void calc(int row) {
    if (row == n) {
        ans += 1;
    }
    for (int col=0; col<n; col++) {
        a[row][col] = true;
        if (check(row, col)) {
            calc(row+1);
        }
        a[row][col] = false;
    }
}
```

$$n^2 \rightarrow n^n$$

(row, col)

3 1 2 4 0 2 5

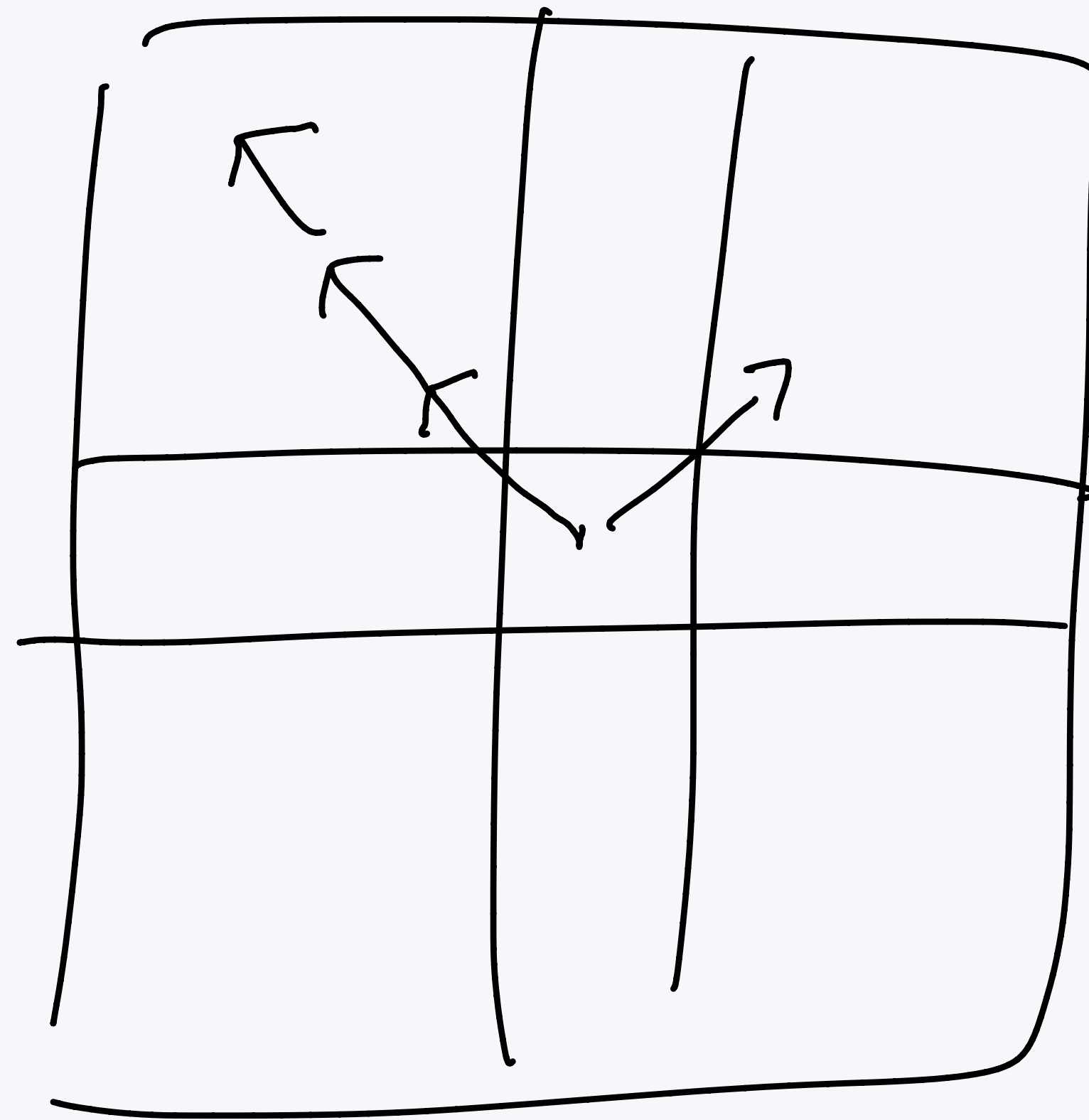
0 1 2 3

N-Queen

103

<https://www.acmicpc.net/problem/9663>

- C++: <https://gist.github.com/Baekjoon/1945a35cb532d5d294768d89822fbbfe>



N-Queen

<https://www.acmicpc.net/problem/9663>

- `check_col[i] = i번 열에 퀸이 놓여져 있으면 true`

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

N-Queen

105

<https://www.acmicpc.net/problem/9663>

row + col

- `check_dig[i] = /` 대각선에 퀸이 있으면

	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	6
2	2	3	4	5	6	7
3	3	4	5	6	7	8
4	4	5	6	7	8	9
5	5	6	7	8	9	10

N-Queen

106

<https://www.acmicpc.net/problem/9663>

- `check_dig2[i] = \` 대각선에 퀸이 있으면

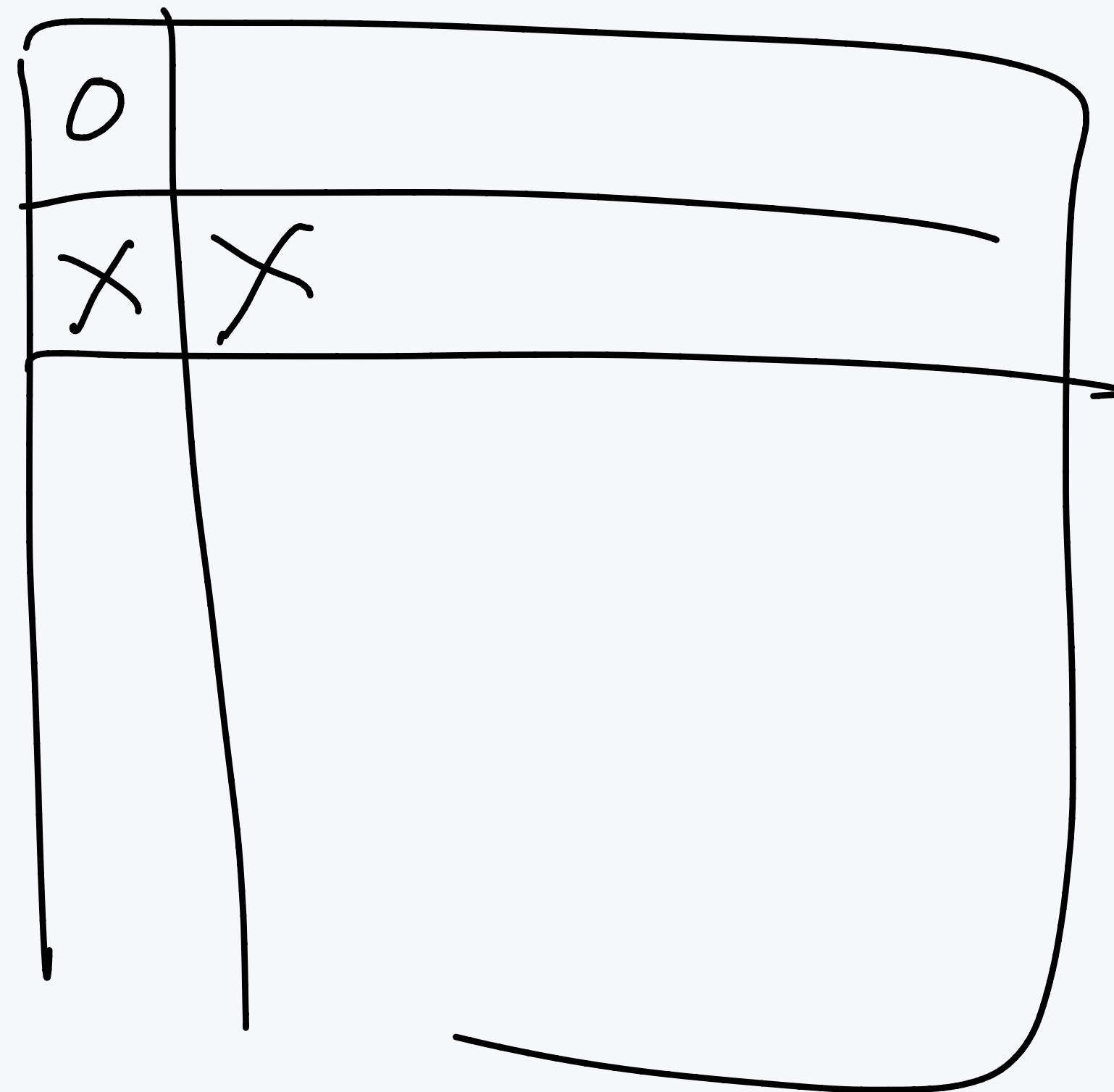
	0	1	2	3	4	5
0	5	4	3	2	1	0
1	6	5	4	3	2	1
2	7	6	5	4	3	2
3	8	7	6	5	4	3
4	9	8	7	6	5	4
5	10	9	8	7	6	5

$row - col + n - 1$

N-Queen

<https://www.acmicpc.net/problem/9663>

- Check 부분을 배열을 이용하면 $O(1)$ 만에 해결 할 수 있다.
- <https://gist.github.com/Baekjoon/7ce9963ec6d292ad9cfd2aeb3face717>



14^{14}

스도쿠

1 2 3 4 5 6 7 8 9

108

<https://www.acmicpc.net/problem/2580>

- 스도쿠를 푸는 문제

	3	5	4	6	9	2	7	8
7	8	2	1		5	6		9
	6		2	7	8	1	3	5
3	2	1		4	6	8	9	7
8		4	9	1	3	5		6
5	9	6	8	2		4	1	3
9	1	7	6	5	2		8	
6		3	7		1	9	5	2
2	5	8	3	9	4	7	6	

1	3	5	4	6	9	2	7	8
7	8	2	1	3	5	6	4	9
4	6	9	2	7	8	1	3	5
3	2	1	5	4	6	8	9	7
8	7	4	9	1	3	5	2	6
5	9	6	8	2	7	4	1	3
9	1	7	6	5	2	3	8	4
6	4	3	7	8	1	9	5	2
2	5	8	3	9	4	7	6	1

스도쿠

<https://www.acmicpc.net/problem/2580>

- $go(z)$: z 번째 칸을 채우는 함수
- $(x, y) \rightarrow 9 * x + y$ 번째 칸

0	1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26
27	28	29	30	31	32	33	34	35
36	37	38	39	40	41	42	43	44
45	46	47	48	49	50	51	52	53
54	55	56	57	58	59	60	61	62
63	64	65	66	67	68	69	70	71
72	73	74	75	76	77	78	79	80

- `c2[i][j]` = i열에 숫자 j가 있으면 true

[illegible]

스도쿠

<https://www.acmicpc.net/problem/2580>

- $c3[i][j] = i$ 번 작은 정사각형에 숫자 j 가 있으면 true
- (x, y) 는 $(x/3)*3+(y/3)$ 번째 칸

0	0	0	1	1	1	2	2	2
0	0	0	1	1	1	2	2	2
0	0	0	1	1	1	2	2	2
3	3	3	4	4	4	5	5	5
3	3	3	4	4	4	5	5	5
3	3	3	4	4	4	5	5	5
6	6	6	7	7	7	8	8	8
6	6	6	7	7	7	8	8	8
6	6	6	7	7	7	8	8	8

<https://www.acmicpc.net/problem/2580>

```
for (int i=0; i<n; i++) {
    for (int j=0; j<n; j++) {
        cin >> a[i][j];
        if (a[i][j] != 0) {
            c[i][a[i][j]] = true;
            c2[j][a[i][j]] = true;
            c3[square(i,j)][a[i][j]] = true;
        }
    }
}

go(0);
```

스도쿠

114

<https://www.acmicpc.net/problem/2580>

```
void go(int z) {
    if (z == 81) {
        // check
        exit(0);
    }
    int x = z/n, y = z%n;
    if (a[x][y] != 0) {
        go(z+1);
    } else {
        // next
    }
}
```

<https://www.acmicpc.net/problem/2580>

```
// check
for (int i=0; i<n; i++) {
    for (int j=0; j<n; j++) {
        cout << a[i][j] << ' ';
    }
    cout << '\n';
}
exit(0);
```

<https://www.acmicpc.net/problem/2580>

```
// next
```

```
for (int i=1; i<=9; i++) {
    if (c[x][i] == 0 && c2[y][i] == 0 && c3[square(x,y)][i] == 0) {
        c[x][i] = c2[y][i] = c3[square(x,y)][i] = true;
        a[x][y] = i;
        go(z+1);
        a[x][y] = 0;
        c[x][i] = c2[y][i] = c3[square(x,y)][i] = false;
    }
}
```

스도쿠

117

<https://www.acmicpc.net/problem/2580>

- C++: <https://gist.github.com/Baekjoon/fbf225181c5946773106>
- C++: (exit 함수 없음) <https://gist.github.com/Baekjoon/23b7f9e8cd454d94d2a7>
- Java: <https://gist.github.com/Baekjoon/687e776019684b5aca54>

Knuth Dancing Link X

알파벳

<https://www.acmicpc.net/problem/1987>

- 세로 R칸, 가로 C칸으로 된 표 모양의 보드가 있다
- 보드의 각 칸에는 대문자 알파벳이 하나씩 적혀 있고, 좌측 상단 칸 (1행 1열) 에는 말이 놓여 있다
- 말은 상하좌우로 인접한 네 칸 중의 한 칸으로 이동할 수 있다
- 같은 알파벳이 적힌 칸을 두 번 지날 수 없다
- 좌측 상단에서 시작해서, 말이 최대한 몇 칸을 지날 수 있는지를 구하는 문제

알파벳

<https://www.acmicpc.net/problem/1987>

- go(board, check, x, y, cnt)
 - board: 보드
 - check: 방문한 알파벳
 - x, y: 현재 위치
 - cnt: 방문한 칸의 수

알파벳

120

<https://www.acmicpc.net/problem/1987>

- `go(board, check, x, y, cnt)`
 - board: 보드
 - check: 방문한 알파벳
 - x, y: 현재 위치
 - cnt: 방문한 칸의 수
- 새로운 칸 `nx, ny`로 이동할 수 있는 경우
 - `go(board, check, nx, ny, cnt+1)`
 - 이 때, check는 변경해 줘야함

알파벳

<https://www.acmicpc.net/problem/1987>

```
void go(vector<string> &board, vector<bool> &check, int x, int y, int
cnt) {
    if (cnt > ans) ans = cnt;
    for (int k=0; k<4; k++) {
        int nx = x+dx[k];
        int ny = y+dy[k];
        if (nx >= 0 && nx < board.size() && ny >= 0 && ny <
board[0].size()) {
            if (check[board[nx][ny]-'A'] == false) {
                check[board[nx][ny]-'A'] = true;
                go(board, check, nx, ny, cnt+1);
                check[board[nx][ny]-'A'] = false;
            }
        }
    }
}
```

알파벳

122

<https://www.acmicpc.net/problem/1987>

- `go(board, check, x, y)`
- `board`: 보드
- `check`: 방문한 알파벳
- `x, y`: 현재 위치
- 리턴 값: 방문할 수 있는 칸의 최대 개수
- 의미: (x, y) 에서 이동을 시작하고, 방문한 알파벳이 `check`일 때, 방문할 수 있는 칸의 최대 개수

알파벳

123

<https://www.acmicpc.net/problem/1987>

```
int go(vector<string> &board, vector<bool> &check, int x, int y) {
    int ans = 0;
    for (int k=0; k<4; k++) {
        int nx = x+dx[k], ny = y+dy[k];
        if (nx >= 0 && nx < board.size() && ny >= 0 && ny <
board[0].size()) {
            if (check[board[nx][ny]-'A'] == false) {
                check[board[nx][ny]-'A'] = true;
                int next = go(board, check, nx, ny);
                if (ans < next) ans = next;
                check[board[nx][ny]-'A'] = false;
            }
        }
    }
    return ans + 1;
}
```

알파벳

124

<https://www.acmicpc.net/problem/1987>

- C++: <https://gist.github.com/Baekjoon/f412bcc16f3b3f0cbffd>
- Java: <https://gist.github.com/Baekjoon/411767759d38830b5911>

로또

125

<https://www.acmicpc.net/problem/6603>

- 로또의 모든 조합을 출력해보는 문제
- 6중 for문을 사용해도 된다

로또

126

<https://www.acmicpc.net/problem/6603>

- 6중 for문을 사용해도 된다
- C++: <https://gist.github.com/Baekjoon/0c0bd0aa6b91b7018550>

부분집합의 합

127

<https://www.acmicpc.net/problem/1182>

- 서로 다른 N개의 정수로 이루어진 집합이 있을 때, 이 집합의 공집합이 아닌 부분집합 중에서 그 집합의 원소를 다 더한 값이 S가 되는 경우의 수를 구하는 문제
- $1 \leq N \leq 20$

부분집합의 합

128

<https://www.acmicpc.net/problem/1182>

- 서로 다른 N개의 정수로 이루어진 집합이 있을 때, 이 집합의 공집합이 아닌 부분집합 중에서 그 집합의 원소를 다 더한 값이 S가 되는 경우의 수를 구하는 문제
- $1 \leq N \leq 20$

부분집합의 합

129

<https://www.acmicpc.net/problem/1182>

- C++: <https://gist.github.com/Baekjoon/5d90f9d1582559c619ad2821b126ac16>
- Java: <https://gist.github.com/Baekjoon/923eddd3d8d3bef43372433c83afb6cf>

비트마스크 사용하기

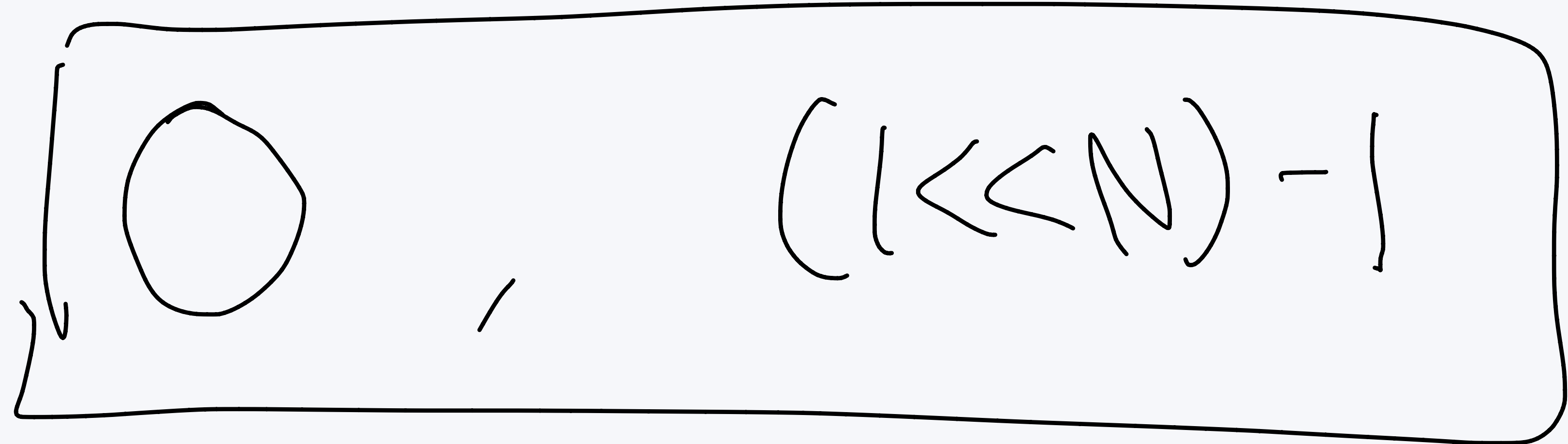
부분집합의 합

<https://www.acmicpc.net/problem/1182>

131



- 서로 다른 N 개의 정수로 이루어진 집합이 있을 때, 이 집합의 공집합이 아닌 부분집합 중에서 그 집합의 원소를 다 더한 값이 S 가 되는 경우의 수를 구하는 문제
- $1 \leq N \leq 20$



부분집합의 합

132

<https://www.acmicpc.net/problem/1182>

- 모든 집합의 개수 = 2^N
- 모든 집합을 구해보면 된다!

부분집합의 합

133

<https://www.acmicpc.net/problem/1182>

- 전체 집합 = $(1 \ll N) - 1$

```
for (int i=0; i<(1<<n); i++) {  
  
}
```

부분집합의 합

<https://www.acmicpc.net/problem/1182>

- 전체 집합 = $(1 \ll N) - 1$
- 공집합은 제외해야 한다

```
for (int i=1; i<(1<<n); i++) {  
  
}
```

부분집합의 합

<https://www.acmicpc.net/problem/1182>

- 전체 집합 = $(1 \ll N) - 1$
- 공집합은 제외해야 한다
- 집합에 무엇이 포함되어 있는지 확인하기

```
for (int i=1; i<(1<<n); i++) {  
    for (int k=0; k<n; k++) {  
        if (i&(1<<k)) {  
            }  
        }  
    }  
}
```

부분집합의 합

<https://www.acmicpc.net/problem/1182>

for (int i=1; i<(1<<n); i++) { —————> 모든 부분집합

int sum = 0;

```
for (int k=0; k<n; k++) {
    if (i&(1<<k)) {
        sum += a[k];
    }
}
```

합 구하기

```
if (sum == s) {
    ans += 1;
}
```

결과 구하기

}

부분집합의 합

137

<https://www.acmicpc.net/problem/1182>

- C++: <https://gist.github.com/Baekjoon/f4154089addcd1adacc5>
- Java: <https://gist.github.com/Baekjoon/bddda372acf45d698817>