

Chapter. 03

알고리즘

모의 코딩테스트 풀이 Mock Coding Test

FAST CAMPUS
ONLINE

알고리즘 공채 대비반 I

강사. 류호석

Chapter. 03



모의 코딩테스트 풀이 - 1

Mock Coding Test Solution - 2

I 주의사항

실제 시험이라고 생각하고 5시간을 고정시켜 놓은 상태로

<https://www.acmicpc.net/category/detail/2338>

위 대회를 직접 경험해보신 다음에 이후 풀이 영상을 보세요.

어떤 부분이 어려운 것인지, 생각의 방향이 어떻게 다른 지를 직접 느껴보셔야 합니다.


I BOJ 21275 – 폰 호석만

난이도: 2

$$0 \leq X < 2^{63}$$

$$2 \leq A, B \leq 36$$

Given


$$\mathbf{P}_A = \mathbf{Q}_B = X_{10}$$

$$\text{Ex) } \mathbf{ep}_A = \mathbf{jh}_B \rightarrow A = 32, B = 24, X = 473$$

$$\mathbf{ep}_{32} = 14 * 32 + 25 = 473 = X$$

$$\mathbf{jh}_{24} = 19 * 24 + 17 = 473 = X$$

I 출제 의도

- 완전 탐색 접근을 통해서 모든 경우를 직접 하나하나 찾아내 보자.
본 문제에서 “경우”란, 조건을 만족하게 A, B를 모두 결정해보는 것이다.
- 진법 변환을 구현할 수 있는 가?

I 완전 탐색 접근

가능한 A, B의 조합: (2,2), (2,3), ..., (36, 36)

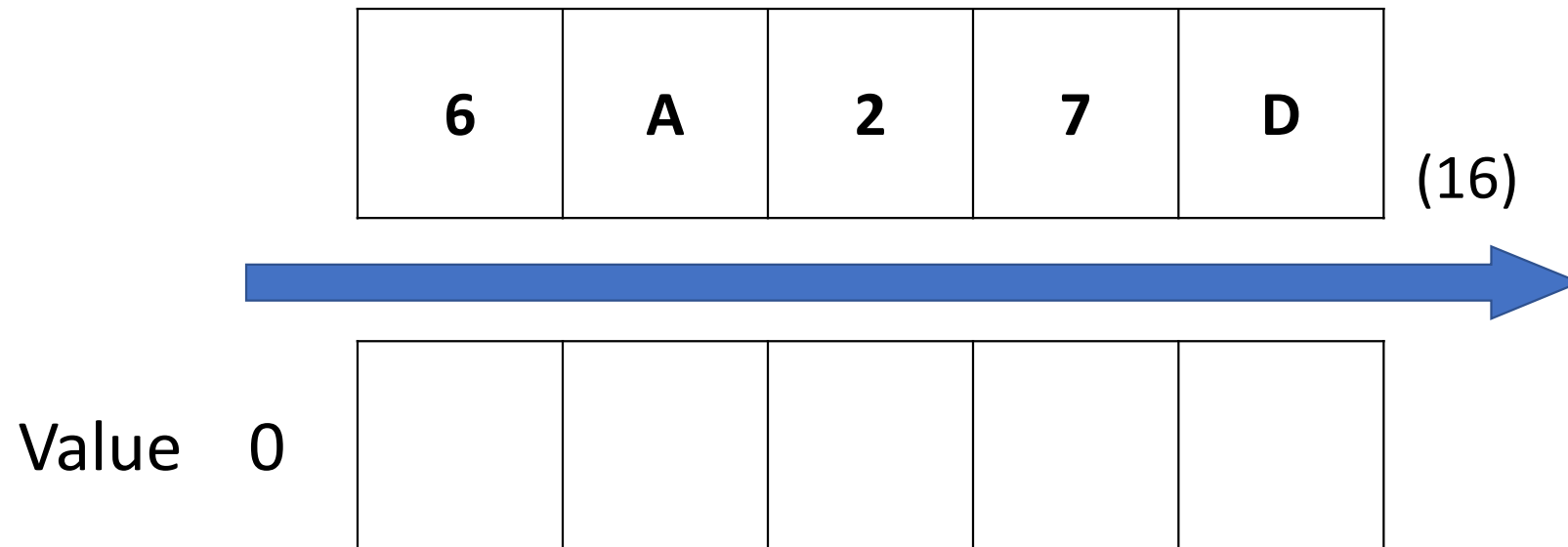
매 조합마다 진법 변환을 수행하면 된다.

주의할 점

1. 변환 시에 2^{63} 을 넘어가지는 않는가?
2. 등장하는 문자가 진법으로 올바른가?

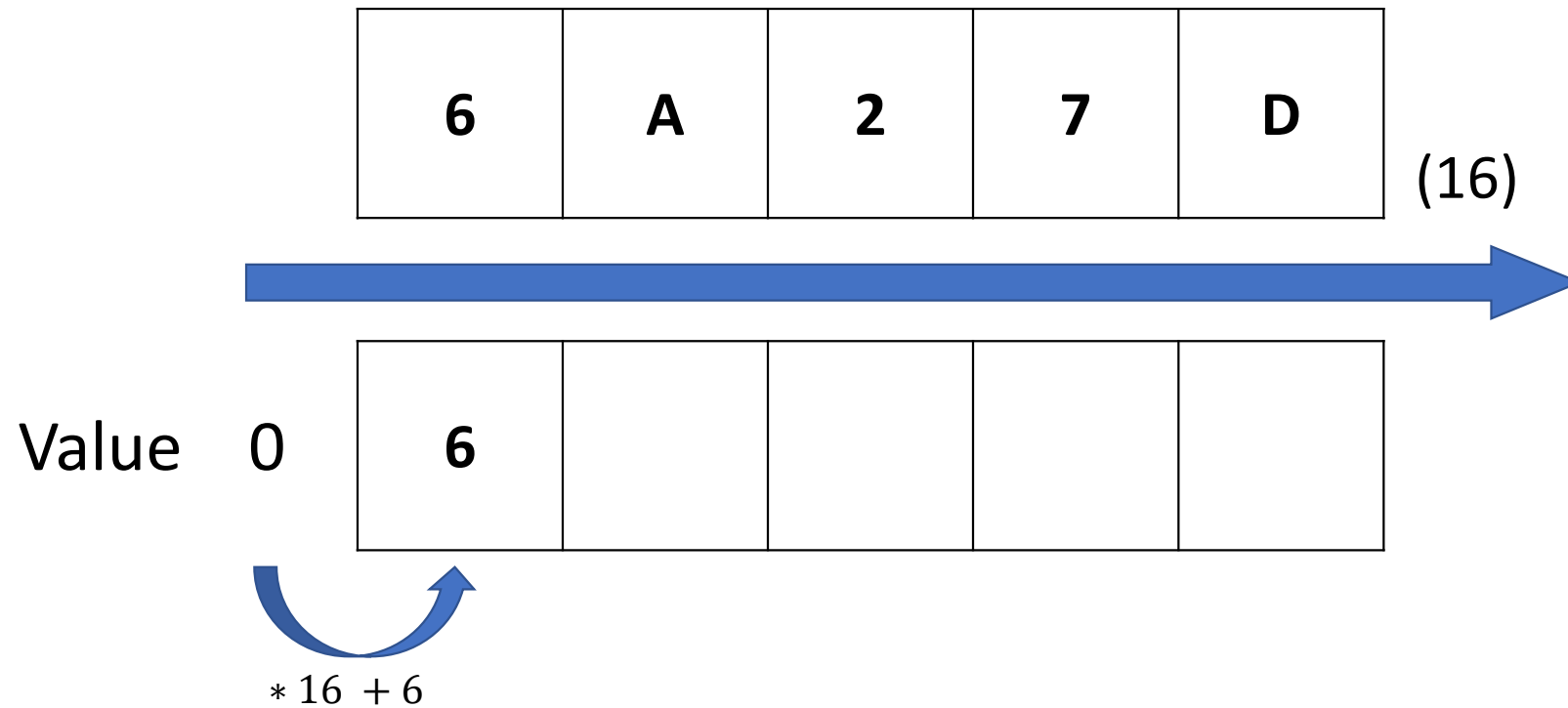
I진법 변환하기

생각보다 코딩 테스트에 자주 등장!



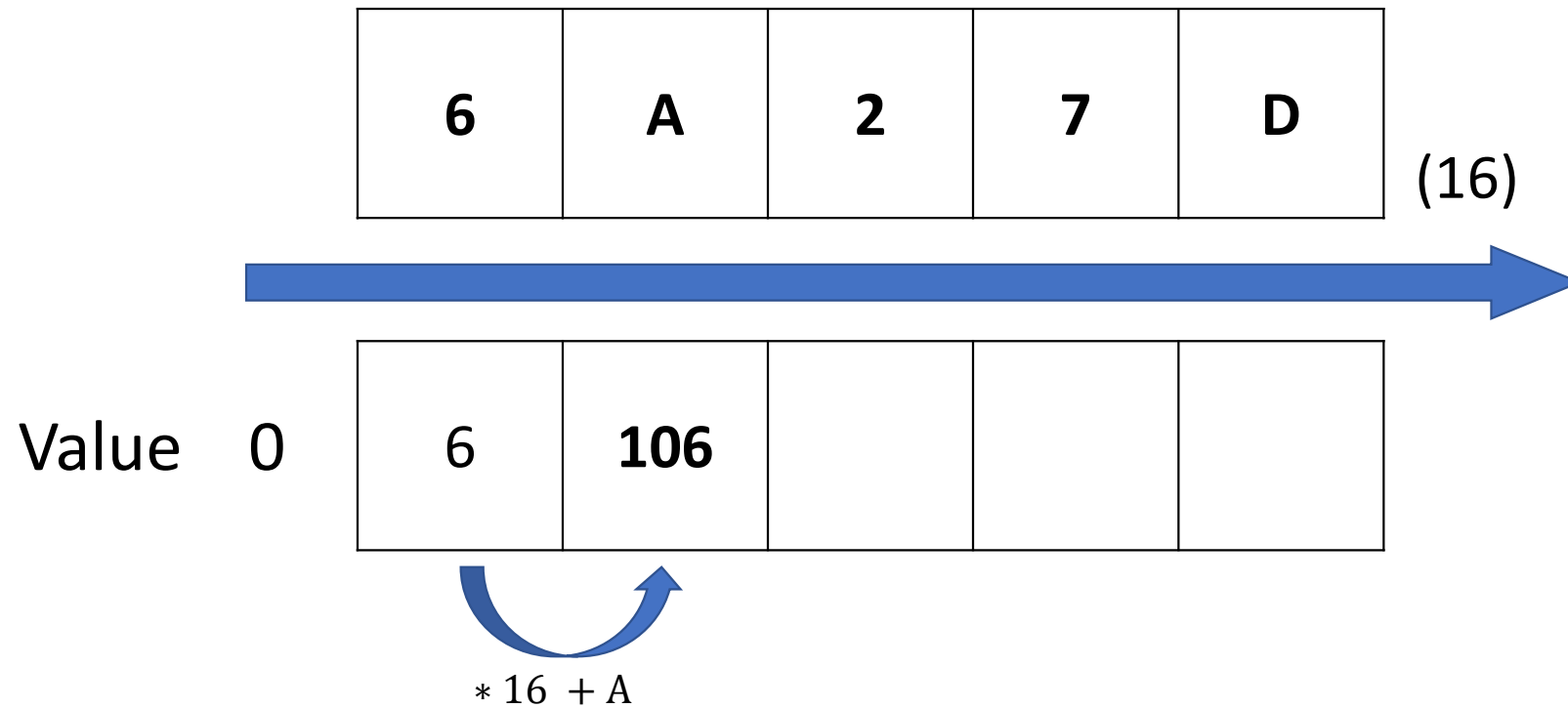
I진법 변환하기

생각보다 코딩 테스트에 자주 등장!



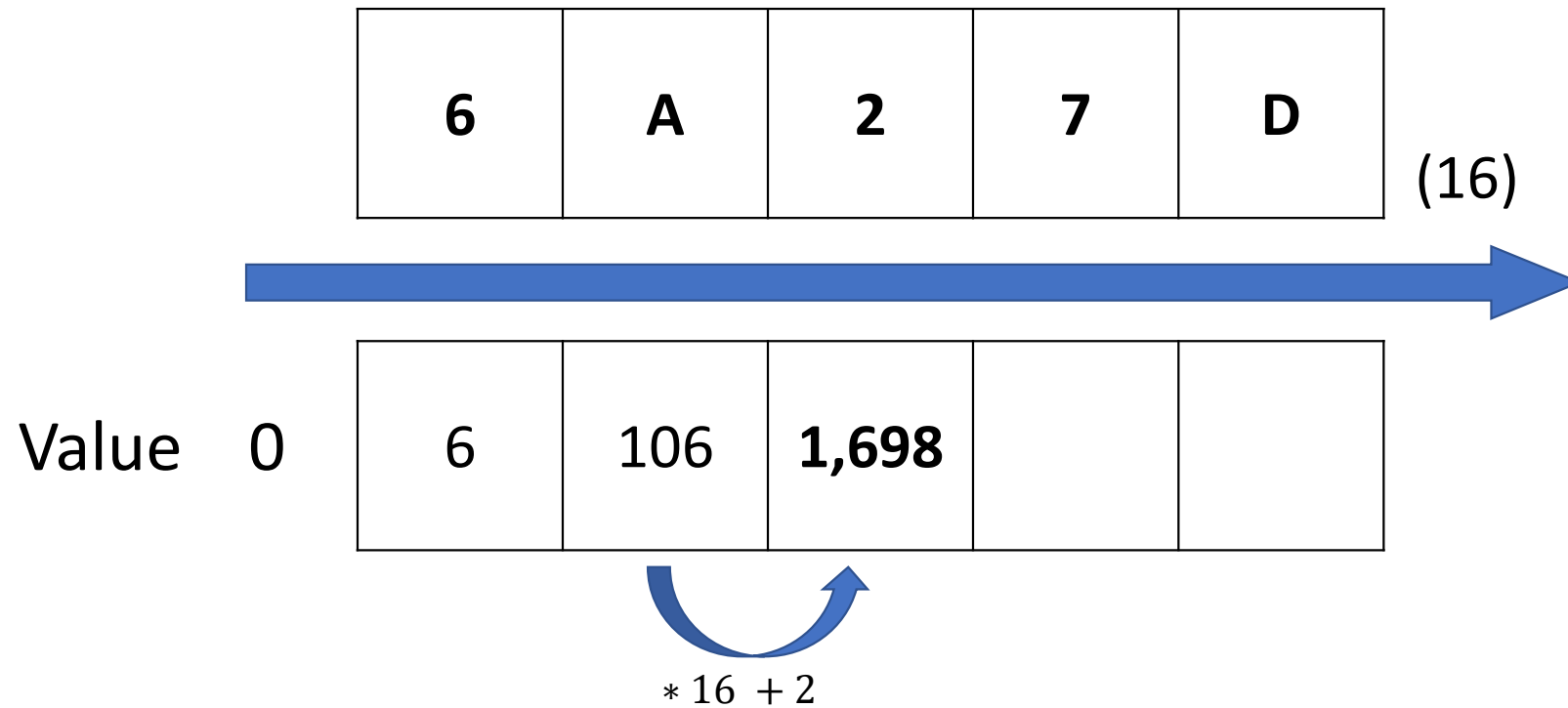
I진법 변환하기

생각보다 코딩 테스트에 자주 등장!



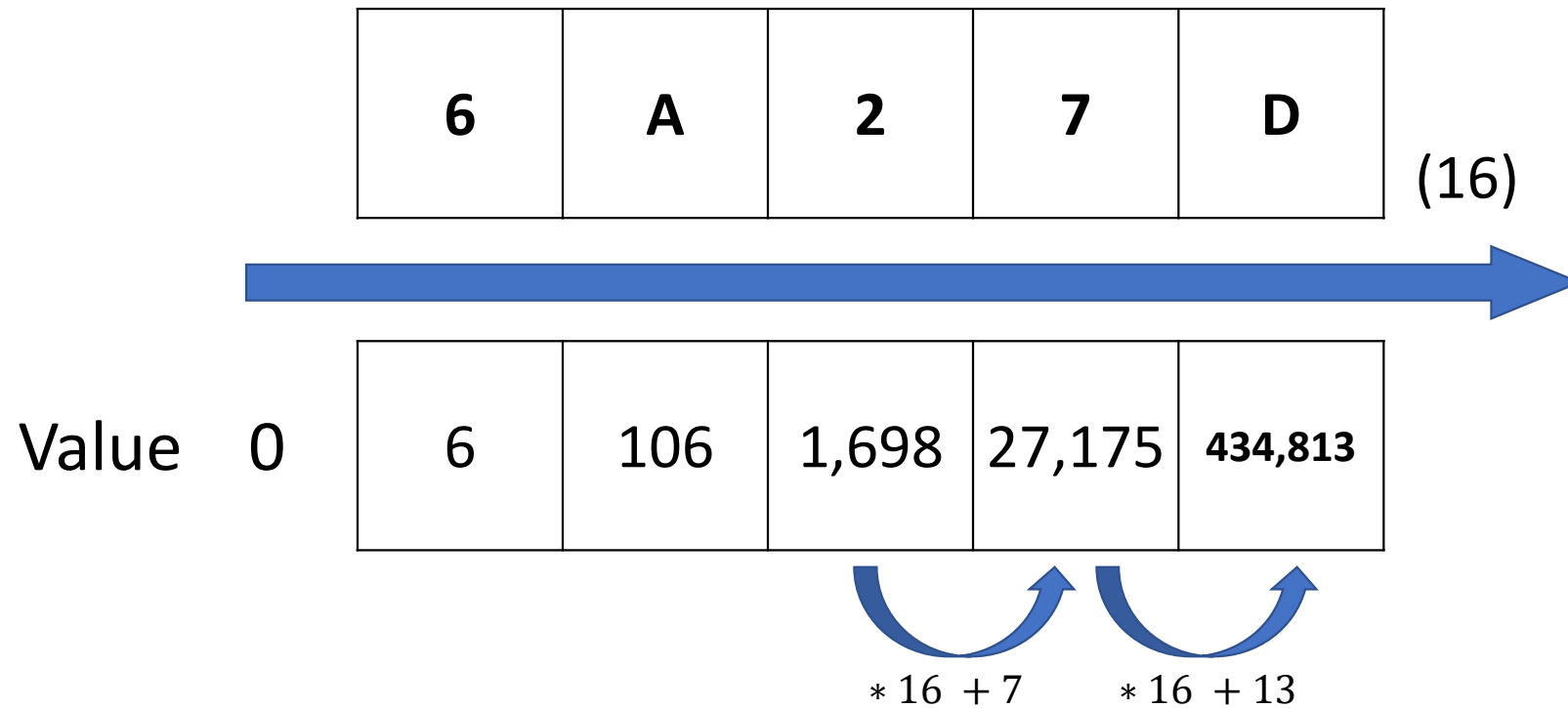
I진법 변환하기

생각보다 코딩 테스트에 자주 등장!



I진법 변환하기

생각보다 코딩 테스트에 자주 등장!



I 시간, 공간 복잡도 계산하기

A, B의 조합의 경우의 수가 $35 * 35 = 1,225$ 개이다.

진법 변환은 문자열의 길이만큼 소요되므로, 연산 횟수는
약 $35 * 35 * 70 = 85,750$ 에 비례한다.

구현

```
static int conv(char x) {
    if ('0' <= x && x <= '9') return x - '0';
    return x - 'a' + 10;
}

// str 을 base 진법으로 변환이 가능하면 변환 결과를, 불가능하면 -1 을 리턴하는 함수
static long possible(String str, int base) {
    /* TODO */
}

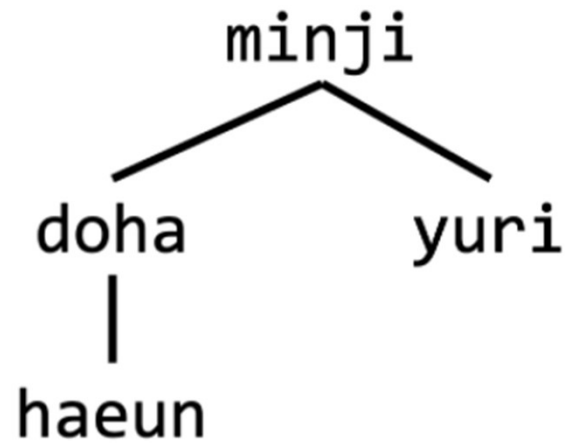
static void pro() {
    /* TODO */
}
```

BOJ 21276 – 계보 복원가 호석

난이도: 2

$1 \leq \text{사람 수}, N \leq 1,000$

$0 \leq \text{정보 개수}, M \leq N \times (N - 1)/2$



I 출제 의도

관련 강의: Graph, Tree

- 그래프의 정점이 **문자열**인 경우는 어떻게 하는가?
- 그래프, 그 중에서도 **Rooted Tree**에 대한 올바른 이해를 했는가?

I Rooted Tree의 용어 (Tree 강의 복기)

1. Node

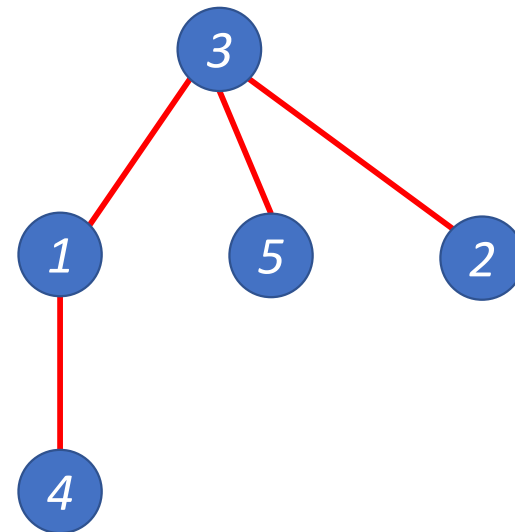
2. Root

3. Depth

4. Parent, Child, Ancestor, Sibling

5. Leaf Node

Rooted Tree



I 생각의 흐름 - 1. 정점을 번호로 바라보기

그래프의 간선을 저장하는 방식은 정점이 **숫자**일 때 편하다.

인접 행렬 \rightarrow `con[i][j]` = i번 정점과 j번 정점이 연결되어 있다.

인접 리스트 \rightarrow `con[i].push_back(j)`

I 생각의 흐름 - 1. 정점을 번호로 바라보기

정점마다 주어지는 이름을 숫자로 바꾸고 싶다.

자료구조 중 `HashMap<String, Integer>` 을 사용하자!

문자열을 숫자로 바꿔서 저장하고, 원하는 문자열을 탐색하는 행위를 모두 $O(1)$ 에 수행해주기 때문에 이와 같은 상황에서 매우 좋은 자료구조이다.

I 생각의 흐름 – 2. Root 찾기

모든 정점이 자신의 **조상**을 기억하고 있다.

한 가문의 시조, 즉, **Root** 정점들은 조상이 존재하지 않는다.

즉, 조상이 없는 정점들을 Root라고 판단하면 된다.

I 생각의 흐름 - 3. 직속 부모, 자식 관계 찾기

어떤 정점이 자신의 조상을 전부 기억한다면?

➔ 자신의 Depth를 계산할 수 있다.

모든 정점이 자신의 Depth를 안다면, 부모 정점은?

➔ 당연히도 자신보다 Depth가 1 낮은 정점일 것이다.

즉, Depth를 통해 조상들 중에 부모를 찾을 수 있다.

I 생각의 흐름

1. 정점을 번호로 나타내기 (문자열과 자료구조에 능숙하면 회피 가능)
2. 각 가문의 시조, Root 찾기
3. Depth를 이용해서 부모 자식 관계를 통해 그래프 복원하기

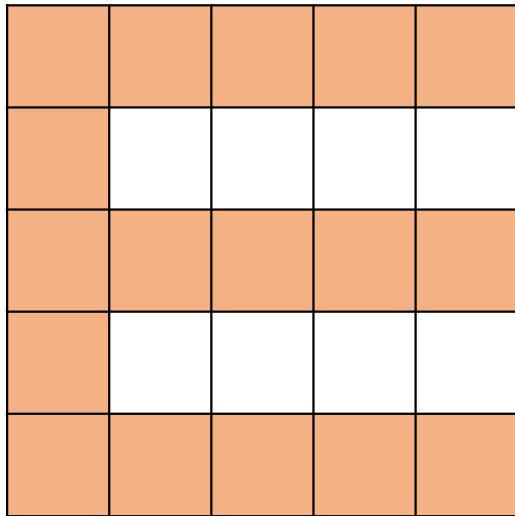
I 구현

```
static void input() {  
    FastReader scan = new FastReader();  
    /* TODO */  
}  
  
static void pro() {  
    /* TODO */  
  
    // 하단은 출력 부분입니다.
```

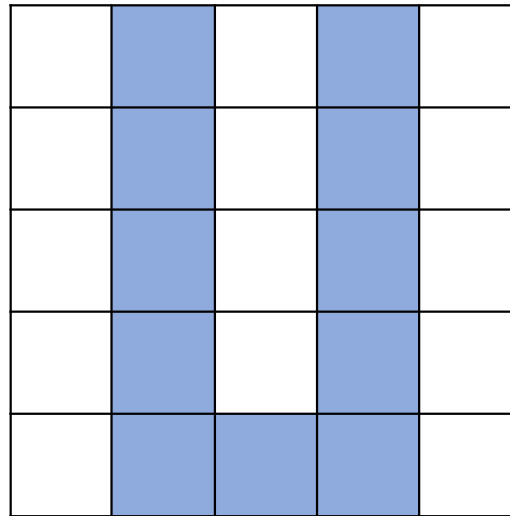
I BOJ 21277 – 짝돌이 호석

난이도: 3

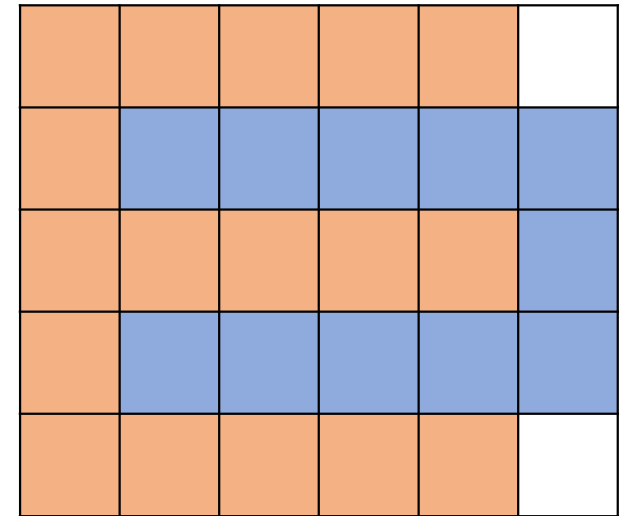
$1 \leq \text{퍼즐 행, 열 크기} \leq 50$



+



=



I 출제 의도

2차원 배열이 주어졌을 때,

1. 배열 평행 이동이 능숙한가?
2. 배열 돌리기가 능숙한가?

두 가지에 대한 구현력을 확인하는 문제입니다.

I 생각의 흐름 - 1. 비교 순서

먼저, 두 퍼즐을 모두 회전시켜 볼 필요가 있을까?

아니다. 첫번째 퍼즐은 고정시키고 두번째 퍼즐만 **4방향 회전**을 시켜도 된다!

I 생각의 흐름 - 1. 비교 순서

같은 이유로 첫번째 퍼즐은 고정시키고 두번째 퍼즐만 **평행 이동** 시켜보면 된다.

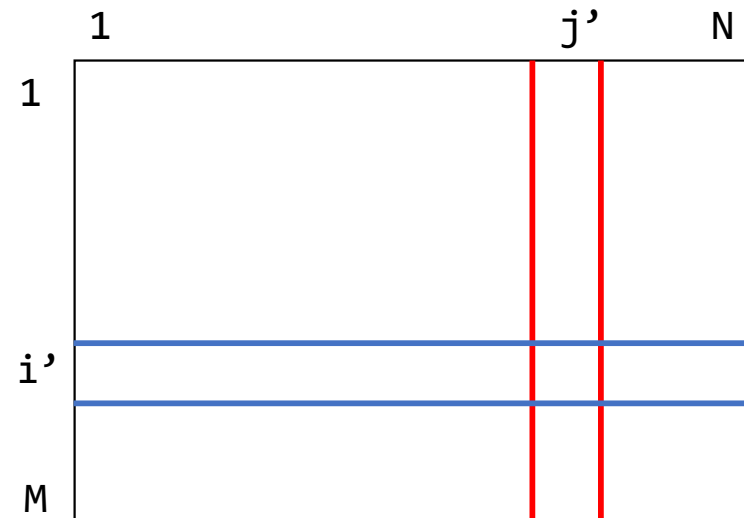
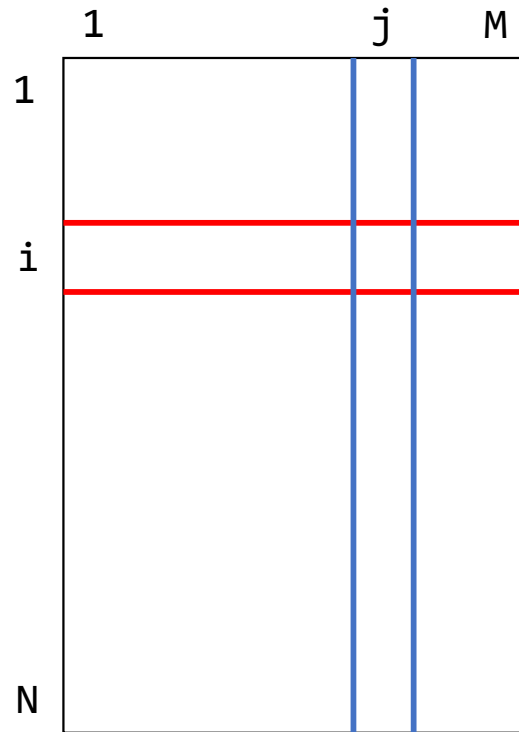
for (두번째 퍼즐 회전 4번):

for (모든 x 축 평행 이동):

for (모든 y 축 평행 이동):

두 퍼즐이 겹치지 않으면 정답 갱신

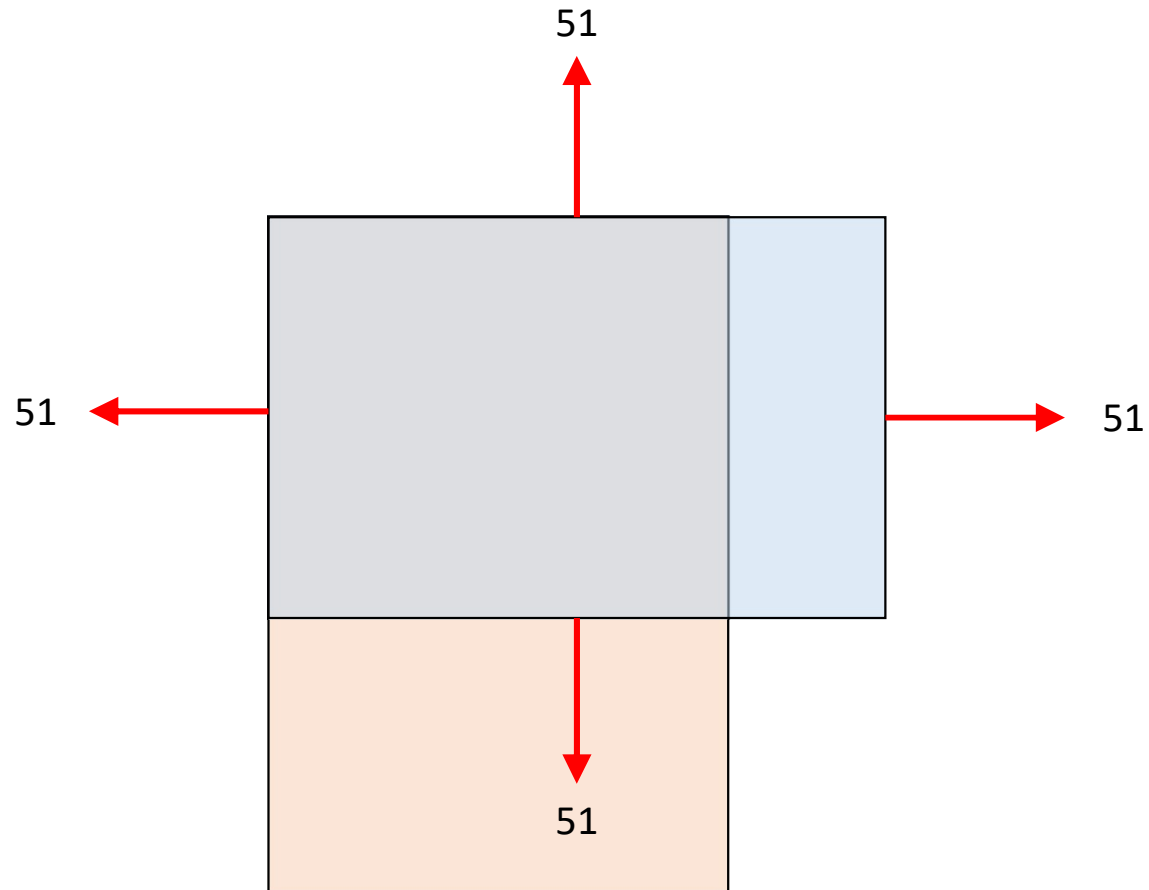
I 생각의 흐름 - 2. 배열 회전



$$\begin{aligned} i' &= j \\ j' &= N - i + 1 \end{aligned}$$

Chapter. 03 모의 코딩테스트 풀이 - 2

I 생각의 흐름 - 3. 평행 이동



I 시간, 공간 복잡도 계산하기

1. 4번의 회전
2. 가로 방향으로 약 100 번의 이동 가능성
3. 세로 방향으로 약 100 번의 이동 가능성
4. 배열 전체에 대해 겹치는 부분 확인 – $O(NM)$

즉 총 연산 횟수는 $4 * 100 * 100 * 50 * 50 = 1\text{억}$ 에 비례한다.

I 구현

```
// 두 번째 퍼즐 회전 함수
static void rotate() {
    /* TODO */
}

// 평행 이동한 결과에서 겹치는 것이 존재하는 지 확인하는 함수
static boolean possible(int shift_row, int shift_col) {
    /* TODO */
}

static void pro() {
    /* TODO */
}
```

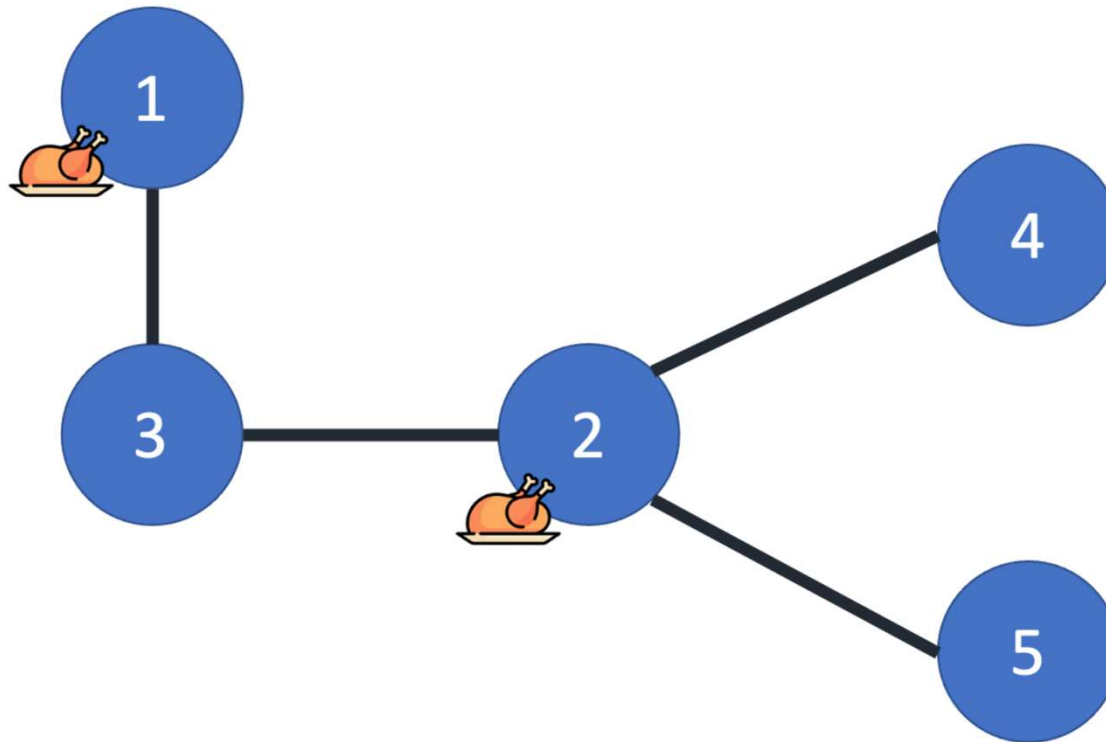
Chapter. 03 모의 코딩테스트 풀이 - 2

I BOJ 21278 – 호석이 두 마리 치킨

난이도: 4

$2 \leq \text{건물 개수}, N \leq 100$

$N - 1 \leq \text{도로 개수}, M \leq N \times (N - 1) / 2$



FAST CAMPUS
ONLINE

류호석 강사.

I 출제 의도

문제를 올바르게 이해했는가?

모든 거리 관계를 파악할 줄 아는가? 전처리를 통해 계산하는 것을 떠올리는가?

BFS를 통한 최단 거리를 떠올리고 구현하는가?

I 생각의 흐름 – 1. 최단 시간 키워드 발견

최단 거리 알고리즘

<배운 것>

1. BFS

2. Dijkstra

둘의 장단점은? 시간 복잡도는? 이 문제에 적합한 것은?

I생각의 흐름 - 1. 최단 시간 키워드 발견

최단 거리 알고리즘

<배운 것>

1. BFS → 간선의 가중치 모두 동일, $O(N)$

2. Dijkstra → 음이 아닌 가중치, $O(E \log V)$

둘의 장단점은? 시간 복잡도는? 이 문제에 적합한 것은?

I 생각의 흐름 - 2. 가능한 모든 조합으로 치킨집을 세워볼까?

치킨집을 지을 건물 조합

$(1,1), (1,2), \dots, (N-1, N) \rightarrow O(N^2)$ 가지

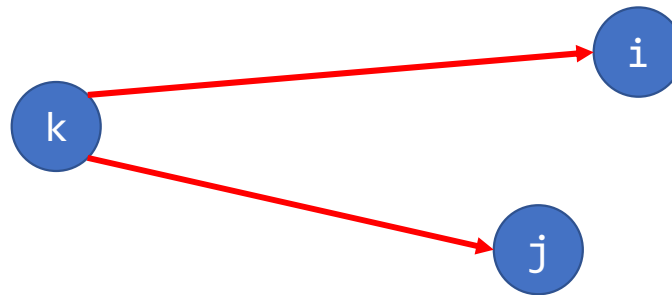
모든 조합을 모두 시도해보자.

I 생각의 흐름 - 2. 가능한 모든 조합으로 치킨집을 세워볼까?

i 번 건물과 j 번 건물에 치킨집을 세운다고 하자.

“모든 건물에서 가장 가까운 치킨집까지 왕복하는 최단 시간의 총합”

을 구하기 위해서는 **모든 건물**에 대해 i 번 건물과 j 번 건물까지의 최단 거리를 계산해야만 한다.



I 생각의 흐름 - 2. 가능한 모든 조합으로 치킨집을 세워볼까?

```
for i = 1 ~ N:
```

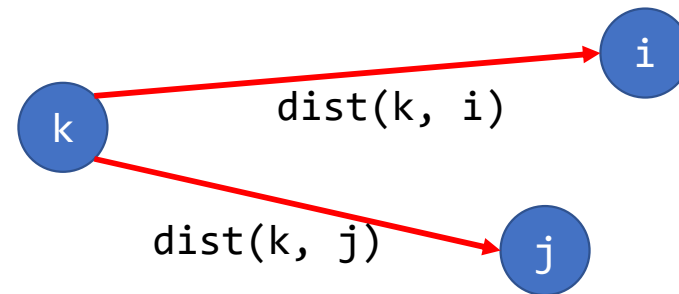
```
    for j = i+1 ~ N:
```

```
        cnt = 0 // i, j에 건물을 세울 때의 왕복 시간 합
```

```
        for k = 1 ~ N:
```

```
            cnt += min( dist(k, i), dist(k, j) )
```

```
        ans = min(ans, cnt)
```



I 생각의 흐름 - 3. 두 건물 x 와 y 사이의 최단거리 $\text{dist}(x, y)$

$\text{dist}(x, y)$ 는 x 를 시작점으로 BFS를 수행하면 알 수 있다.

시간복잡도는 $O(N + M)$ 이다.

I 생각의 흐름 - 3. 두 건물 x와 y 사이의 최단거리 $\text{dist}(x, y)$

for $i = 1 \sim N$: $\rightarrow O(N)$

for $j = i+1 \sim N$: $\rightarrow O(N)$

$\text{cnt} = 0$ // i, j 에 건물을 세울 때의 왕복 시간 합

for $k = 1 \sim N$: $\rightarrow O(N)$

$\text{cnt} += \min(\text{dist}(k, i), \text{dist}(k, j)) \rightarrow \text{BFS}, O(M)$

$\text{ans} = \min(\text{ans}, \text{cnt})$

총 $O(N^3 * M) \rightarrow 100^3 * 5000 = 50\text{억} \rightarrow \text{시간초과!}$

I 생각의 흐름 - 3. 두 건물 x 와 y 사이의 최단거리 $\text{dist}(x, y)$

문제점: dist 함수가 너무 많이 호출된다.

해결책: 미리 계산해 놓을 수 있다면 $O(1)$ 에 $\text{dist}(x, y)$ 를 가져온다!

$D[i][j] := i$ 에서 j 로 가는 최단 거리

I 생각의 흐름 - 3. 두 건물 x와 y 사이의 최단거리 $\text{dist}(x, y)$

for $i = 1 \sim N$: $\rightarrow O(N)$

BFS(i) 를 통해 $D[i, 1 \sim N]$ 을 미리 계산 $\rightarrow \text{BFS}, O(M)$

for $i = 1 \sim N$: $\rightarrow O(N)$

for $j = i+1 \sim N$: $\rightarrow O(N)$

$\text{cnt} = 0$ // i, j 에 건물을 세울 때의 왕복 시간 합

for $k = 1 \sim N$: $\rightarrow O(N)$

$\text{cnt} += \min(D[k][i], D[k][j]) \rightarrow O(1)$

$\text{ans} = \min(\text{ans}, \text{cnt})$

I 시간, 공간 복잡도 계산하기

전처리를 통해 불필요한 시간 증가를 해소해야 한다.

이를 통해 $O(N^3)$ 으로 문제를 해결할 수 있다.

I 구현

```
// S 를 시작으로 BFS를 수행하며 거리 계산하기
static void bfs(int S) {
    /* TODO */
}

static void pro() {
    // 전처리로 모든 거리 계산 해두기
    /* TODO */

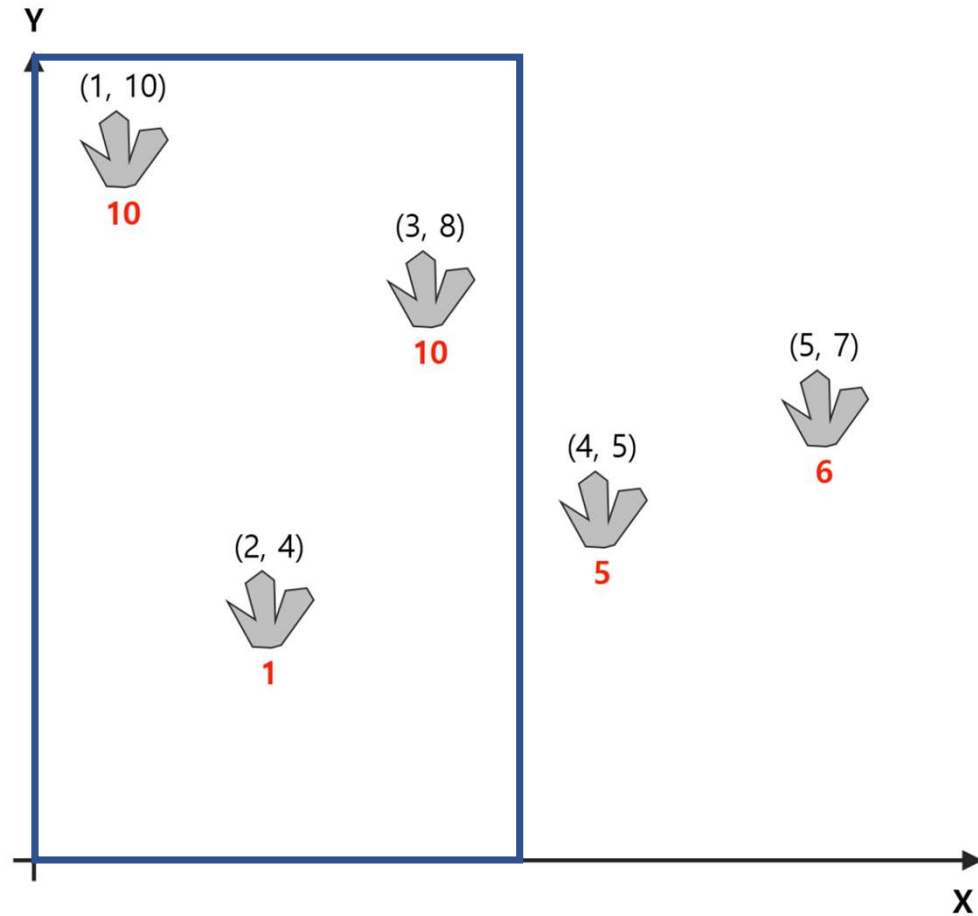
    // 모든 조합으로 세워보고 정답 계산하기
    /* TODO */
}
```

I BOJ 21278 – 광부 호석

난이도: 5

$1 \leq \text{광물 개수}, N \leq 500,000$

$1 \leq \text{좌표 범위} \leq 100,000$



I 출제 의도

“광산 뒤집기” 스킬을 쓰는 영역에 대한 관찰을 충분히 했는가?

H와 W의 관계를 이용해서 투 포인터 기법을 떠올렸는가?

필요한 연산에 적합한 자료 구조를 스스로 떠올릴 수 있는가?

이 모든 과정을 성공적으로 구현할 줄 아는가?

I 생각의 흐름 - 1. 기본 접근 시도

뒤집는 영역의 높이 H 와 너비 W 를 정하고 점수 계산해서 최대치 찾기

```
for H = 1 ~ 100,000:
```

```
    for W = 1 ~ 100,000:
```

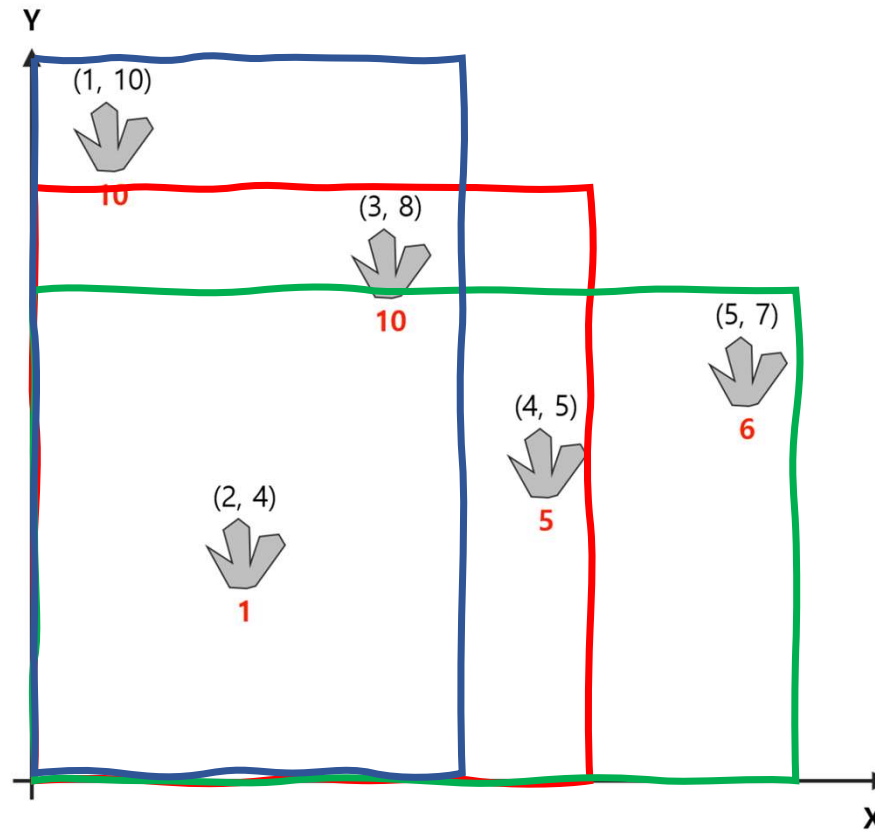
```
        If (H, W)에 포함되는 광석이 k개 이하인가?
```

```
            score = (H, W)로 뒤집으면 얻는 점수
```

```
            ans = max(ans, score)
```

I 생각의 흐름 - 2. 예제에 대한 관찰

주어진 입력을 직접 그려보면서
탐색 여부를 모두 확인해본다.

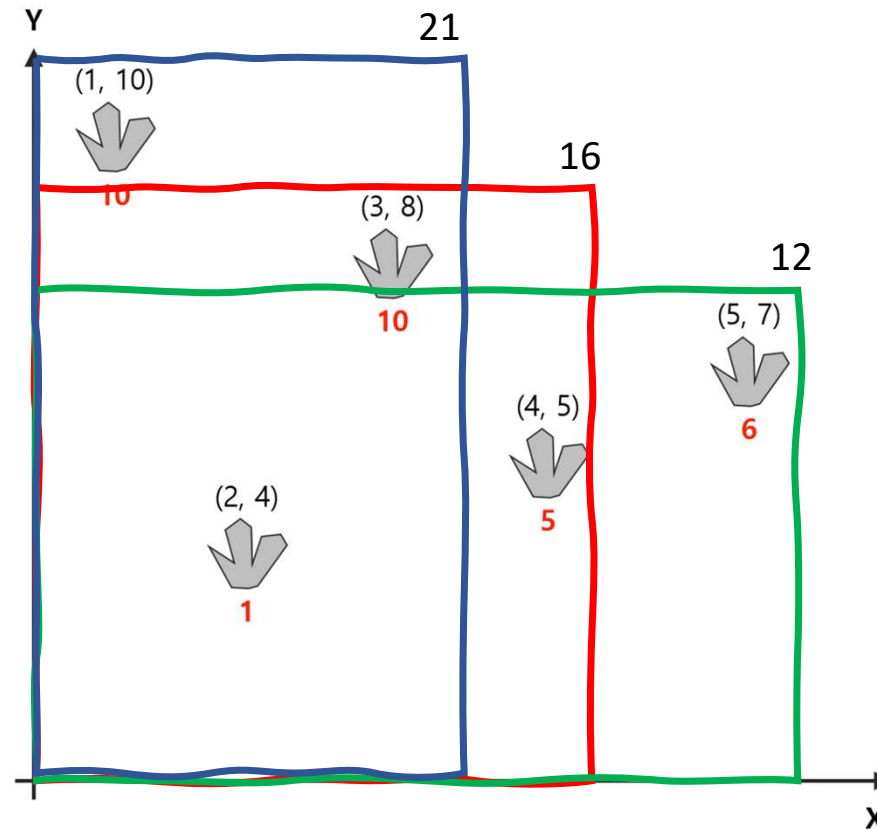


I 생각의 흐름 - 2. 예제에 대한 관찰

주어진 입력을 직접 그려보면서
탐색 여부를 모두 확인해본다.

가능한 “뒤집는 영역”은
가로가 길어질 수록
세로는 짧아진다.

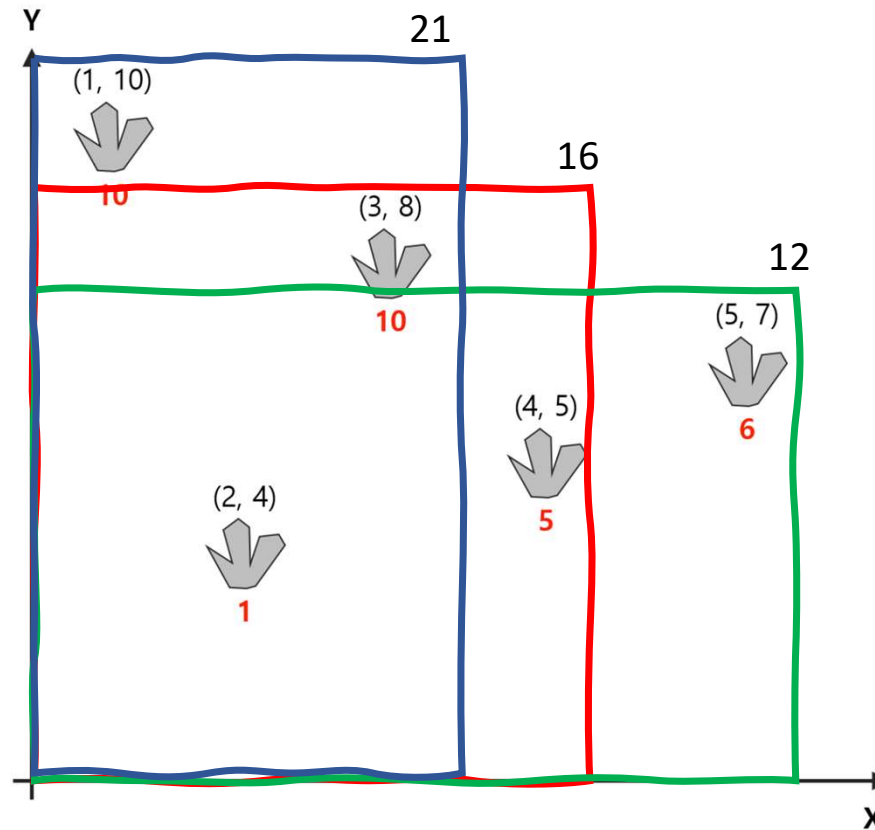
$$\text{가로 길이} \propto \frac{1}{\text{세로 길이}}$$



I 생각의 흐름 - 2. 예제에 대한 관찰

즉 K 개를 포함하는 (H, W) 을
찾았다면, H 가 증가하면 W 는
무조건 같거나 작아야 한다.

➔ Two Pointer 적용 가능!



I 생각의 흐름 - 2. 예제에 대한 관찰

$W = -1$

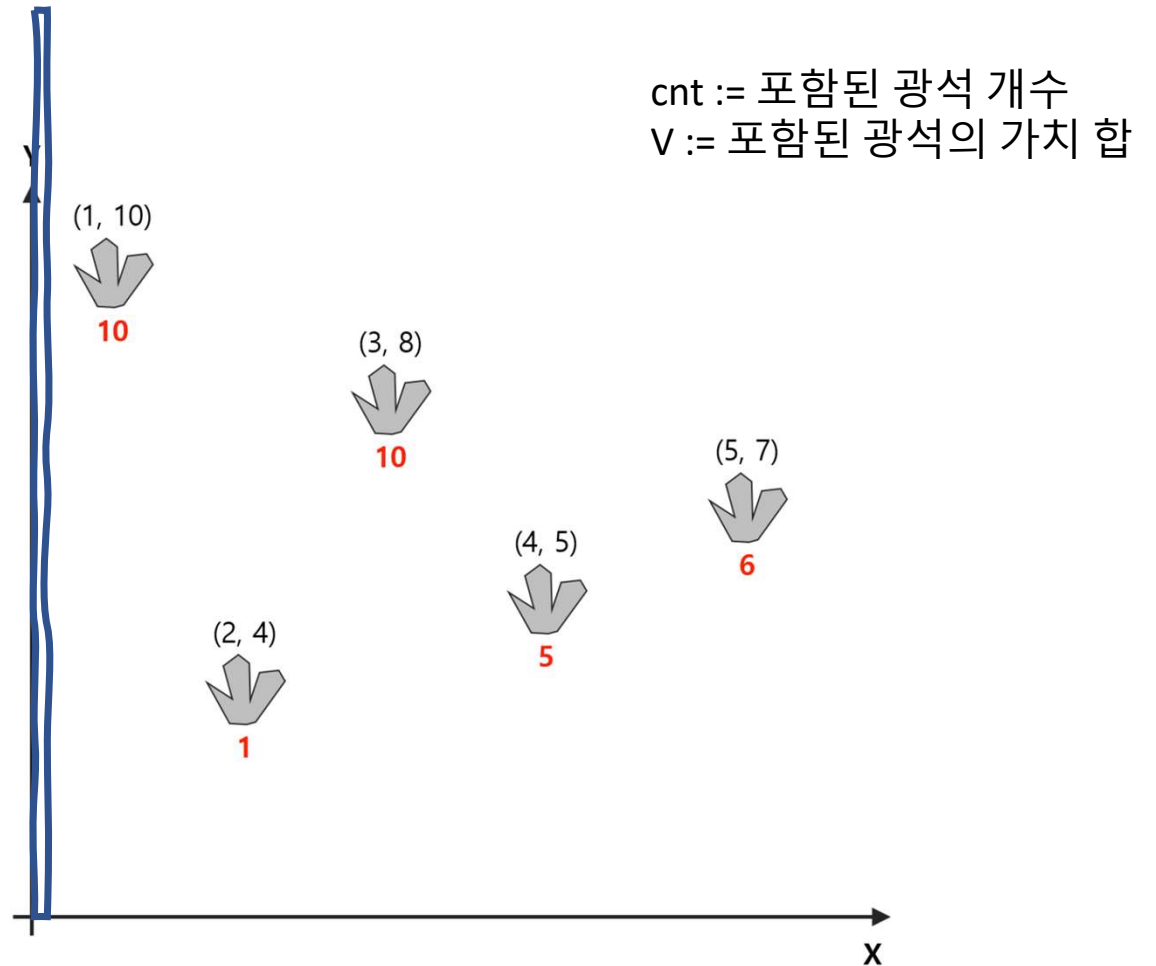
$H = 100,000$

$cnt = 0$

$V = 0$

매 순간 cnt 값을 보고

H 줄이기 or W 늘리기



I 생각의 흐름 - 2. 예제에 대한 관찰

$$W = 1$$

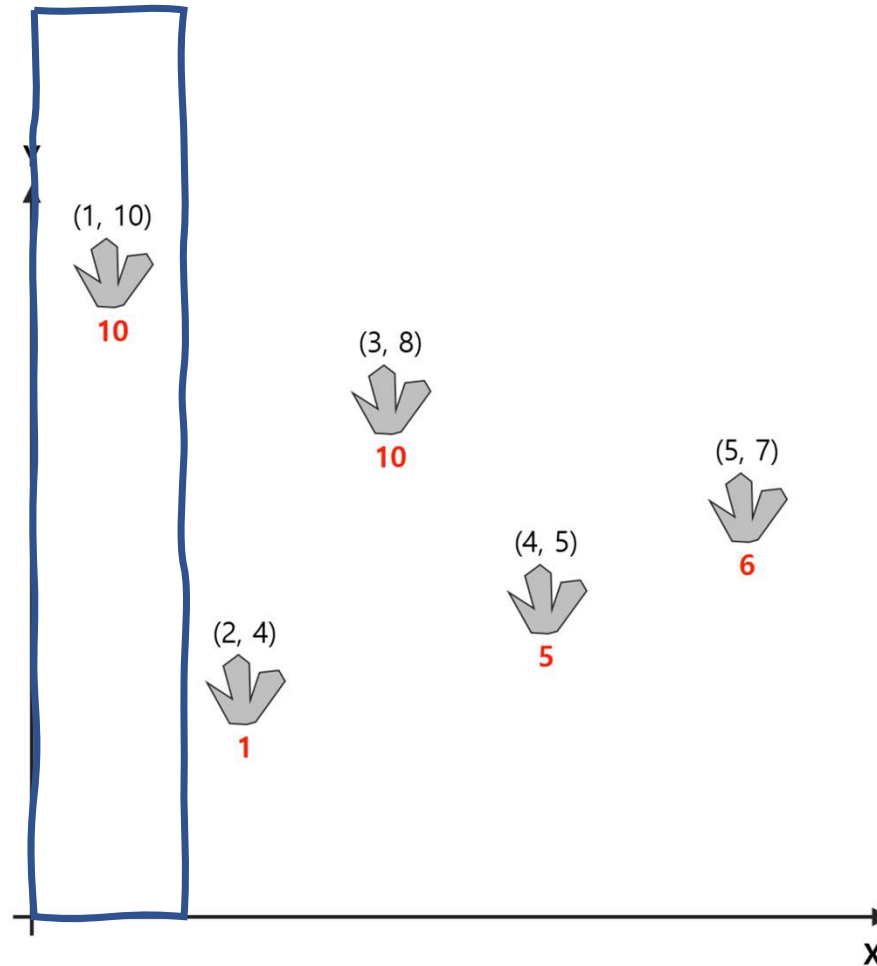
$$H = 100,000$$

$$\text{cnt} = 1$$

$$V = 10$$

매 순간 cnt 값을 보고

H 줄이기 or W 늘리기



I 생각의 흐름 - 2. 예제에 대한 관찰

$$W = 2$$

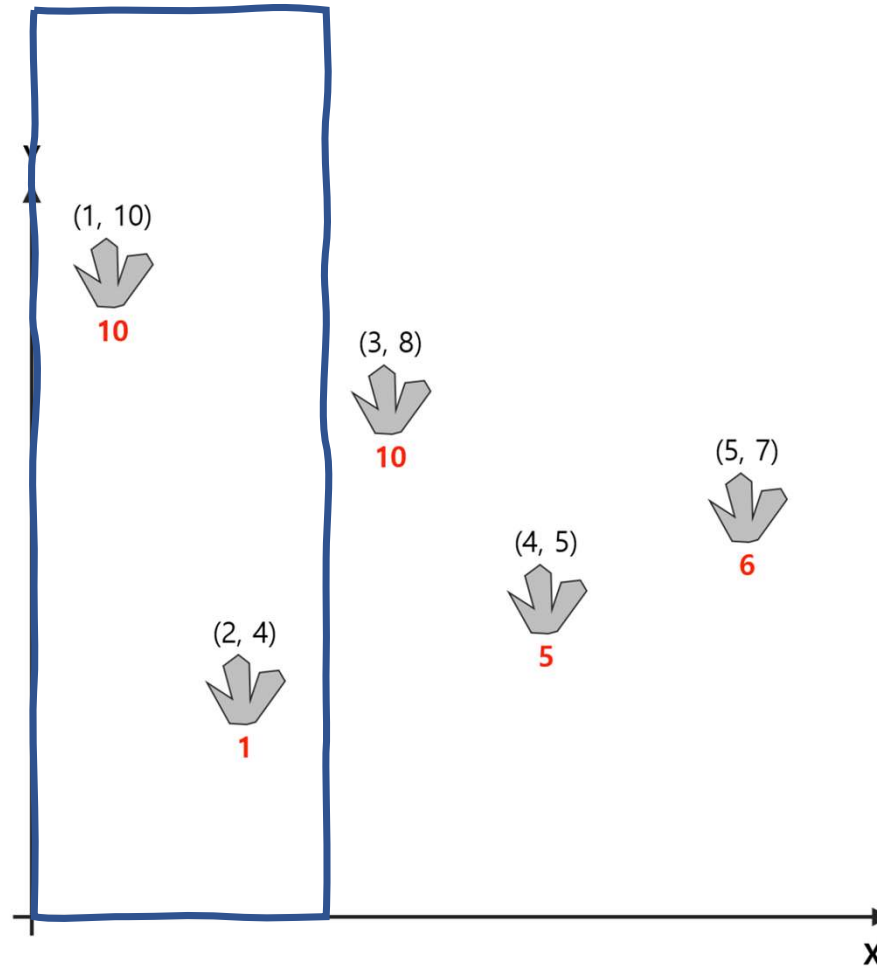
$$H = 100,000$$

$$\text{cnt} = 2$$

$$V = 11$$

매 순간 cnt 값을 보고

H 줄이기 or W 늘리기



I 생각의 흐름 - 2. 예제에 대한 관찰

$$W = 3$$

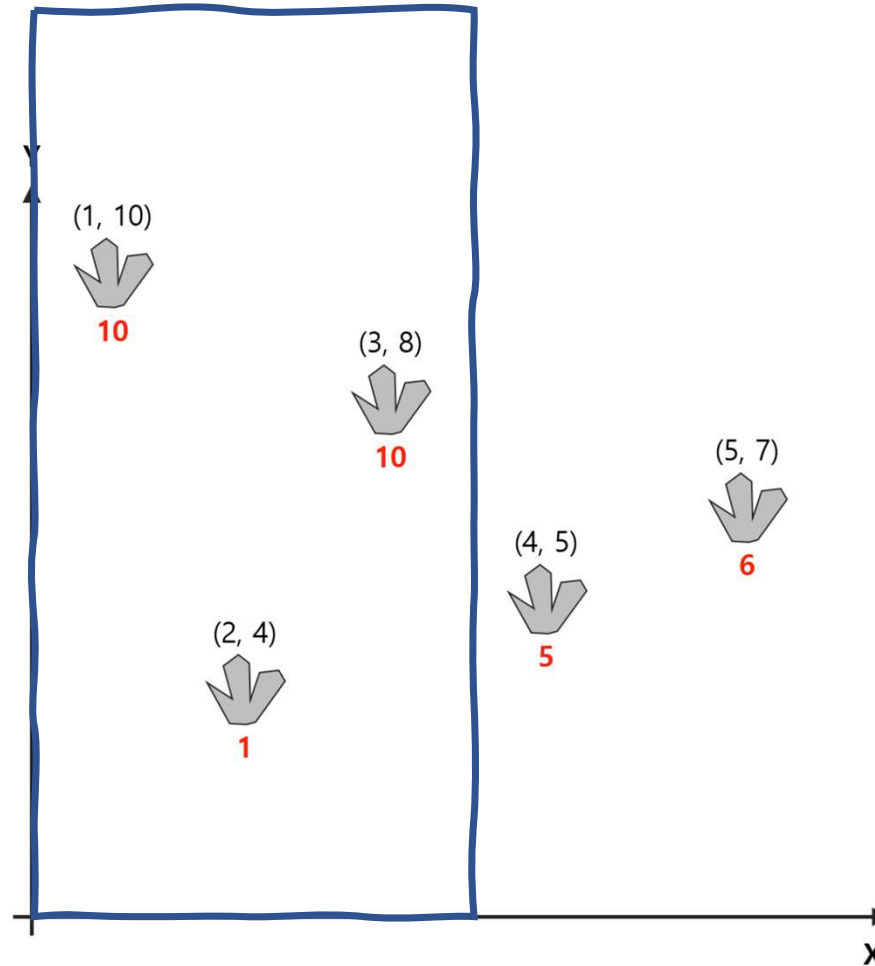
$$H = 100,000$$

$$\text{cnt} = 3$$

$$V = 21$$

매 순간 cnt 값을 보고

H 줄이기 or W 늘리기



I 생각의 흐름 - 2. 예제에 대한 관찰

$$W = 4$$

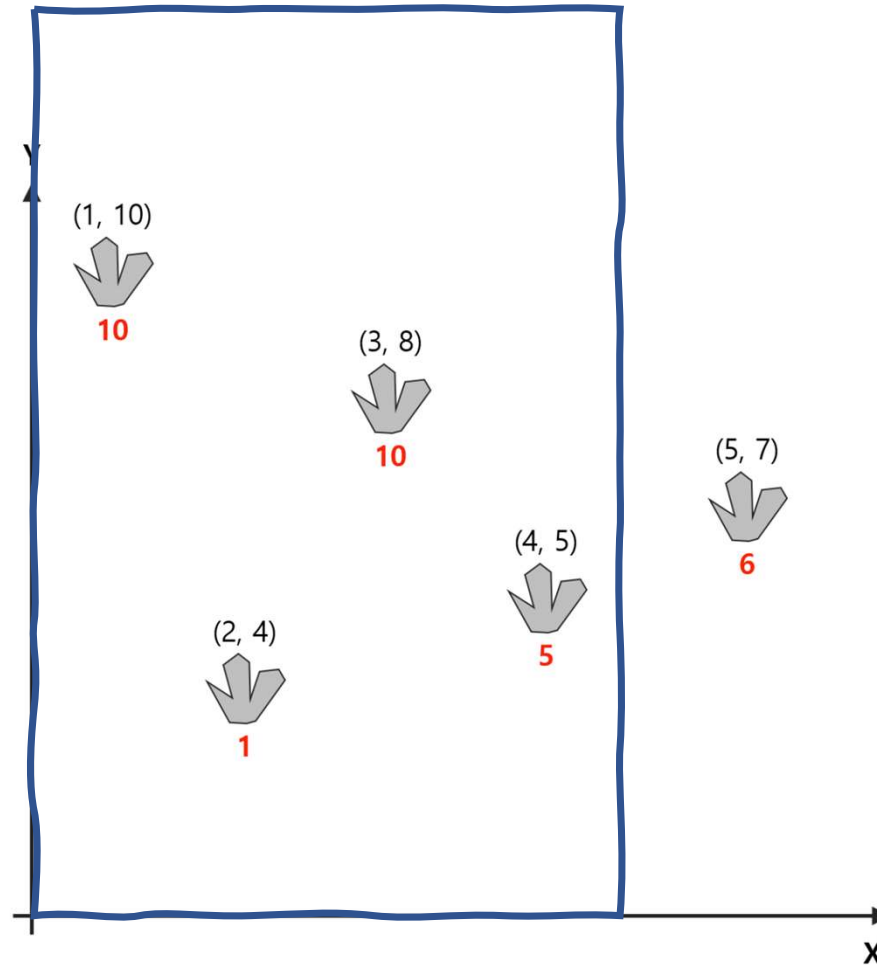
$$H = 100,000$$

$$\text{cnt} = 4$$

$$V = 26$$

매 순간 cnt 값을 보고

H 줄이기 or W 늘리기



I 생각의 흐름 - 2. 예제에 대한 관찰

$$W = 4$$

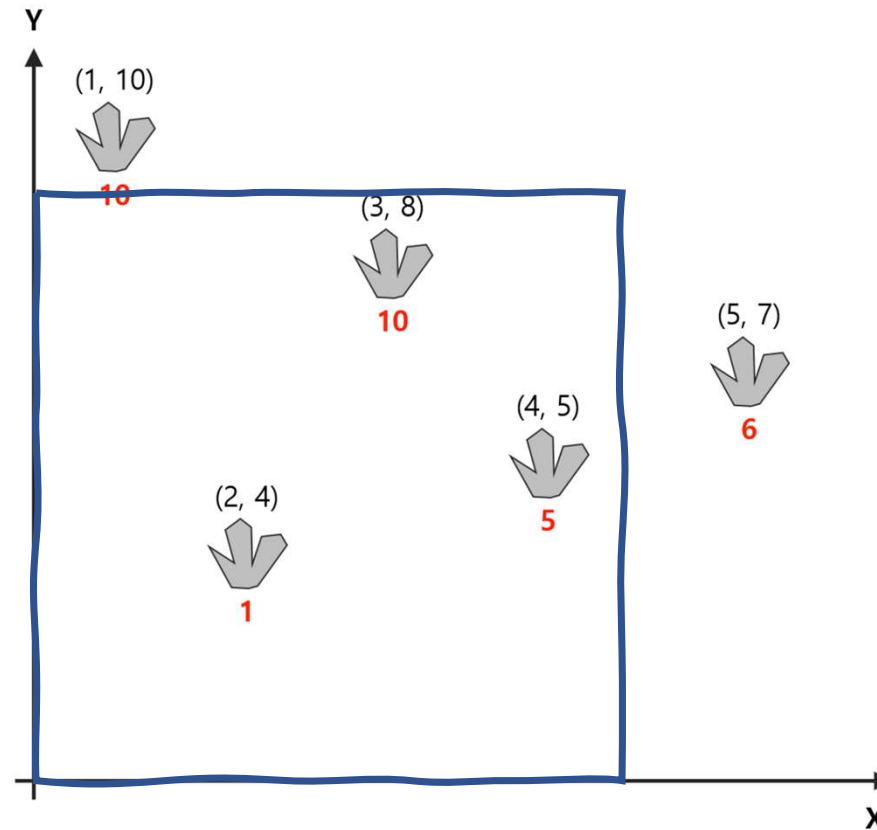
$$H = 9$$

$$\text{cnt} = 3$$

$$V = 16$$

매 순간 cnt 값을 보고

H 줄이기 or W 늘리기



I 생각의 흐름 - 2. 예제에 대한 관찰

$$W = 5$$

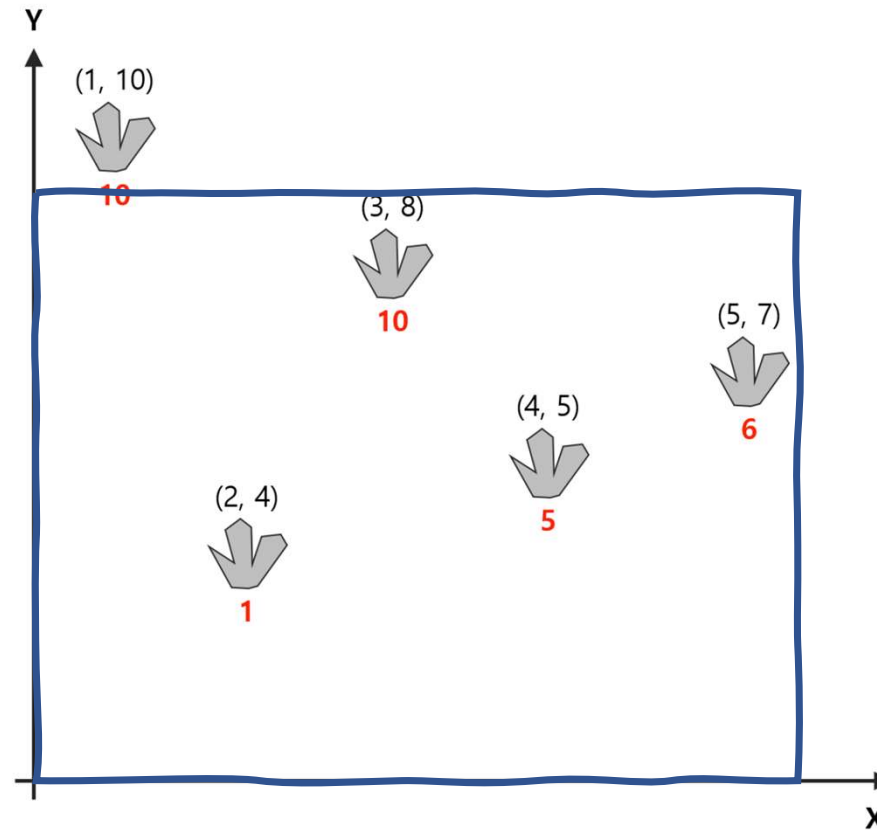
$$H = 9$$

$$\text{cnt} = 4$$

$$V = 22$$

매 순간 cnt 값을 보고

H 줄이기 or W 늘리기



I 생각의 흐름 - 2. 예제에 대한 관찰

$$W = 5$$

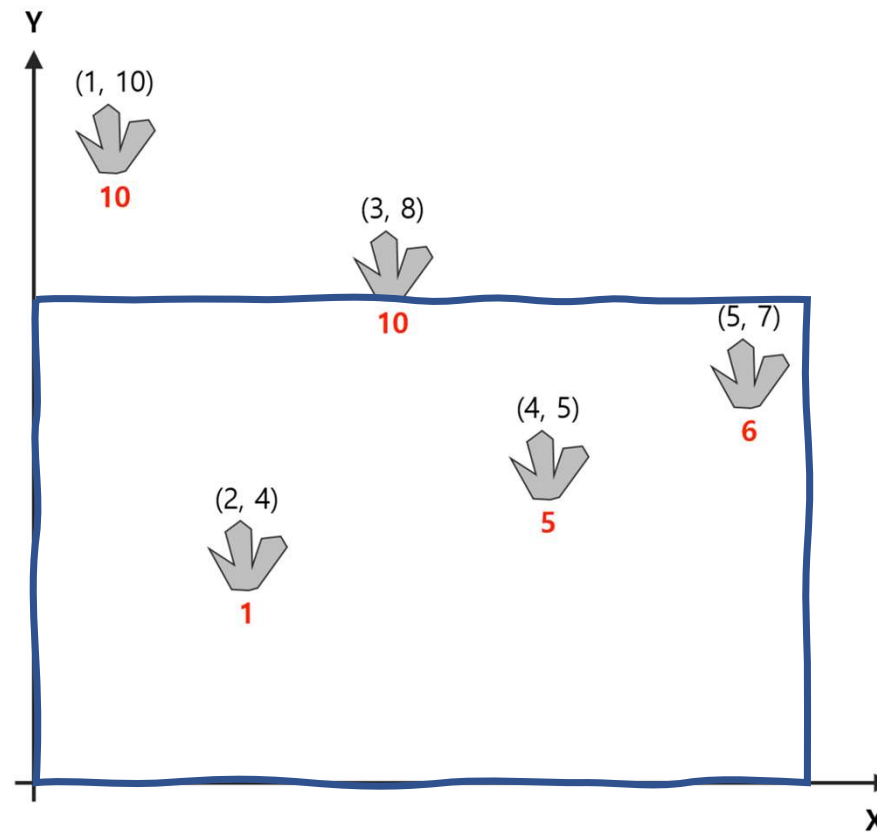
$$H = 7$$

$$\text{cnt} = 3$$

$$V = 12$$

매 순간 cnt 값을 보고

H 줄이기 or W 늘리기



I 생각의 흐름 - 2*. Two Pointer 적용해보기

$W = -1$ / $H = 100,000$ / $\text{cnt} := (H, W)$ 에 포함된 광석 개수 = 0

While $W \leq 100,000$ && $H \geq 0$:

If $\text{cnt} > C$: → 영역이 너무 크니까 높이를 낮춰라
 높이 낮추기

else: → 더 넓힐 수 있으니까 너비를 넓혀라
 너비 넓히기

I 생각의 흐름 - 3. 영역 변경 시에 추가, 제거되는 광석 찾기

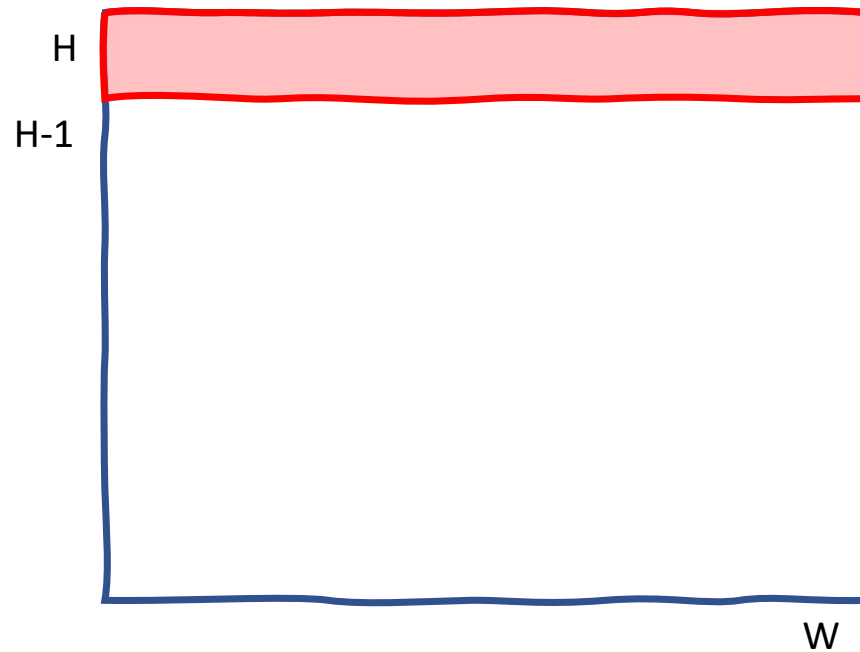


Add($x = W + 1$, $y = H$)

초록 영역 안에 있는 광석 추가

이후 $W++$

I 생각의 흐름 - 3. 영역 변경 시에 추가, 제거되는 광석 찾기


 $\text{Del}(y = H, x = W)$

빨간 영역 안에 있는 광석 추가
이후 H--

I 생각의 흐름 - 3*. 함수화 하기

$W = -1, H = 100,000, cnt := (H, W)$ 에 포함된 광석 개수 = 0

While $W \leq 100,000 \ \&\& \ H \geq 0$:

If $cnt > C$: → 영역이 너무 크니까 높이를 낮춰라

Del($H--$, W)

else: → 더 넓힐 수 있으니까 너비를 넓혀라

Add($++W$, H)

I 생각의 흐름 - 4. 자료구조 만들기

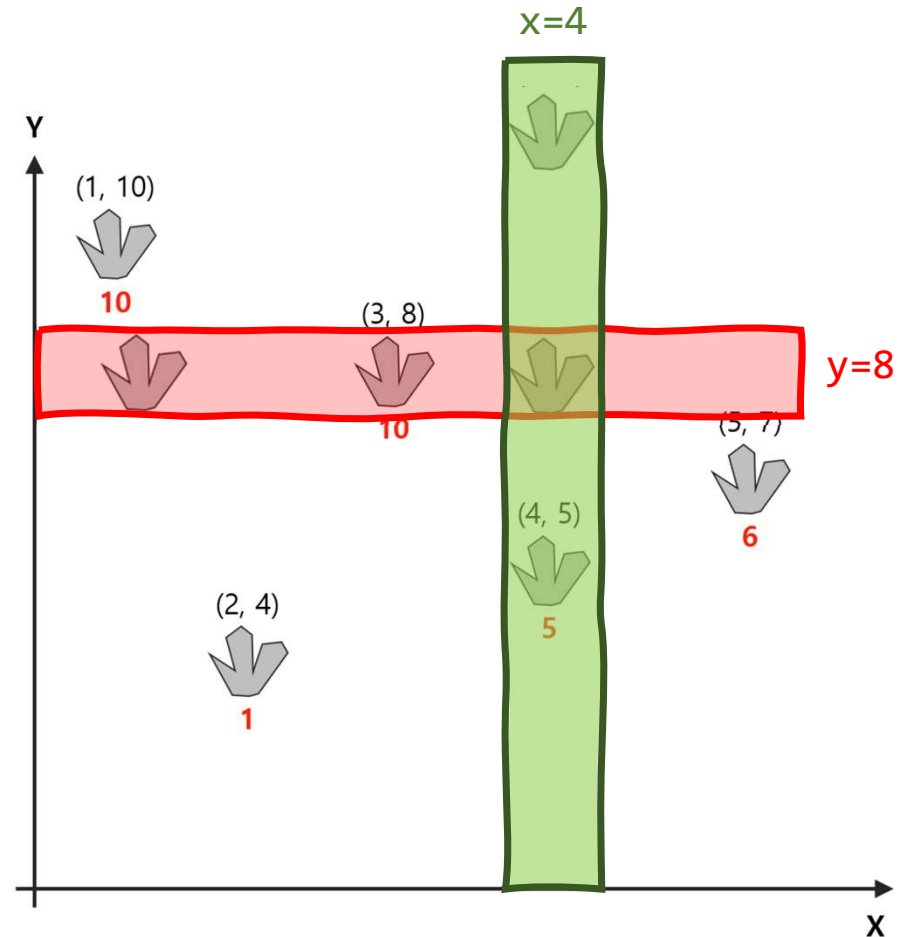
필요한 연산

1. 특정 x 좌표의 광석들 정보 받아오기

→ $X[x] = \{x\text{값인 광석들}\}$

2. 특정 y 좌표의 광석들 정보 받아오기

→ $Y[y] = \{y\text{값인 광석들}\}$



I 생각의 흐름 - 4*. 함수 구현 하기

Del(y, x): ($\leq x$, y) 인 광석들 찾기

→ Y[y] 에 들어있는 광석들만 주욱 훑어보면 된다.

Add(x, y): (x , $\leq y$) 인 광석들 찾기

→ X[x]에 들어있는 광석들만 주욱 훑어보면 된다.

X[1...100,000], Y[1...100,000] 을 모두 보더라도 결국 $O(N)$ 이다.

I 시간, 공간 복잡도 계산하기

1. X, Y 자료 구조 만들기 $\rightarrow O(N)$
2. W, H의 변화 총량 $O(20만)$
3. Del, Add 함수가 모든 W, H에 대해 호출 되어도 $O(N)$

즉, 총 시간 복잡도는 $O(\max(N, 20만))$ 이므로 시간 안에 충분히 나온다.

구현

```
static void del(int y, int x) {  
    /* TODO */  
}  
  
static void add(int x, int y) {  
    /* TODO */  
}  
  
static void pro() {  
    /* TODO */  
    out.println(ans);  
    out.close();  
}
```