DFS / BFS: 한 정점에서 시작해서 모든 정점을 (연속 방문

BFS

최백준 choi@startlink.io

BFS

BFS

- BFS의 목적은 임의의 정점에서 시작해서, 모든 정점을 한 번씩 방문하는 것이다.

BFS

BFS

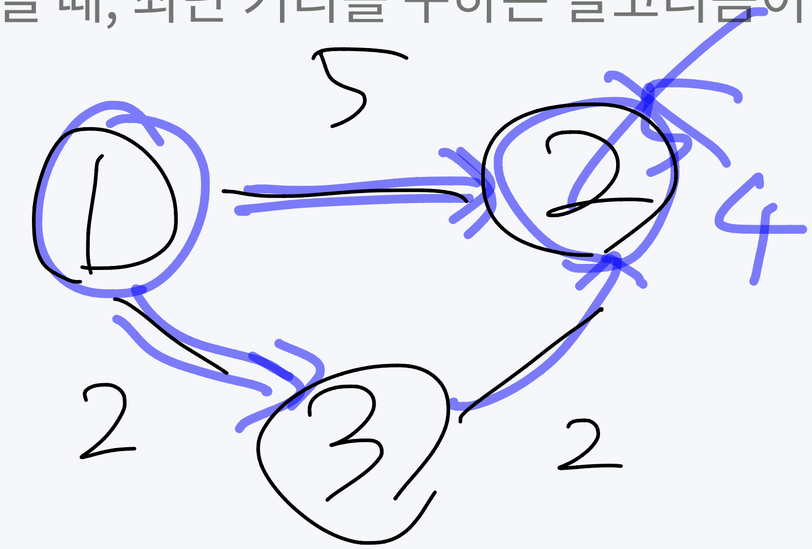
- BFS는 최단 거리를 구하는 알고리즘이다.

BFS

BFS

최소 비용

- BFS는 모든 가중치가 1일 때, 최단 거리를 구하는 알고리즘이다.



BFS

BFS

- BFS를 이용해 해결할 수 있는 문제는 아래와 같은 조건을 만족해야 한다

1. 최소 비용 문제이어야 한다
2. 간선의 가중치가 1이어야 한다
3. 정점과 간선의 개수가 적어야 한다. (적다는 것은 문제의 조건에 맞춰서 해결할 수 있다는 것을 의미한다)

BFS

BFS

- BFS를 이용해 해결할 수 있는 문제는 아래와 같은 조건을 만족해야 한다

1. **최소 비용** 문제이어야 한다
2. **간선의 가중치**가 1이어야 한다
3. 정점과 간선의 개수가 적어야 한다. (적다는 것은 문제의 조건에 맞춰서 해결할 수 있다는 것을 의미한다)

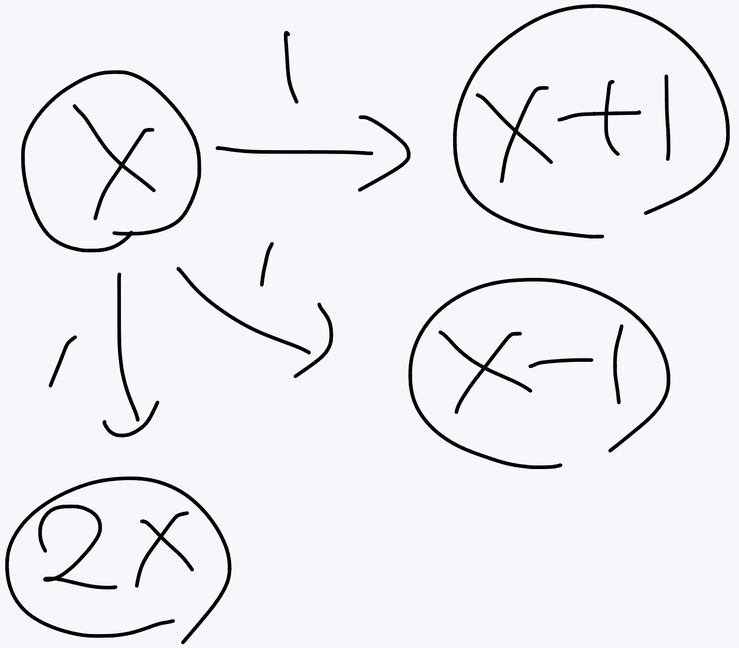
- 간선의 가중치가 문제에서 구하라고 하는 최소 비용과 의미가 일치해야 한다
- 즉, 거리의 최소값을 구하는 문제라면 가중치는 거리를 의미해야 하고, 시간의 최소값을 구하는 문제라면 가중치는 시간을 의미해야 한다

BFS

숨바꼭질

0 ≤ N ≤ 10만

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-1로 이동 (1초)
2. 순간이동: 2*X로 이동 (1초)

숨바꼭질

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

- 수빈이의 위치: N
- 동생의 위치: K
- 동생을 찾는 **가장 빠른 시간**을 구하는 문제

- 수빈이가 할 수 있는 행동 (위치: X)

1. 걷기: X+1 또는 X-1로 이동 (**1초**)
2. 순간이동: 2*X로 이동 (**1초**)

숨바꼭질

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

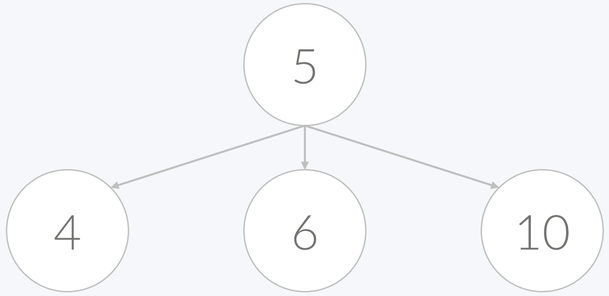
5

- 가장 처음
- Queue: 5

숨바꼭질

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

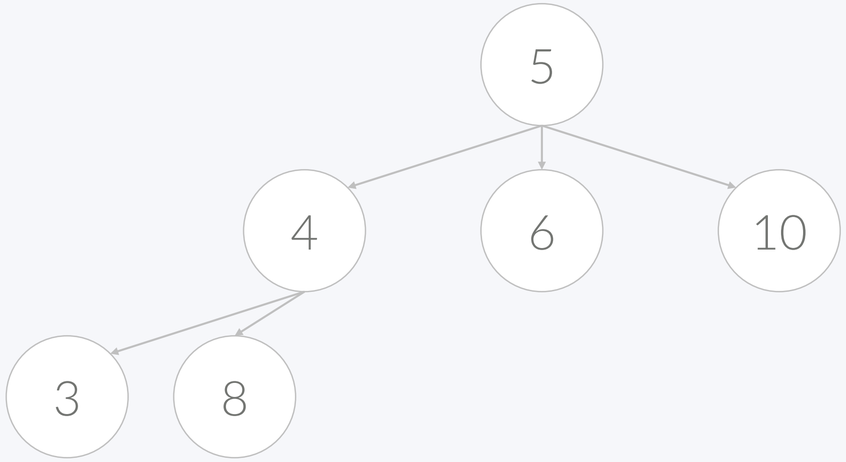
- 5에서 이동
- Queue: 5 4 6 10



숨바꼭질

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

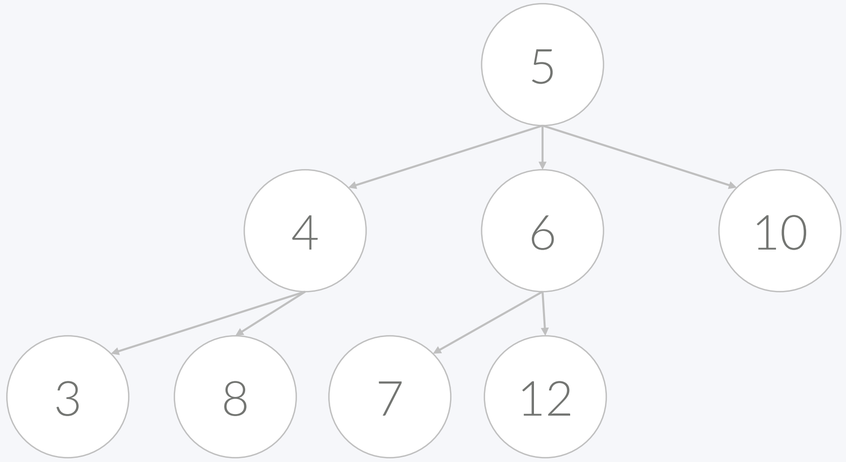
- 4에서 이동
- Queue: 5 4 6 10 3 8



숨바꼭질

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

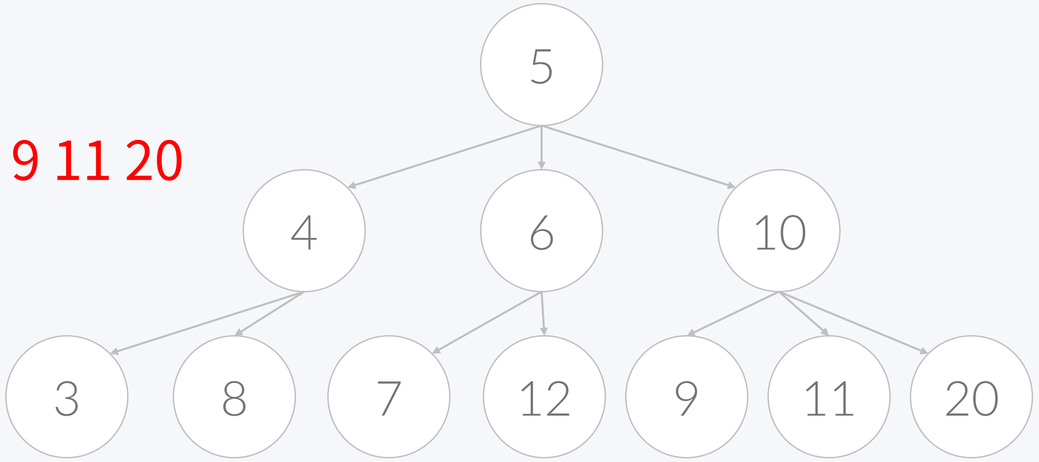
- 6에서 이동
- Queue: 5 4 6 10 3 8 7 12



숨바꼭질

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

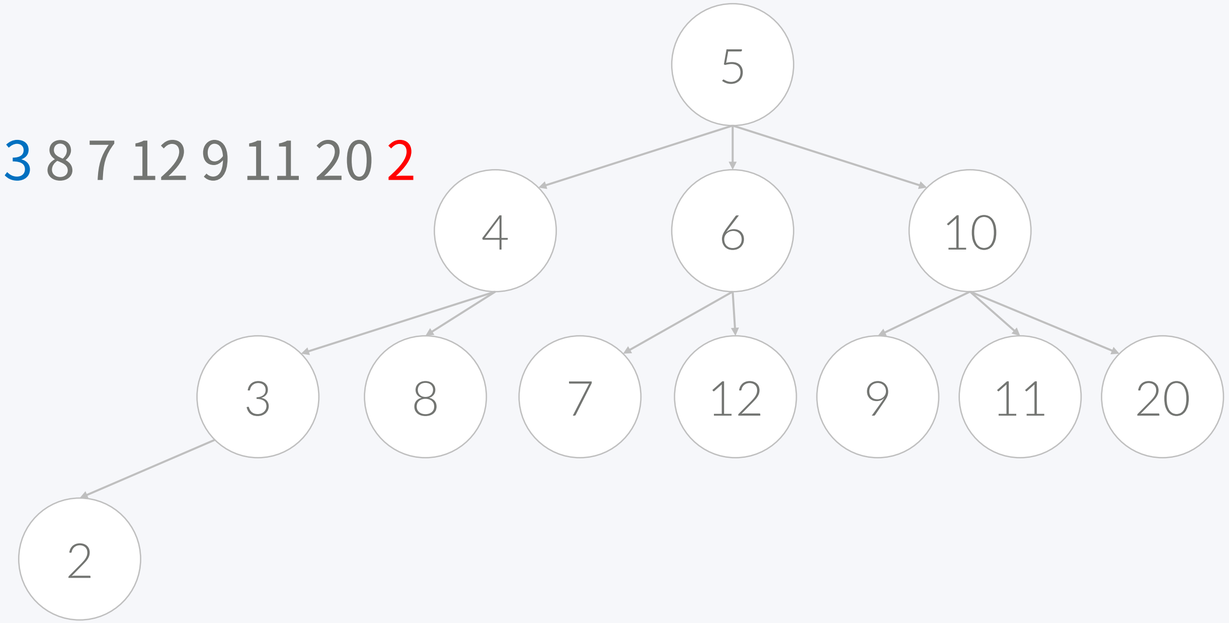
- 10에서 이동
- Queue: 5 4 6 10 3 8 7 12 9 11 20



숨바꼭질

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

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



숨바꼭질

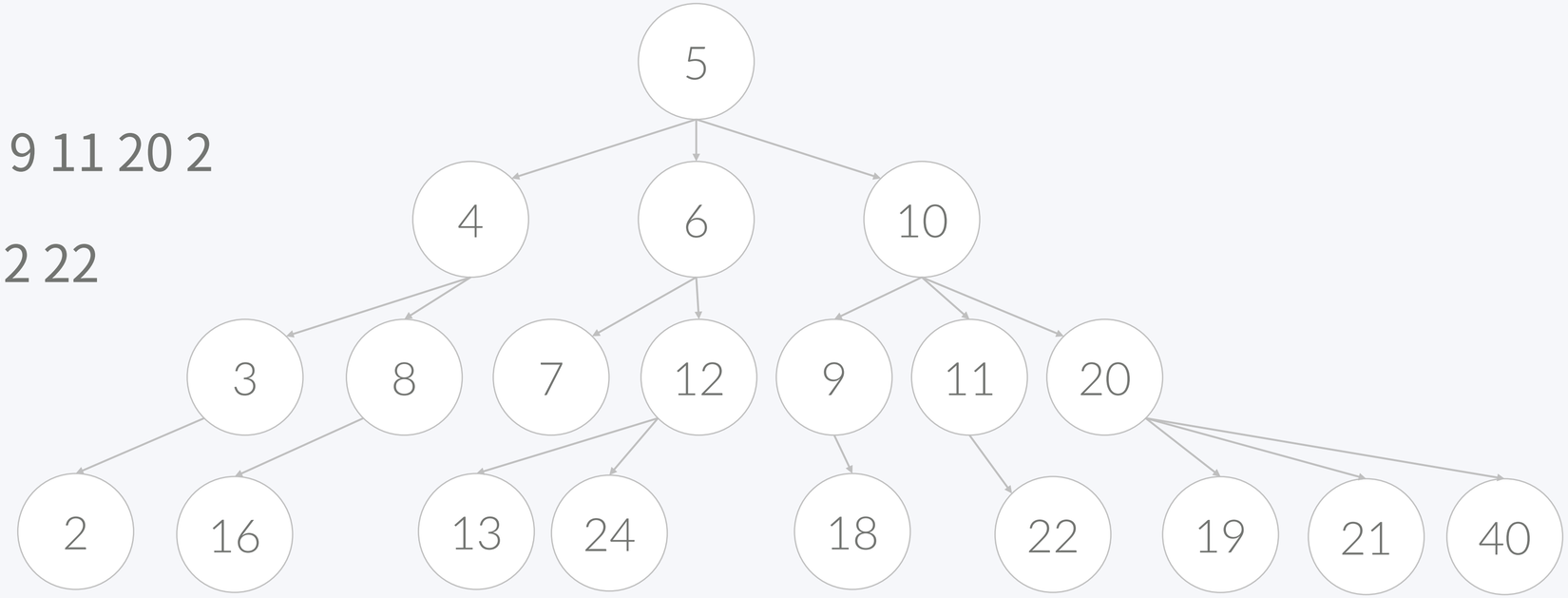
https://www.acmicpc.net/problem/1697

- 이런식으로…

숨바꼭질

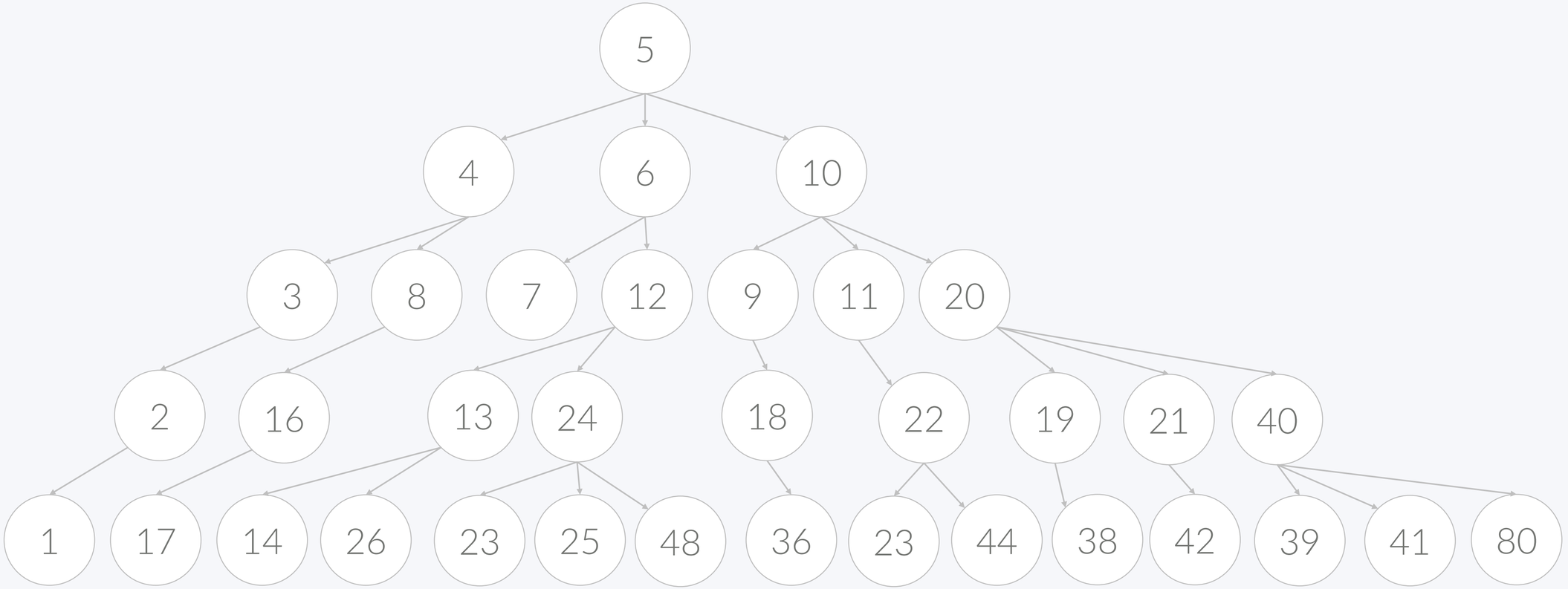
https://www.acmicpc.net/problem/1697

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



숨바꼭질

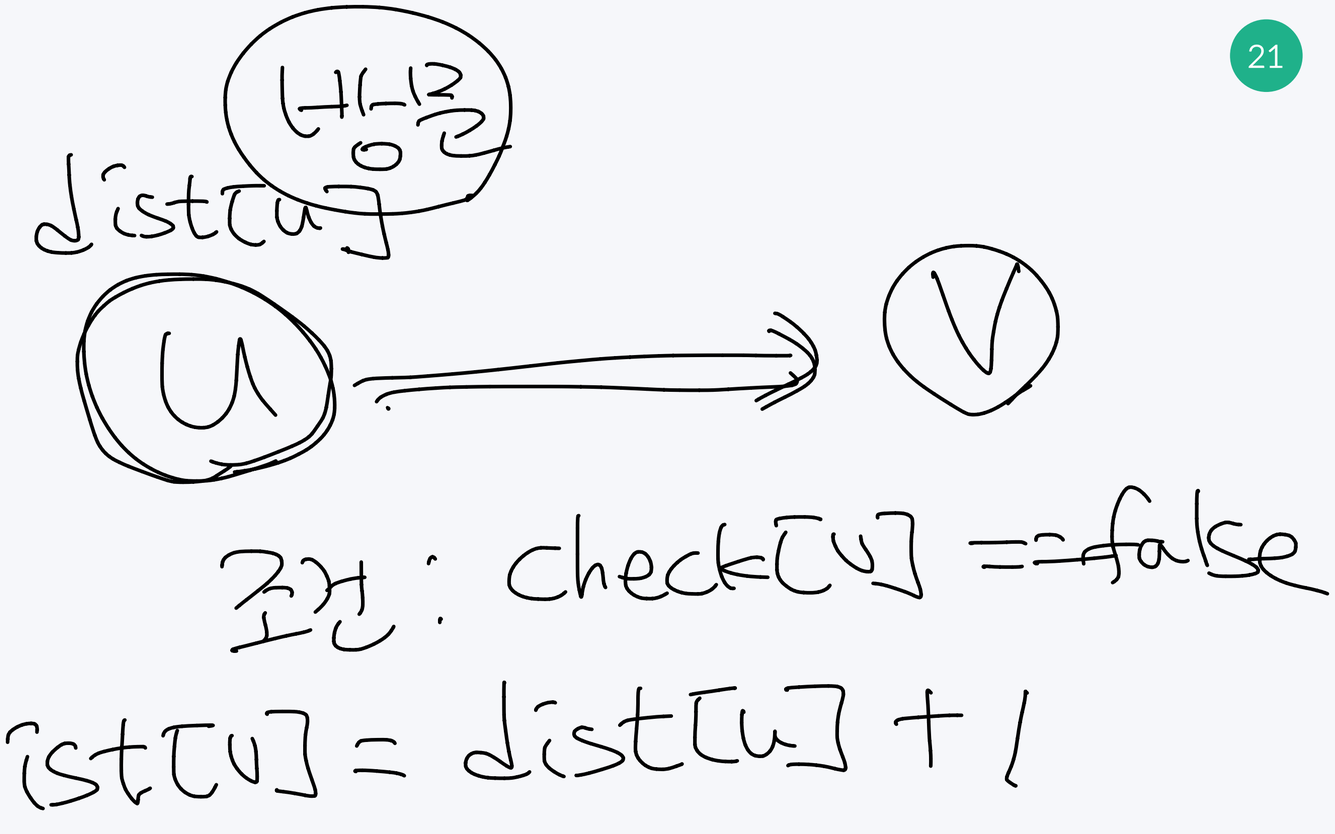
https://www.acmicpc.net/problem/1697



숨바꼭질

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

- check[i] = i를 방문했는지
- dist[i] = i를 몇 번만에 방문했는지



숨바꼭질

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

```
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

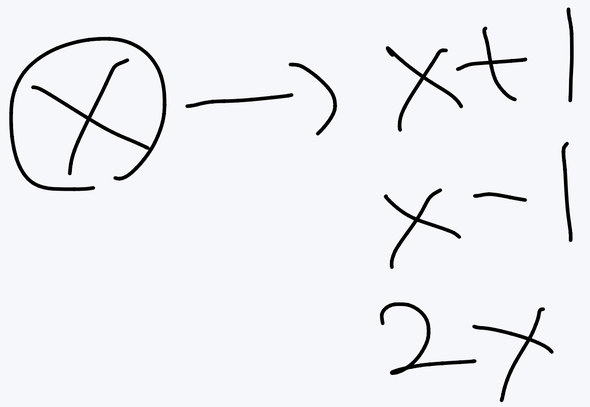
https://www.acmicpc.net/problem/1697

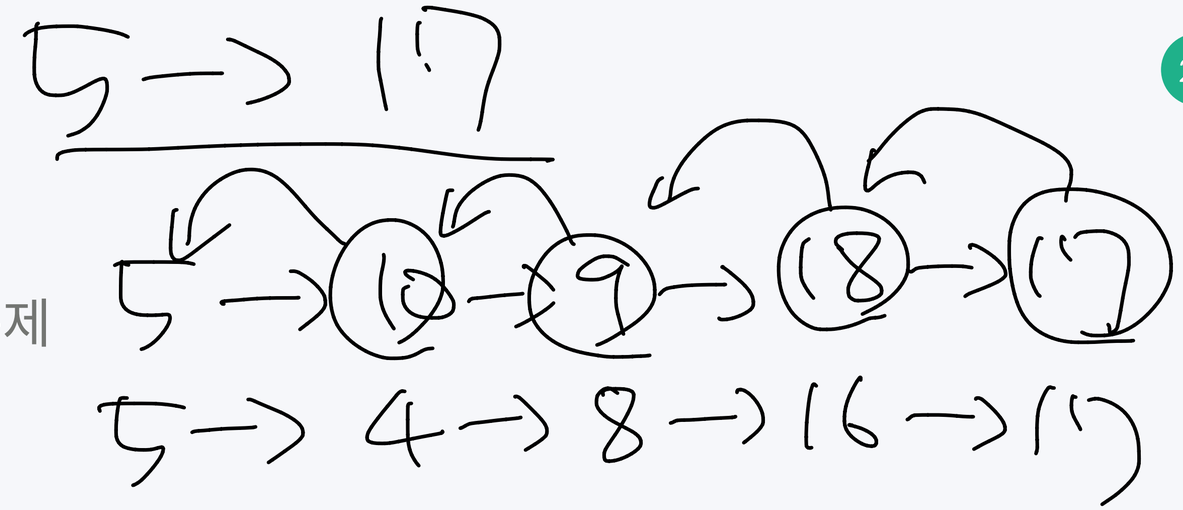
- 소스: http://boj.kr/df84a3678d4a41ec847569ee0d465fb7

숨바꼭질 4

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

- 숨바꼭질 문제 + 이동하는 방법을 출력하는 문제





5 -> 4 -> 8 -> 16 -> 17

숨바꼭질 4

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

- now -> next를 갔다고 한다면

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

from[i] = i는 어떻게?

from[i] => i

now -> next

from[next] = now;

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

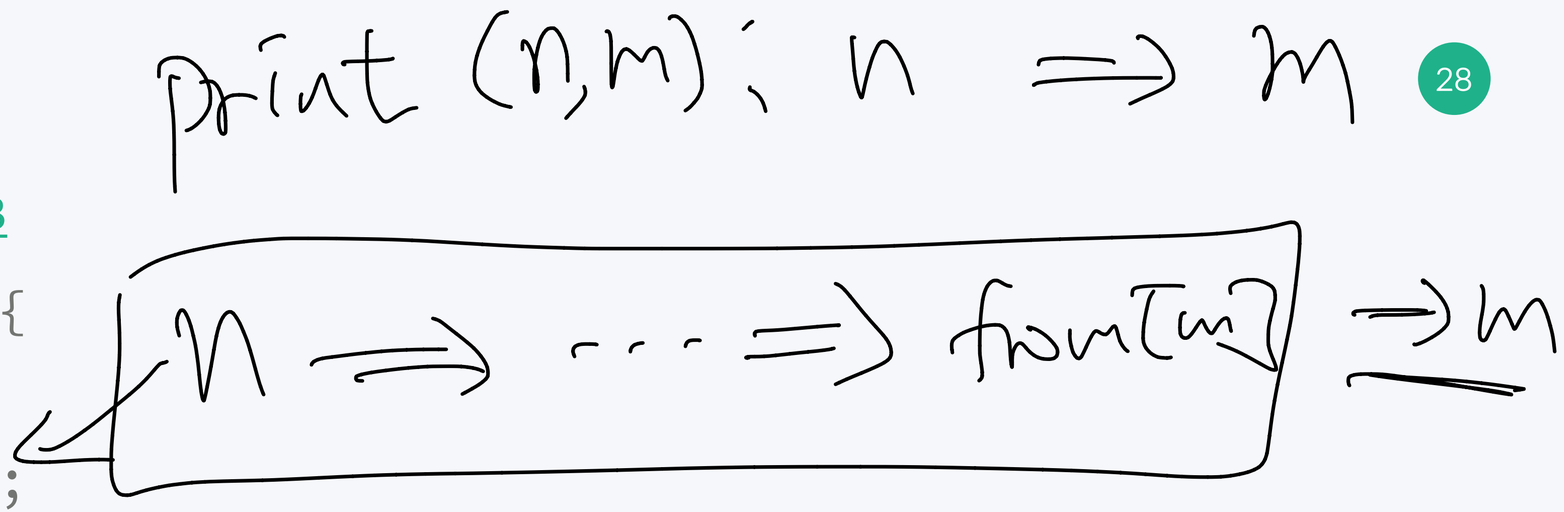
https://www.acmicpc.net/problem/13913

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

숨바꼭질 4

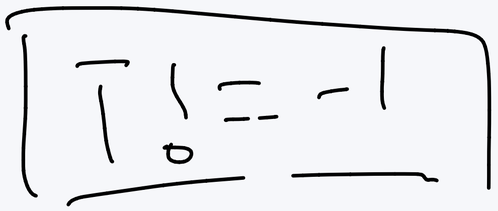
https://www.acmicpc.net/problem/13913

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

숨바꼭질 4

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

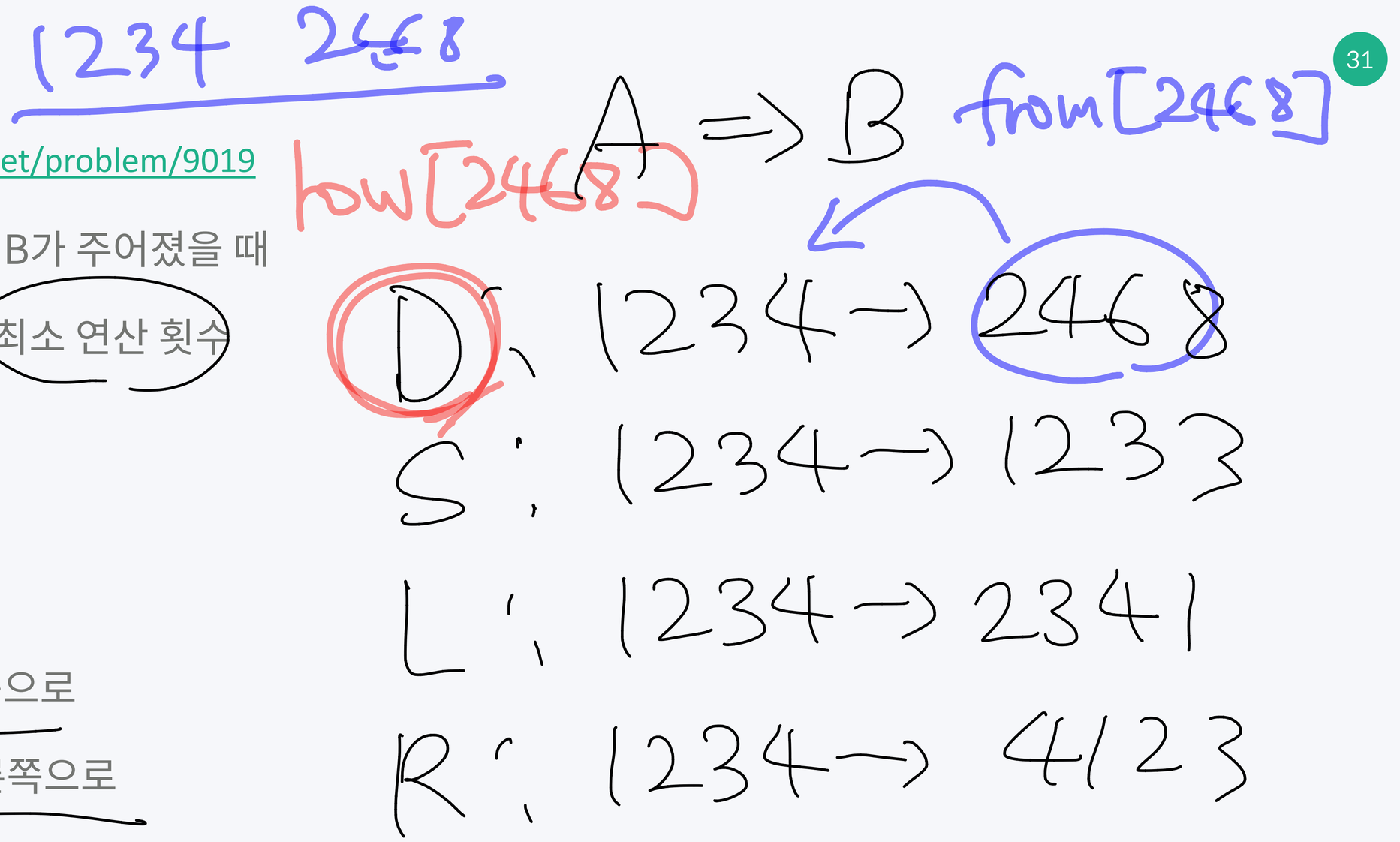
- 소스: http://boj.kr/5595c82d14eb49628e863ec6b928cc7f

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



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

DSLR

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] = ?
```

DSLR

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] = ?
```

DSLR

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
```

DSLR

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
```

DSLR

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
```

DSLR

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
```

DSLR

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
```

DSLR

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
```

DSLR

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
```

DSLR

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
```

DSLR

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
```

DSLR

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

- 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

```
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';
}
```

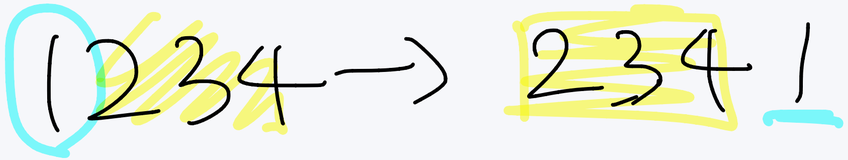
DSLR

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';
}
```

DSLR

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';
}
```

DSLR

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';
}
```

$N \le 10000$

인접행렬 N^2

4×10000

4000만 바이트

= 39MB

DSLR

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';
```

$N \le 10,000$

how[i] = 방법

char how[10000];

String

10000^2

DSLR

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

DSLR

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] = 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
```

DSLR

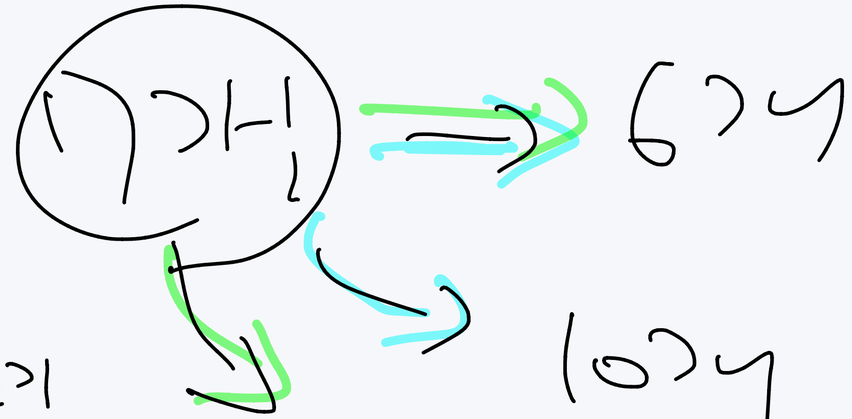
https://www.acmicpc.net/problem/9019

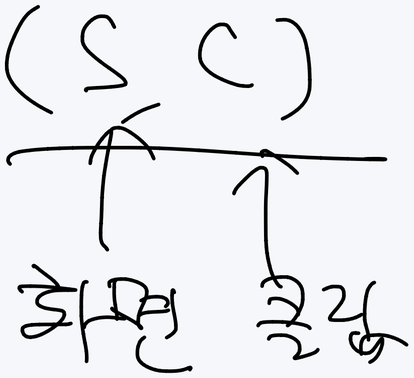
- 소스: http://boj.kr/447474df0fc544cb9ca277a7478cfc5b

이모티콘

https://www.acmicpc.net/problem/14226

- 화면에 이모티콘은 1개다
- 할 수 있는 연산
  1. 이모티콘을 모두 복사해서 클립보드에 ~~저장~~ 덮어쓰기
  2. 클립보드에 있는 모든 이모티콘을 화면에 붙여넣기
  3. 화면에 있는 이모티콘 중 하나를 삭제
- S개의 이모티콘을 만드는데 걸리는 시간의 최소값을 구하는 문제





이모티콘

https://www.acmicpc.net/problem/14226

- BFS에서 하나의 정점이 서로 다른 두 개의 정보를 저장하고 있으면 안된다
- 화면에 있는 이모티콘의 개수가 5개인 경우
- 클립보드에 있는 이모티콘의 개수에 따라서, 클립보드에서 복사하기 연산의 결과가 다르다
- 즉, 화면에 이모티콘의 개수 s와 클립보드에 있는 이모티콘의 개수 c가 중요하다

이모티콘

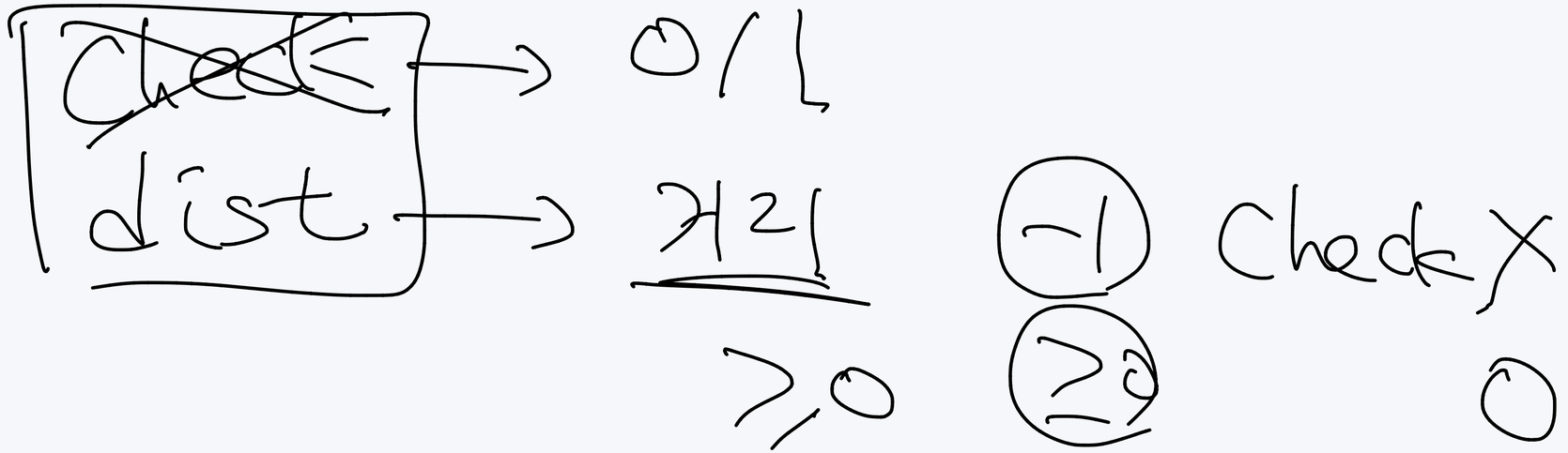
https://www.acmicpc.net/problem/14226

- 복사: (s, c) -> (s, s)
- 붙여넣기: (s, c) -> (s+c, c)
- 삭제: (s, c) -> (s-1, c)
- 2 ≤ S ≤ 1,000 이기 때문에 BFS 탐색으로 가능하다.

이모티콘

https://www.acmicpc.net/problem/14226

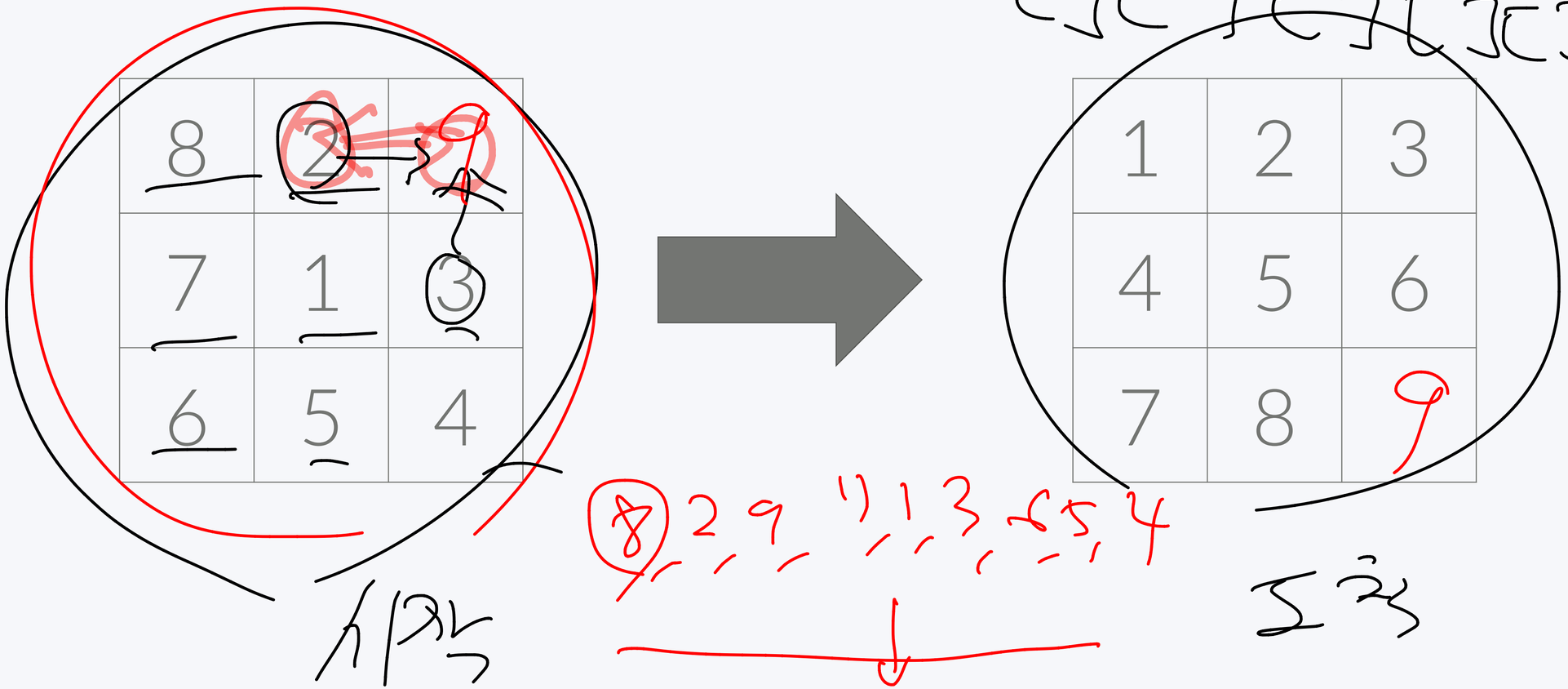
- 소스: http://boj.kr/25824a4d8cf84f1b972f1f1955b958d9



퍼즐

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

- 8퍼즐을 푸는 문제



퍼즐

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

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

퍼즐

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

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

퍼즐

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

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://boj.kr/7672349011e148cea3885324cf17526c

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 물이 양을 알 수 있다

물통

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
}
```

물통

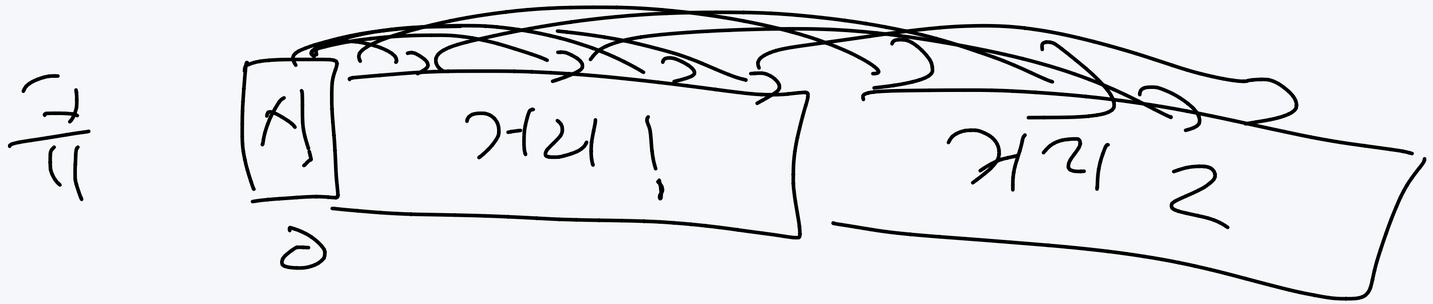
https://www.acmicpc.net/problem/2251

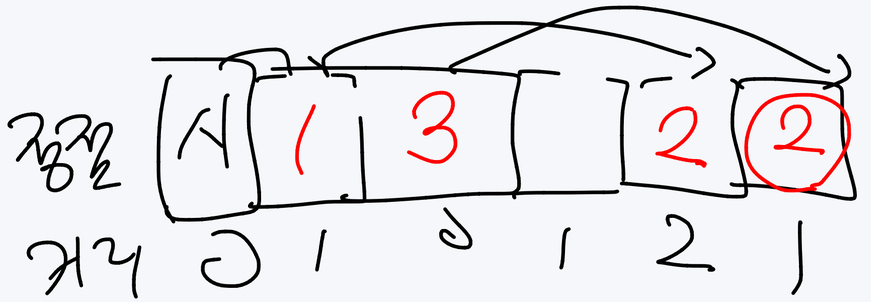
```
// x -> y
ny += nx; nx = 0;
if (ny >= b) {
    nx = ny-b;
    ny = b;
}
if (!check[nx][ny]) {
    check[nx][ny] = true;
    q.push(make_pair(nx,ny));
    if (nx == 0) {
        ans[nz] = true;
    }
}
```

물통

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

- 소스: http://boj.kr/0909893f09494442a9e8f1e9cabb8ebd





덱 사용하기

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 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-1로 이동 (1초)
2. 순간이동: 2*X로 이동 (0초)

숨바꼭질 3

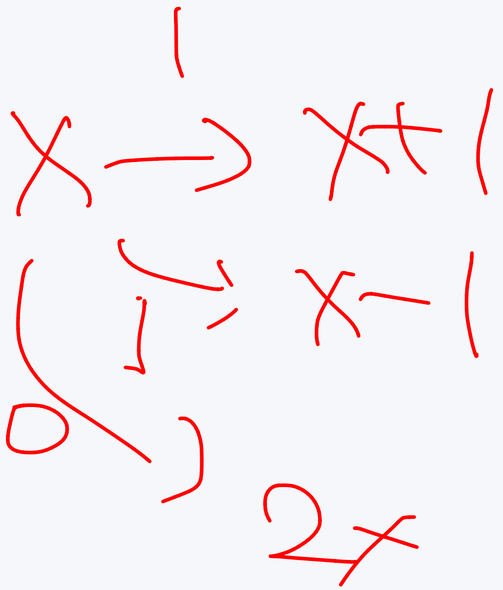
https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정



- 0초:

5	10				

- 1초:

6	4				

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10				

- 1초:

4	6				

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10				

- 1초:

4	6				

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11		

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11		

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정
- 0초:

5	10	20			

- 1초:

4	6	9	11	19	

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8

← 현재 큐

- 2초:

3					

← 다음 큐

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8

- 2초:

3					

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8						

- 2초:

3					

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8	12					

- 2초:

3	7				

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8	12					

- 2초:

3	7				

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8	12	18				

- 2초:

3	7	8			

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8	12	18				

- 2초:

3	7	8			

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8	12	18				

- 2초:

3	7	8			

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8	12	18				

- 2초:

3	7	8			

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 5에서 17을 가는 경우 0 ~ 20까지만 위치가 있다고 가정

- 0초:

5	10	20			

- 1초:

4	6	9	11	19	8	12	18	16			

- 2초:

3	7	8			

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 이런식으로 BFS를 진행한다.

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 덱을 사용해 순간 이동은 덱의 앞에, 걷기는 덱의 뒤에 넣는 방법도 생각해 볼 수 있다.

숨바꼭질 3

https://www.acmicpc.net/problem/13549

- 큐 소스: http://boj.kr/fcaac29becae4dfab3fdd05dcde9aa2e
- 덱 소스: http://boj.kr/1b8d34458c784634ab000622095ae027

알고스팟

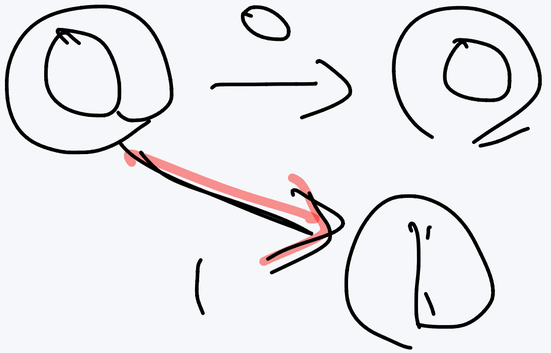
https://www.acmicpc.net/problem/1261

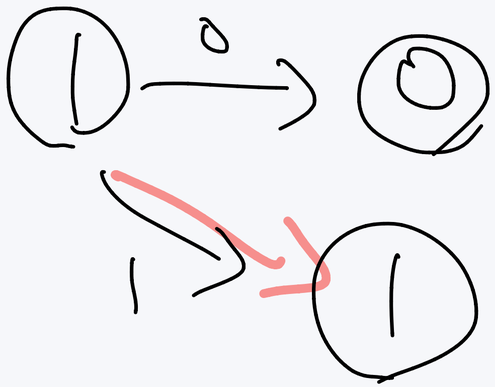
- 미로는 N*M크기이고, 총 1*1크기의 방으로 이루어져 있다
- 빈 방은 자유롭게 다닐 수 있지만, 벽은 부수지 않으면 이동할 수 없다
- (x, y)에 있을 때, 이동할 수 있는 방은 (x+1, y), (x-1, y), (x, y+1), (x, y-1) 이다
- (1, 1)에서 (N, M)으로 이동하려면 벽을 최소 몇 개 부수어야 하는지 구하는 문제

알고스팟

https://www.acmicpc.net/problem/1261

- 처음 상태





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

알고스팟

https://www.acmicpc.net/problem/1261

- 벽을 부수지 않고 이동할 수 있는 곳

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

0	0				
0					
0	0				

알고스팟

https://www.acmicpc.net/problem/1261

- 벽을 1개 부수고 이동할 수 있는 곳

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

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

알고스팟

https://www.acmicpc.net/problem/1261

- 벽을 2개 부수고 이동할 수 있는 곳

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

0	0	1	2	2	2
0	1	1	1	1	1
0	0	1	2	2	2
1	1	1	1	1	2
1	2	1	1	2	2
2	1	1	1	2	2

알고스팟

https://www.acmicpc.net/problem/1261

- BFS탐색을 벽을 부순 횟수에 따라서 나누어서 수행해야 한다.
- 소스: http://boj.kr/5b883c9670014207819f0034bd7b787d

알고스팟

https://www.acmicpc.net/problem/1261

- 어차피 벽을 뚫는다와 안 뚫는다로 나누어지기 때문에, 덱을 사용한다
- 벽을 뚫는 경우에는 뒤에, 안 뚫는 경우에는 앞에 추가한다.
- 소스: http://boj.kr/746846057a894fe692e76d32d259ab9f

BFS

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-1로 이동 (1초)
2. 순간이동: 2*X로 이동 (1초)

숨바꼭질 2

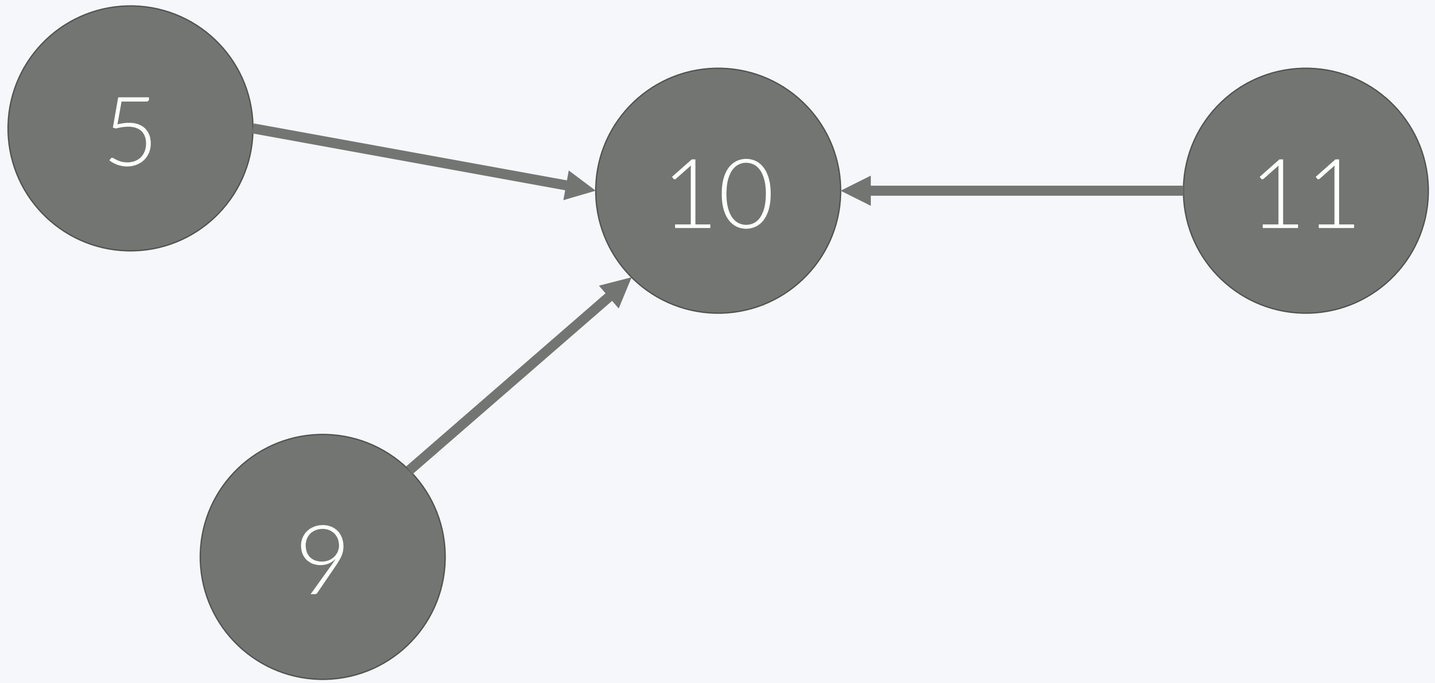
https://www.acmicpc.net/problem/12851

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

숨바꼭질 2

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

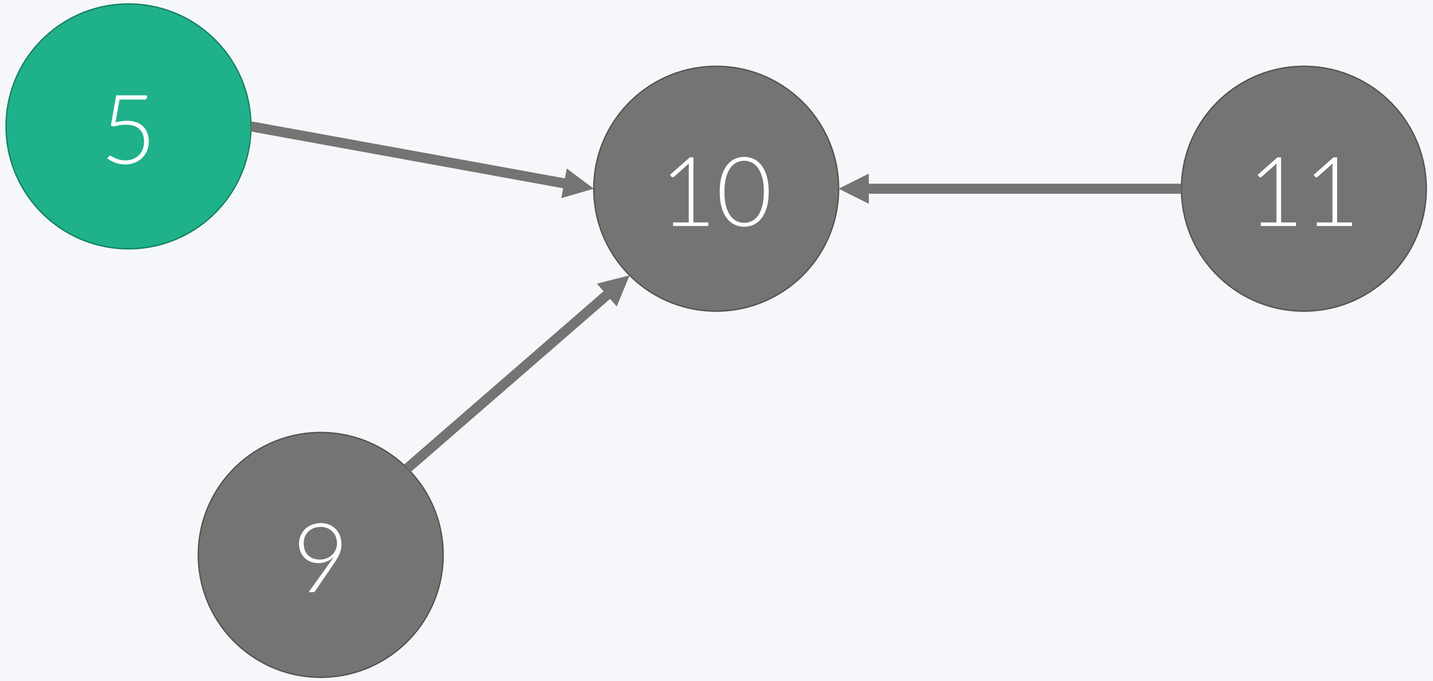
- 10을 아직 방문하지 않았고
- 시작점에서 5와 9까지 가는 거리는 3, 11은 아직 방문하지 않았다고 가정하자



숨바꼭질 2

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

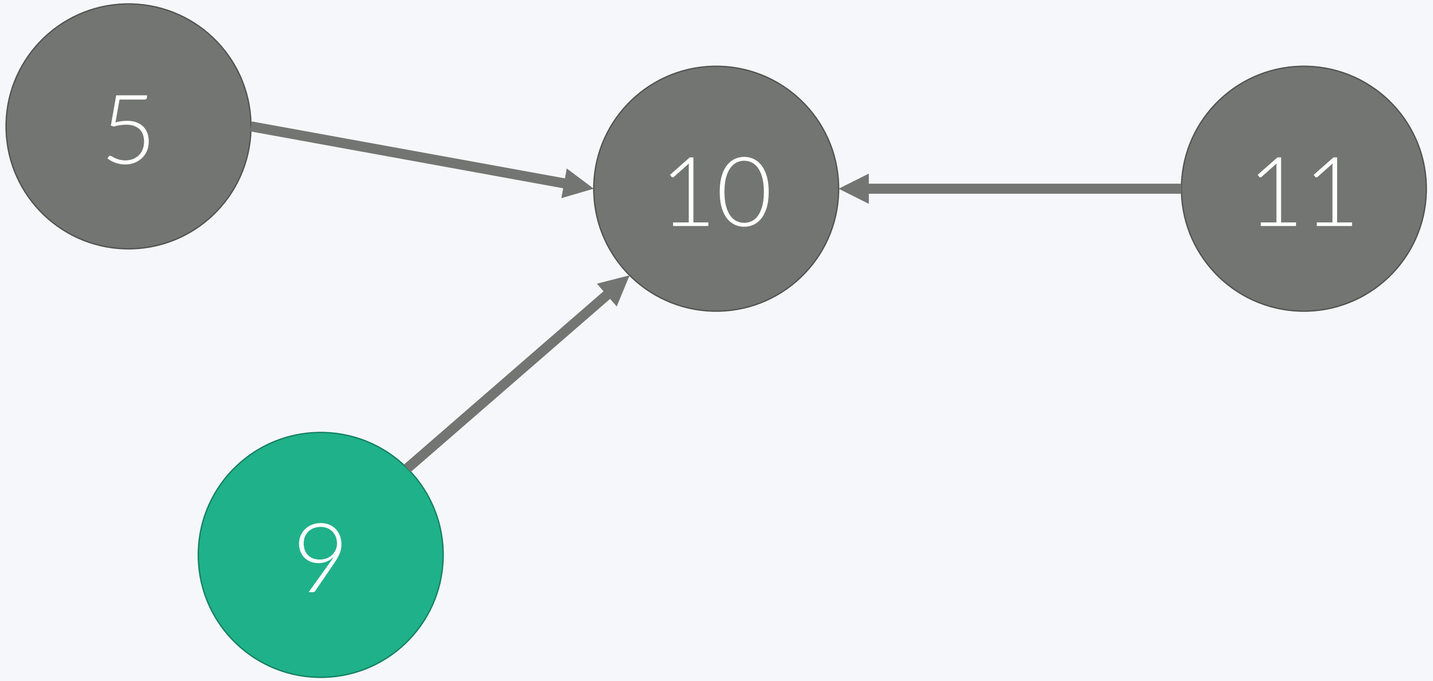
- 10은 아직 방문하지 않았기 때문에
- 10을 방문해야 한다.
- 이 때, cnt[10] = cnt[5]



숨바꼭질 2

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

- 10은 이미 방문했기 때문에
- 10을 방문할 수는 없다. 하지만, 방법의 수는 증가해야 하기 때문에
- cnt[10] += 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

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

- 소스: http://boj.kr/c8366f65436c4fee909ec0963118ce92

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

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 경우 이기 때문에
- 상태 (i, j) 대신에 (i, j, k) (k == 0이면 벽을 부순 적이 없음, 1이면 있음) 으로 BFS 탐색을 진행한다.

벽 부수고 이동하기

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

- 소스: http://boj.kr/7df51300632c4f479aca80098d6a02b7

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

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

.	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

2			
1			
0			
1			

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

3			
2			
1			
0			
1			

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

3	4		
2			
1			
0			
1			

탈출

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

- 소스: http://boj.kr/9405c21d4c0e41acbec8d344d5adf373

탈옥

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

- 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

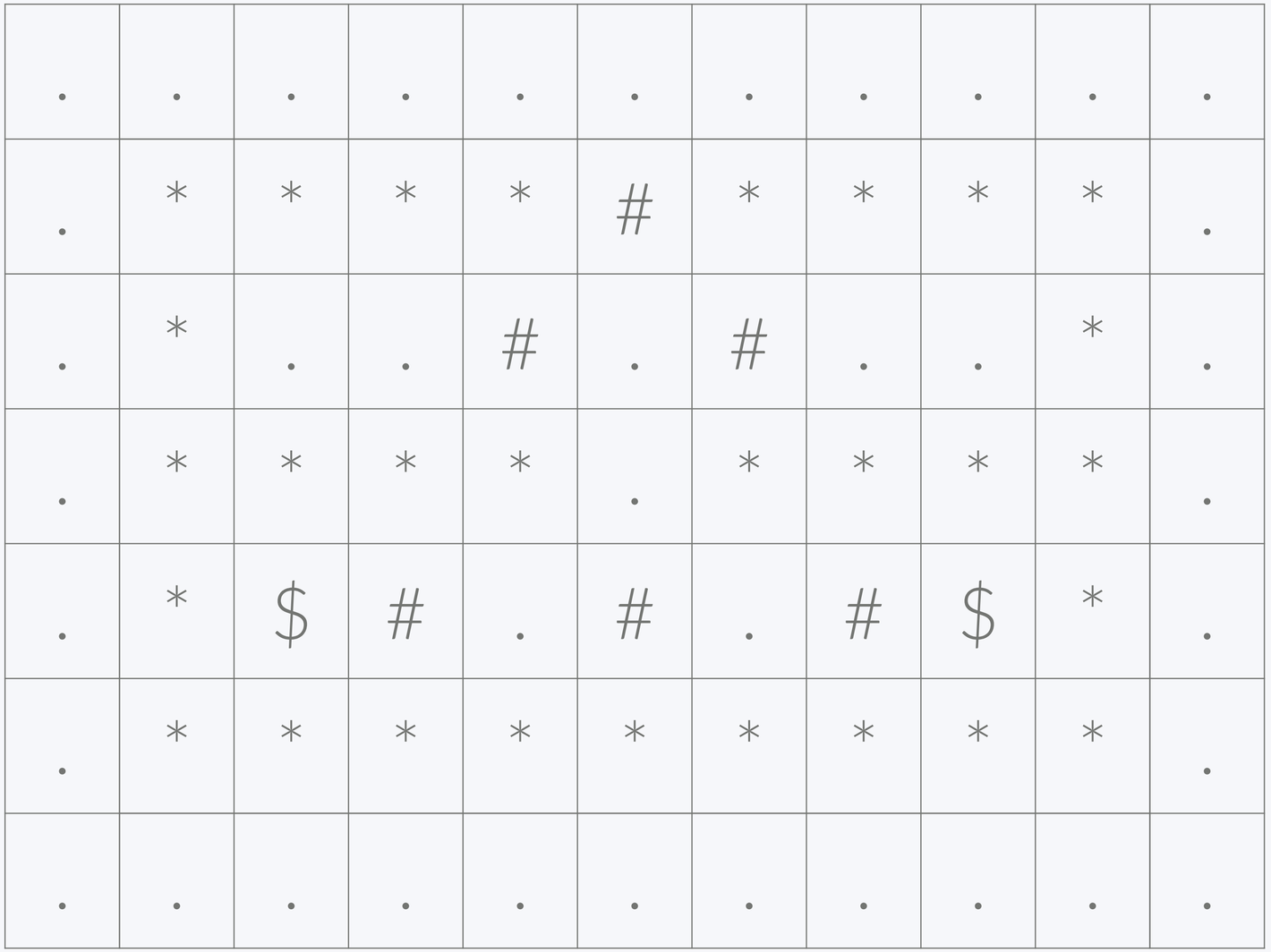
탈옥

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

*	*	*	*	#	*	*	*	*
*	.	.	#	.	#	.	.	*
*	*	*	*	.	*	*	*	*
*	$	#	.	#	.	#	$	*
*	*	*	*	*	*	*	*	*

탈옥

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



탈옥

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

탈옥

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

탈옥

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

- 소스: http://boj.kr/d2567ed218424eabaadc6819b8faa6db

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

열쇠

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

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

- 큐: (1, 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

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

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

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

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

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

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

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

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://boj.kr/bc8b2dc5386d4cc5b2ff82ee369d4e5f