Coder’s high 2016 Round 2 해법 설명 프레젠테이션

2016년 7월 30일

대회 제출 통계

		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
00:00	Yes		1	4	5	13	2	3	5	2	2	7	4	1	2	2		5	1	1	2		1	1	1	1	1	2	1	1	
	No				2	1		1	1	1	2			1	1	1		1			2		2		1	2	2		1		1
00:30	Yes	2					1	1			1		2	1			1	1					1							1	2
	No			1	1	2					1		1						1	1											
01:00	Yes		1			1		1						1		1	1		1		1	1						1			
	No									1	1	1	1			2	1					1				3					3
01:30	Yes	3	2			1					1	1					1			1		1	1	1		1			1		1
	No		1	2		1		1	1		2	4	3		1	2	2		1		1	2	1		1	1	1		1		2
02:00	Yes				1			1	1		1	1		1		1				1	1		2			1					
	No	3	2		1			1	1	1		1	1	3	2	1	1	2		4		2	1	1			2			3	2

대회 제출 통계

		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
02:30	Yes	1	1	1																				1			1				
	No	1	1	1	2	2	2	1		1	2	2		2		2	1		4	2		2	2	1			4	3	1	2	1
03:00	Yes						1		1			1	1					2								1		1		1	
	No	2		2	1	1	2	1	1	2	2	1	1	1	1	1	1		2		1	2	1		1			2	3		
03:30	Yes		1			1				1				1														1			
	No	2		1						2				2	1	1	1				8	6	7	6	4	5	4	4	2	1	1
04:00	Yes				1								1						2		1								1	1	
	No	3	1	2	1	3	5	3	1	3	1	1	4	3	3	2	3	3	2	4	3	3	1	2	4	1	1	4	6	5	5
04:30	Yes				1						1		1	2		1	1											1	1		
	No	4	6		2	4	3	2	3	6	6	5	5	6	2	2	3	4	3	3	5	2	4	4	6	5	1	11	4	6	10
05:00	No	1																													

A. 뉴턴과 사과

- 맞은 팀 수 : 38
- 제출 횟수 : 51
- 정답률: 76.471%
- 처음 맞은 팀: Anti-Q (ch_49, tonyjjw, dotorya, qo) , 1분
- 출제자 : tncks0121 (박수찬)
- 해설 작성자 : tncks0121 (박수찬)

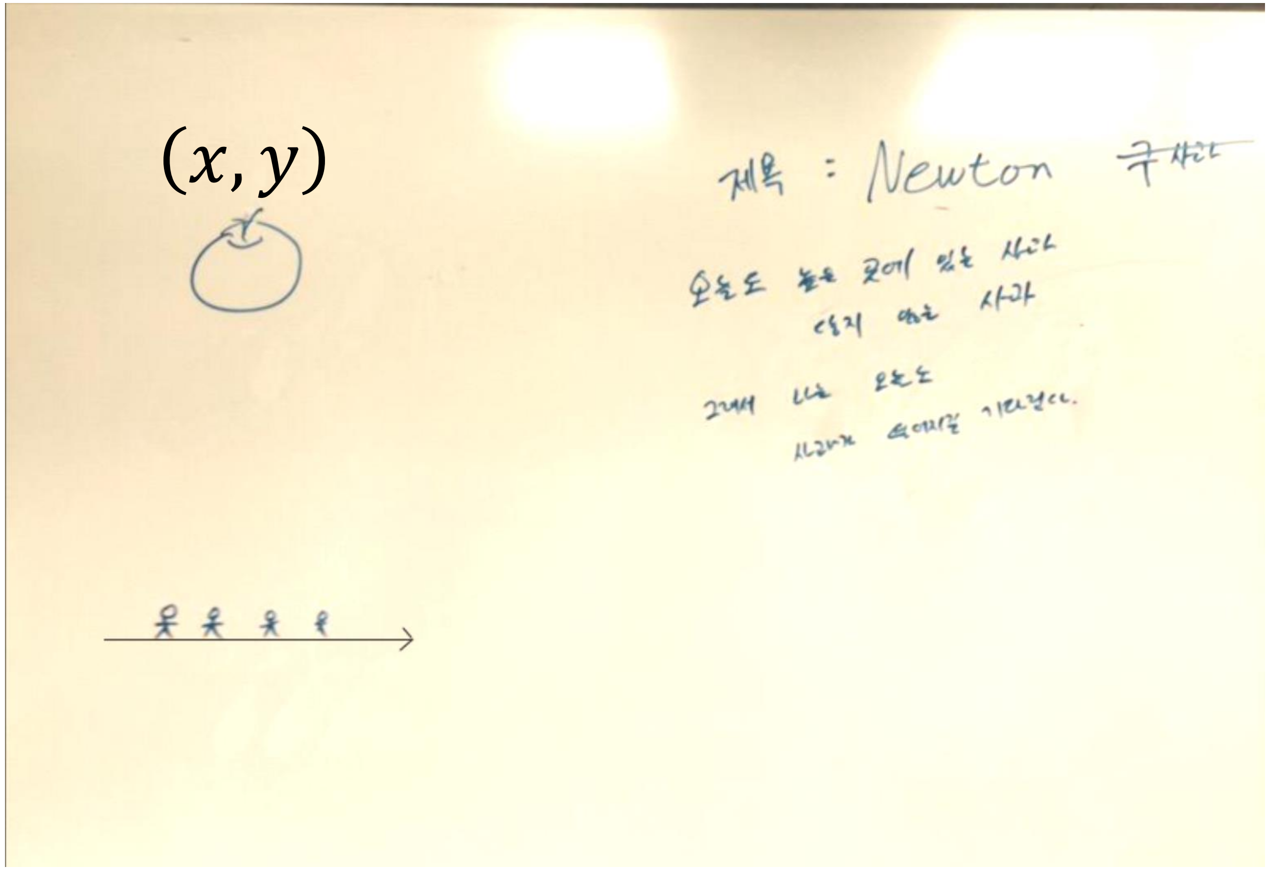
A. 뉴턴과 사과

		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
00:00	Yes		1	4	5	12	2	3	2	1		2	1	1							1						1	1			
	No				2	1		1	1		1					1					1		1								1
00:30	Yes																1						1								
	No																														
01:00	Yes																														
	No																														
01:30	Yes																														
	No																														
02:00	Yes																														
	No																														

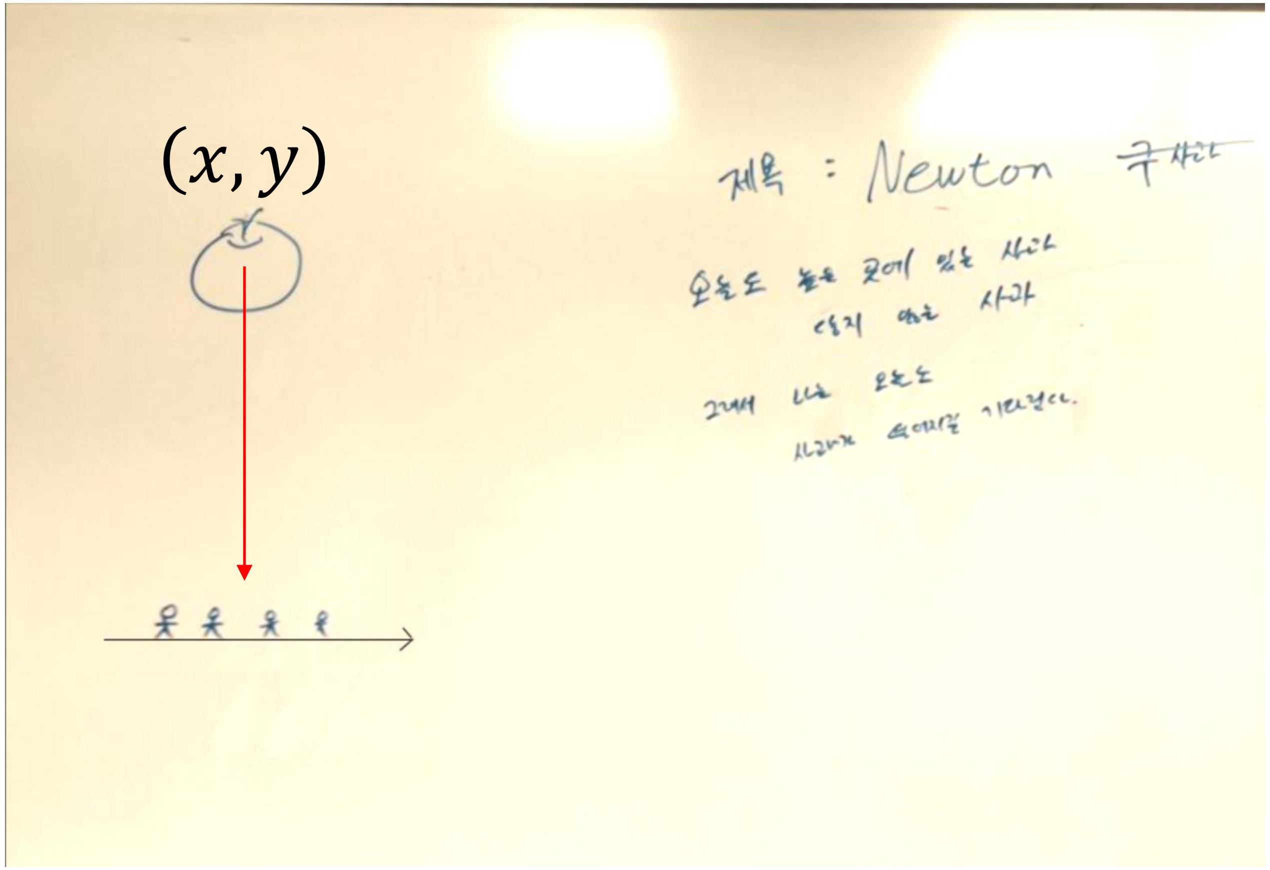
A. 뉴턴과 사과

		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
02:30	Yes																														
	No																														
03:00	Yes																														
	No																												2		
03:30	Yes																														
	No																														
04:00	Yes																														
	No																														
04:30	Yes																														
	No																														

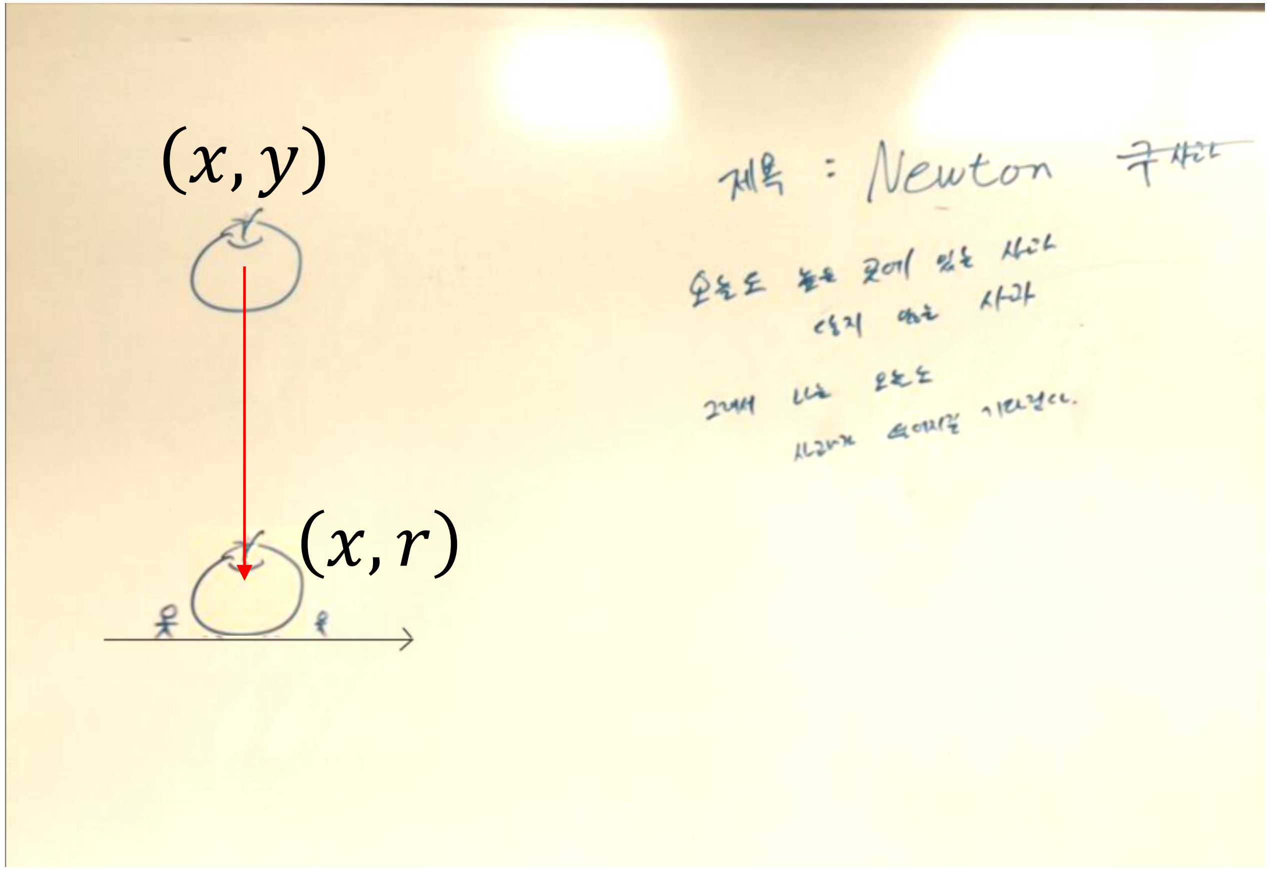
A. 뉴턴과 사과



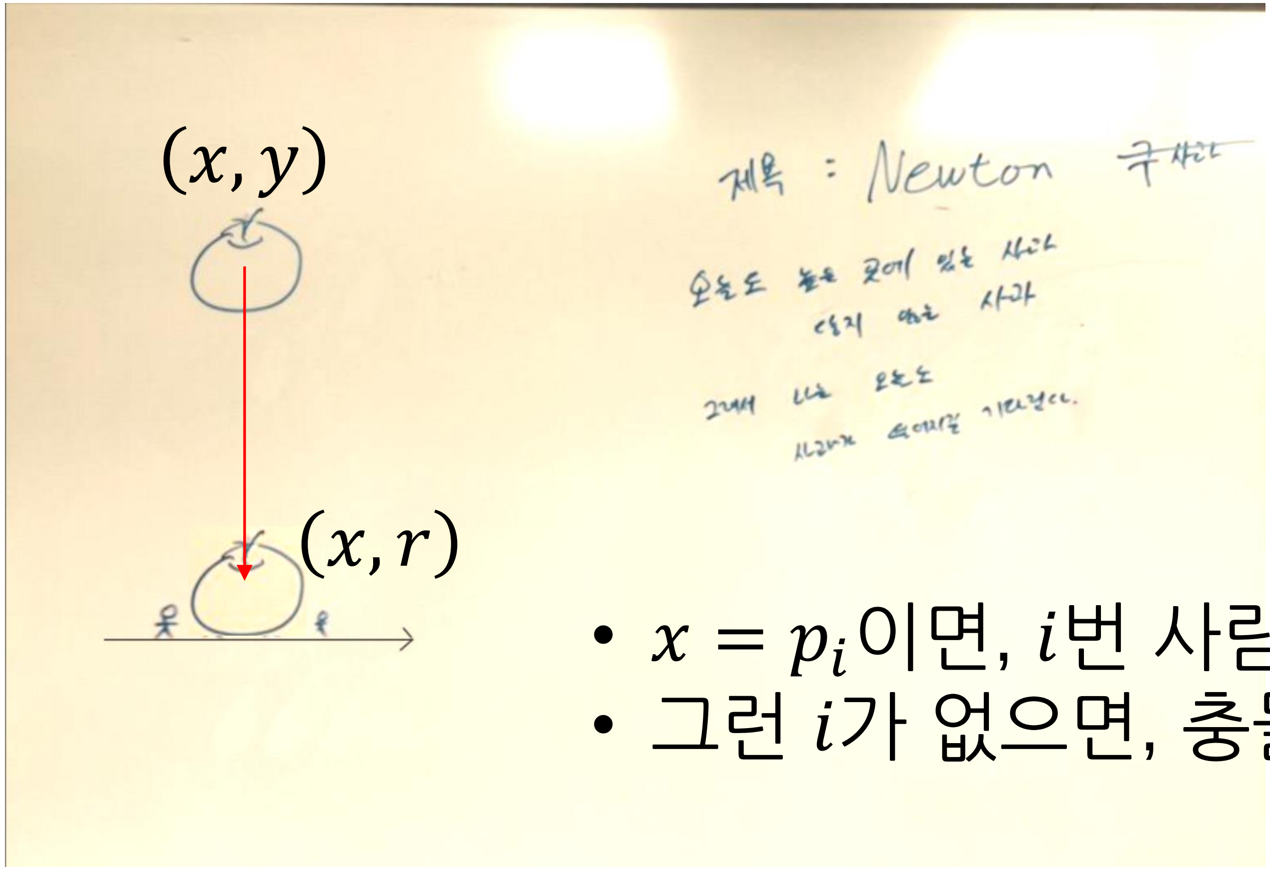
A. 뉴턴과 사과



A. 뉴턴과 사과



A. 뉴턴과 사과



- $x = p_i$이면, i번 사람과 충돌
- 그런 i가 없으면, 충돌하지 않음

B. Mountains Beyond Mountains

- 맞은 팀 수 : 38
- 제출 횟수 : 53
- 정답률: 71.698%
- 처음 맞은 팀: Go-O-Guep-Si-Gye (ch_145, ltg2030, Nyan101, no) , 4분
- 출제자 : koosaga (구재현)
- 해설 작성자 : koosaga (구재현)

B. Mountains Beyond Mountains

		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
00:00	Yes					1			3	1	2	5	3		2	2		5		1	1		1	1	1	1		1	1	1	
	No									1	1			1	1						1		1		1	2	2		1		
00:30	Yes	1									1		2																		
	No					2					1																				
01:00	Yes							1																							
	No																														
01:30	Yes																														
	No																														
02:00	Yes																														
	No																														

B. Mountains Beyond Mountains

```
.....#....
....##....
....##....
----**-*--
.#¦.##.#¦.
.#¦###.##.
.########.
#########.
```

문제에서 시키는 대로 출력하면 되는 문제입니다.
다만..

B. Mountains Beyond Mountains

```
.....#....
....##....
....##....
----**-*--
.#¦.##.#¦.
.#¦###.##.
.########.
#########.
```

$$N, M \leq 10^5$$

B. Mountains Beyond Mountains

```
.....#....
....##....
....##....
----**-*--
.#¦.##.#¦.
.#¦###.##.
.########.
#########.
```

$N, M \leq 10^5$

범위가 꽤 큽니다.
그래서..

B. Mountains Beyond Mountains

```
.....#....
....##....
....##....
----**-*--
.#¦.##.#¦.
.#¦###.##.
.########.
#########.
```

$N, M \leq 10^5$ → $10^5 \times 10^5$ 배열?

B. Mountains Beyond Mountains

```
.....#....
....##....
....##....
----**-*--
.#¦.##.#¦.
.#¦###.##.
.########.
#########.
```

$N, M \le 10^5$ → $\cancel{10^5 \times 10^5}$ 배열?

이런 건 못 만듭니다.
그래서 어떻게 해야 할까요?

B. Mountains Beyond Mountains

```
.....#....
....##....
....##....
----**-*--
.#¦.##.#¦.
.#¦###.##.
.########.
#########.
```

$N, M \le 10^5 \longrightarrow 10^5 \cancel{\times 10^5}$ 배열?

길이가 M인 string으로 크기 N인 배열을 만들어 저장하거나..

`std::string`

B. Mountains Beyond Mountains

```
.....#....
....##....
....##....
----**-*--
.#¦.##.#¦.
.#¦###.##.
.########.
#########.
```

$N, M \leq 10^5$ → $10^5 \cancel{\times 10^5}$ 배열?

`std::string`

`str[x * M + y]`

$N \times M$ 크기 1차원 배열을 만들거나

B. Mountains Beyond Mountains

```
.....#....
....##....
....##....
----**-*--
.#¦.##.#¦.
.#¦###.##.
.########.
#########.
```

$N, M \leq 10^5$ → $10^5 \cancel{\times 10^5}$ 배열?

`std::string`

`str[x * M + y]`

`printf`

답을 애초에 배열에 안 저장하고
if문 등으로 바로 출력하면 됩니다.

C. 트리의 변화

- 맞은 팀 수 : 8
- 제출 횟수 : 22
- 정답률: 36.364%
- 처음 맞은 팀: Andromeda Express (Feat. Kriii) (ch_46, Astein, ainu7, kriii) , 17분
- 출제자 : tncks0121 (박수찬)
- 해설 작성자 : tncks0121 (박수찬)

C. 트리의 변화

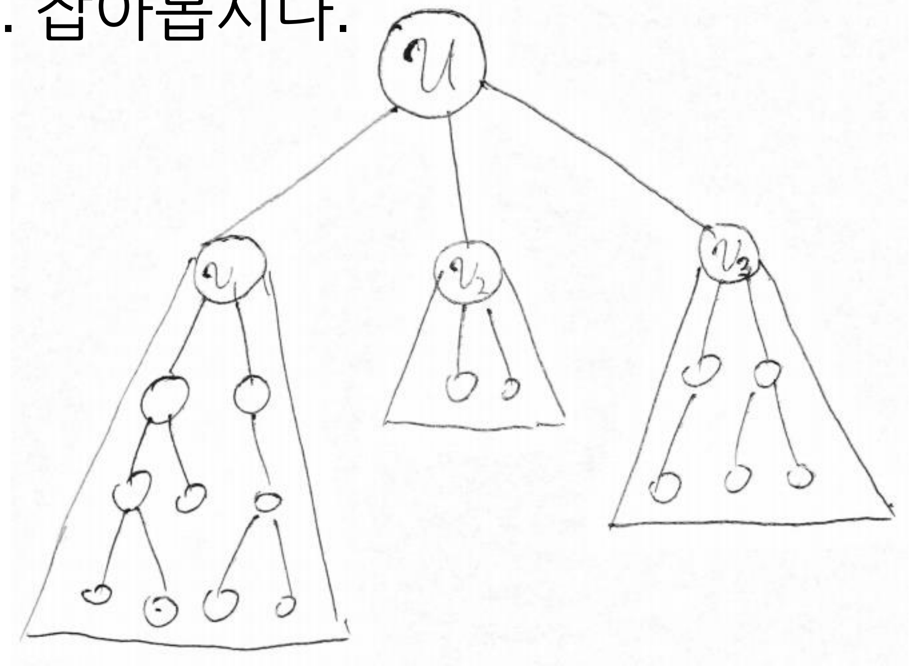
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
00:00	Yes																		1												
	No																	1													
00:30	Yes	1					1											1													
	No																														
01:00	Yes																														
	No																														1
01:30	Yes										1									1											
	No							1																							
02:00	Yes							1																							
	No															1								1							

C. 트리의 변화

		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
02:30	Yes																										1				
	No																														
03:00	Yes																														
	No																														
03:30	Yes																														
	No	2																			3										
04:00	Yes																														
	No	1								1																					1
04:30	Yes																														
	No											1																			

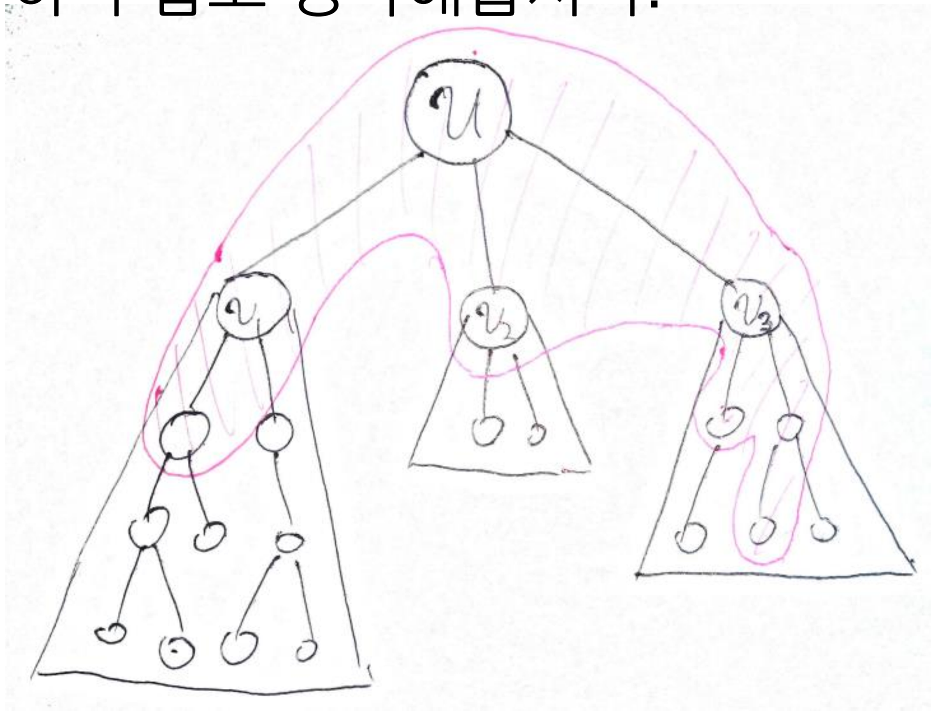
C. 트리의 변화

일반적으로 루트가 없는 트리에서의 문제를 해결할 때, 임의의 노드 u를 루트로 잡는 경향이 있습니다. 잡아봅시다.



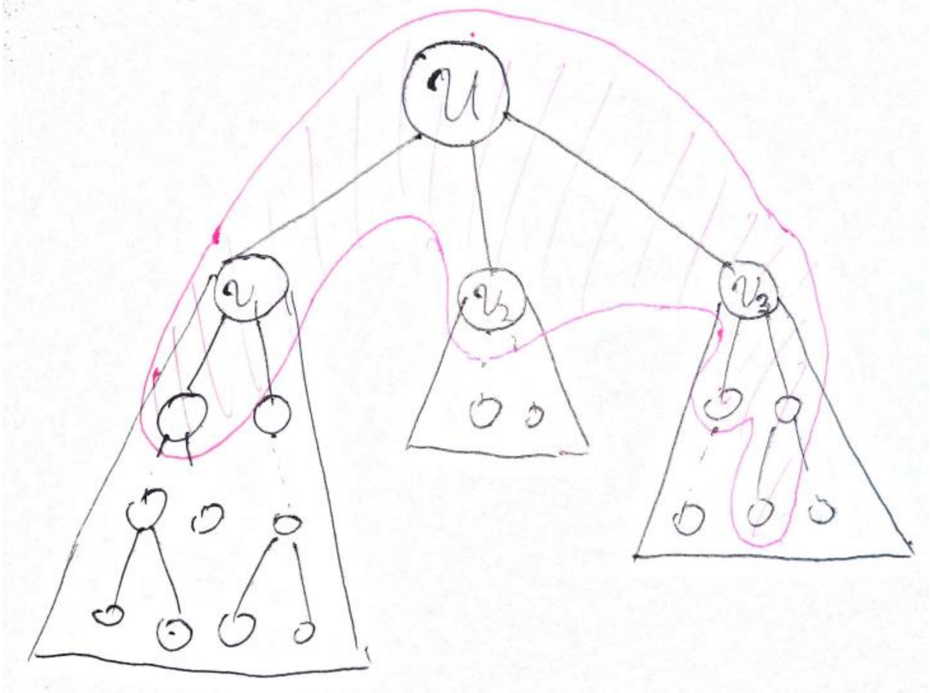
C. 트리의 변화

u를 포함하는 그룹을 하나 잡고 생각해봅시다.



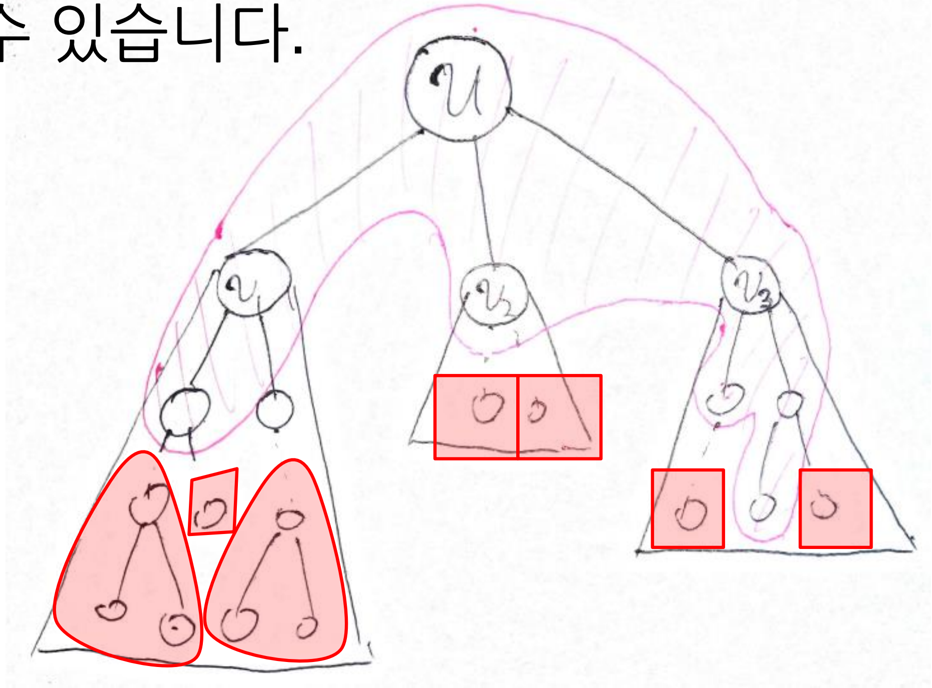
C. 트리의 변화

이 그룹이 만들어지려면, 간선 몇 개가 필연적으로 끊겨야 합니다.



C. 트리의 변화

그 결과, 몇 개의 트리가 생깁니다. 각 트리에서 문제를 해결한 후 합쳐 주면 원래 문제를 해결할 수 있습니다.



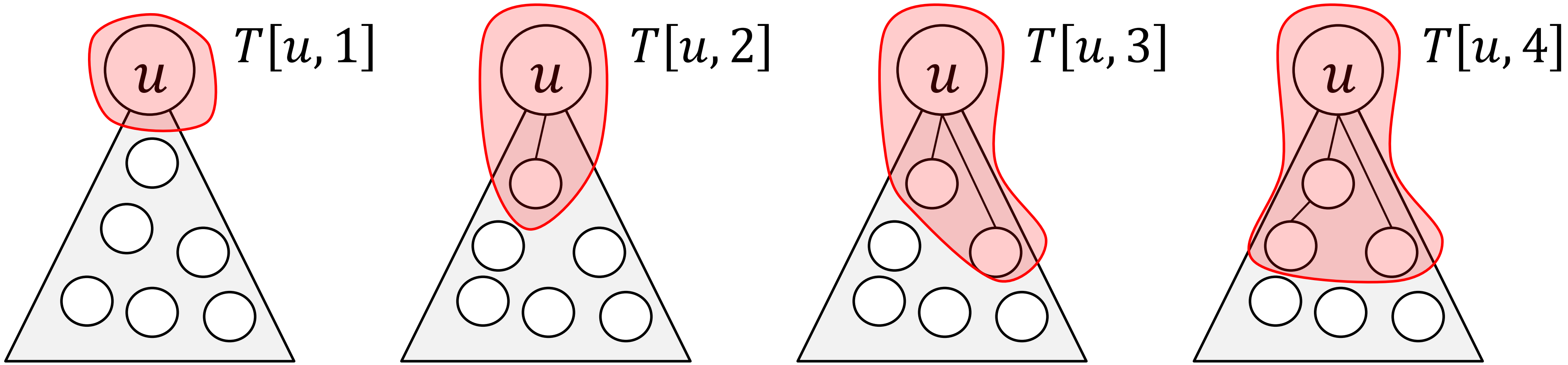
C. 트리의 변화

즉, 부분이 전체의 구조를 따라가고 있다는 것을 알 수 있습니다.

동적 계획법으로 해결할 수 있겠다는 실마리가 보입니다.

C. 트리의 변화

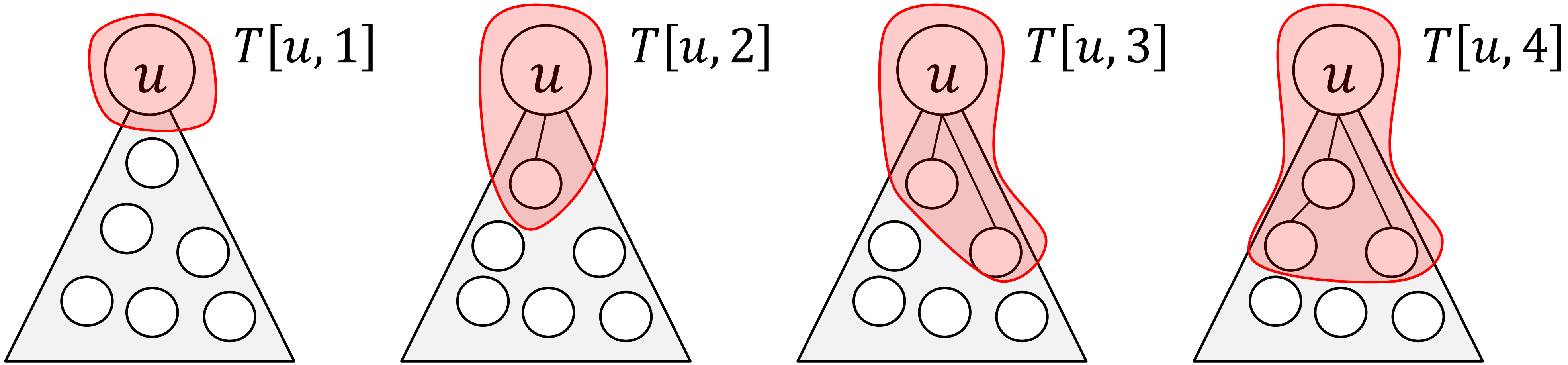
$\boldsymbol{T[u,i]}$: 루트가 u인 서브트리만 고려한 상태에서, u를 포함한 그룹의 크기가 i가 되도록 자르는 (경우의 수, 잘라야 할 간선 수)



C. 트리의 변화

$\boldsymbol{T[u,i]}$: 루트가 u인 서브트리만 고려한 상태에서, u를 포함한 그룹의 크기가 i가 되도록 자르는 (경우의 수, 잘라야 할 간선 수)

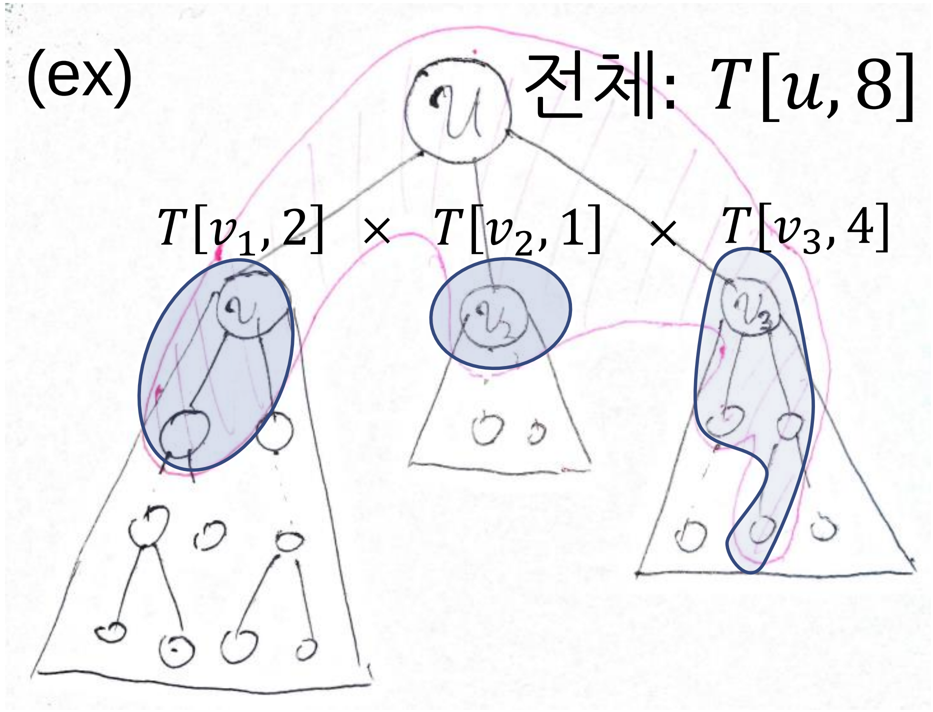
순서쌍입니다. 둘 다 저장한다는 겁니다.



C. 트리의 변화

$\boldsymbol{T[u,i]}$: 루트가 u인 서브트리만 고려한 상태에서, u를 포함한 그룹의 크기가 i가 되도록 자르는 (경우의 수, 잘라야 할 간선 수)

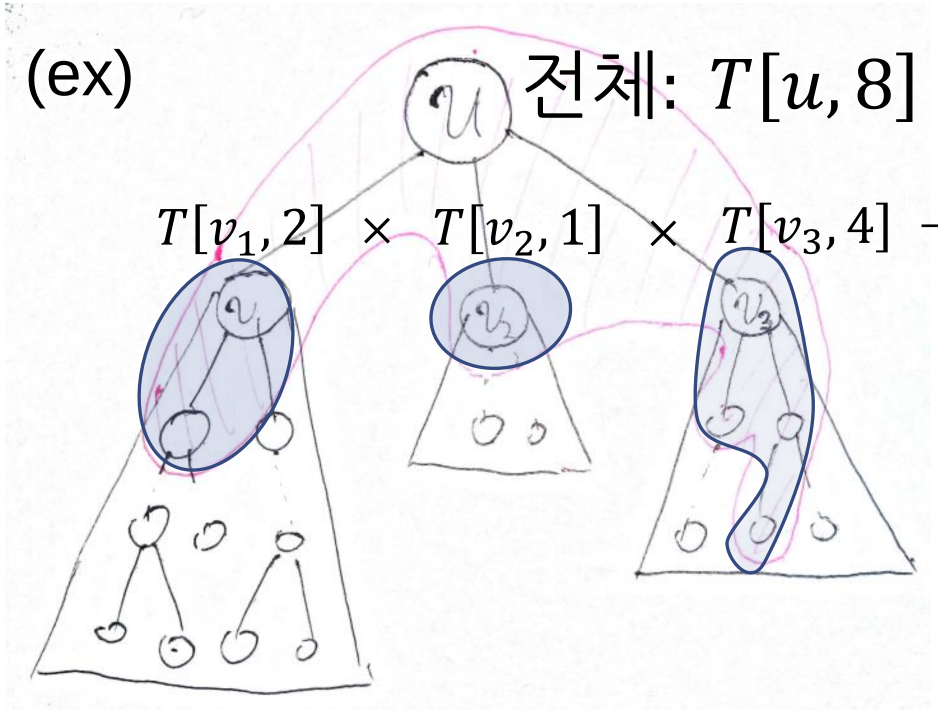
(ex)



C. 트리의 변화

$\boldsymbol{T[u,i]}$: 루트가 u인 서브트리만 고려한 상태에서, u를 포함한 그룹의 크기가 i가 되도록 자르는 (경우의 수, 잘라야 할 간선 수)

(ex)



전체: $T[u,8]$

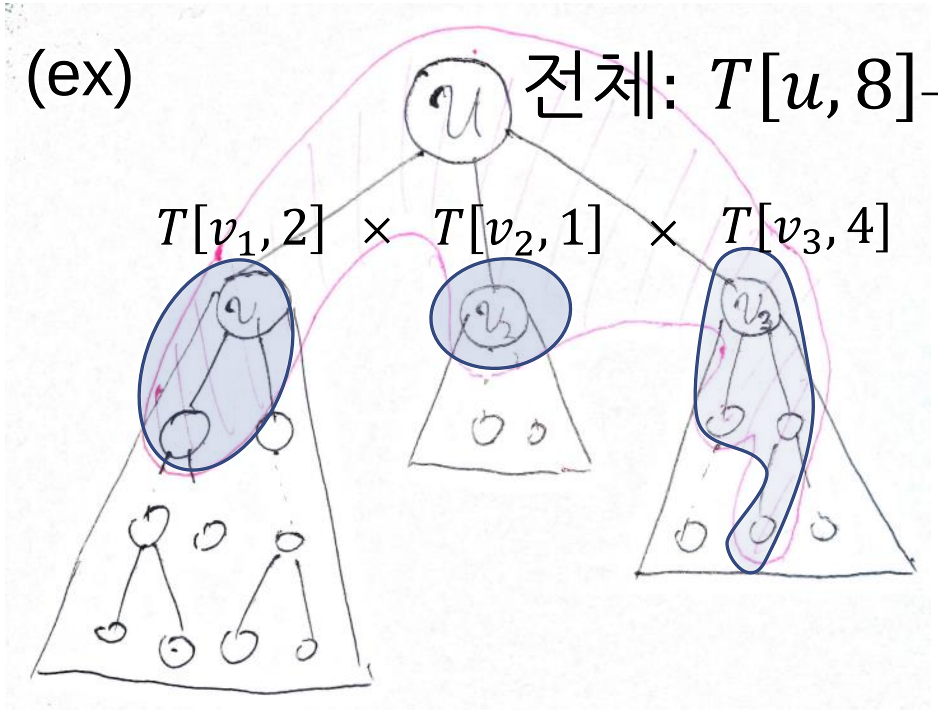
$T[v_1,2] \times T[v_2,1] \times T[v_3,4]$ → u를 포함하는 그룹을 만드는 건 '동시에' 일어나는 일!

→ <u>곱의 법칙</u>

C. 트리의 변화

$\boldsymbol{T[u,i]}$: 루트가 u인 서브트리만 고려한 상태에서, u를 포함한 그룹의 크기가 i가 되도록 자르는 (경우의 수, 잘라야 할 간선 수)

(ex)

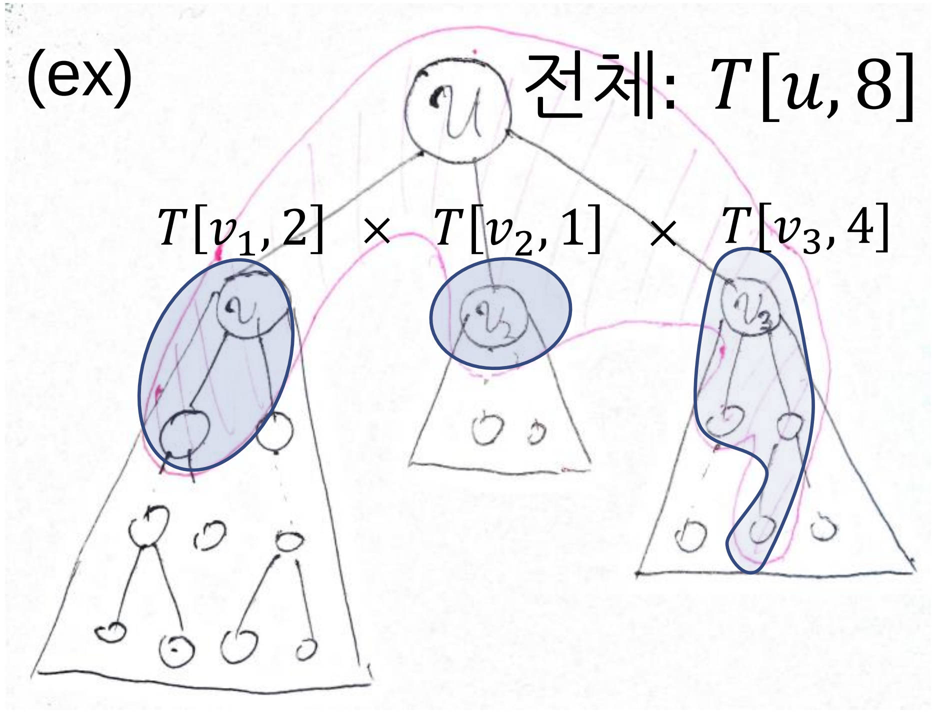


가능한 모든 배치들을 모두 고려해 봐야 할 것
→ <u>합의 법칙</u>

C. 트리의 변화

$\boldsymbol{T[u,i]}$: 루트가 u인 서브트리만 고려한 상태에서, u를 포함한 그룹의 크기가 i가 되도록 자르는 (경우의 수, 잘라야 할 간선 수)

(ex)



$$T[u,8] = \sum_{c_1+c_2+c_3=7} T[v_1,c_1]\times T[v_2,c_2]\times T[v_3,c_3]$$

각 서브트리에 묶인 그룹의 크기

서브트리에서 취합한 그룹의 크기 + 1 (노드 u) = 8

C. 트리의 변화

$\boldsymbol{T[u,i]}$: 루트가 u인 서브트리만 고려한 상태에서, u를 포함한 그룹의 크기가 i가 되도록 자르는 (경우의 수, 잘라야 할 간선 수)

$$\therefore T[u,i] = \sum_{c_1+c_2+\cdots+c_s=i-1} \prod_{j=1}^{s} T[v_j, c_j]$$

C. 트리의 변화

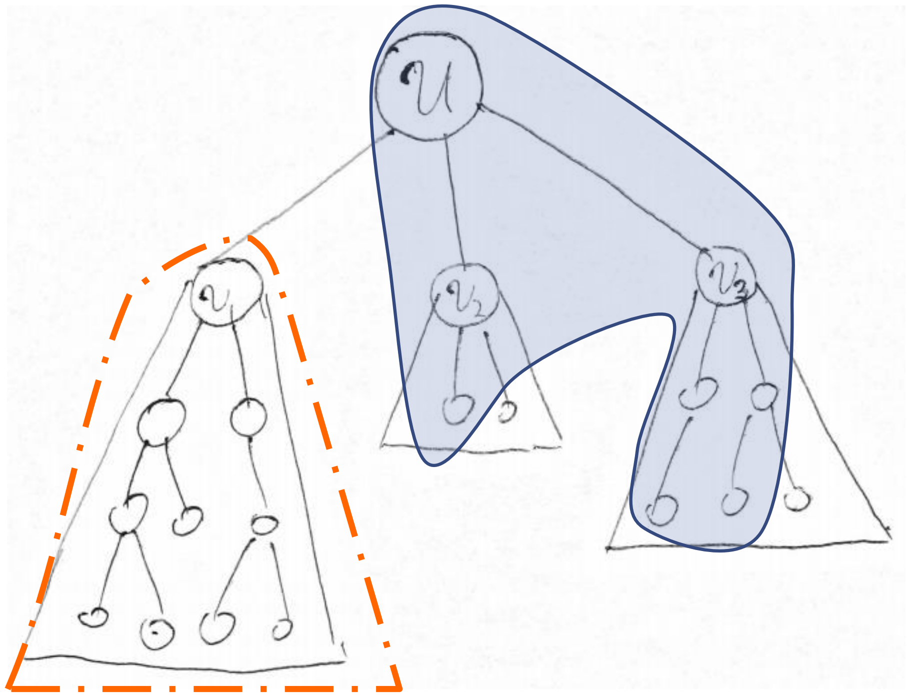
$\boldsymbol{T[u,i]}$: 루트가 u인 서브트리만 고려한 상태에서, u를 포함한 그룹의 크기가 i가 되도록 자르는 (경우의 수, 잘라야 할 간선 수)

$$\therefore T[u,i] = \sum_{c_1+c_2+\cdots+c_s=i-1} \prod_{j=1}^{s} T[v_j, c_j]$$

그럴듯해 보이지만, 아직 해결하지 않은 문제점이 몇 가지 있습니다. 각각을 해결해보도록 하겠습니다.

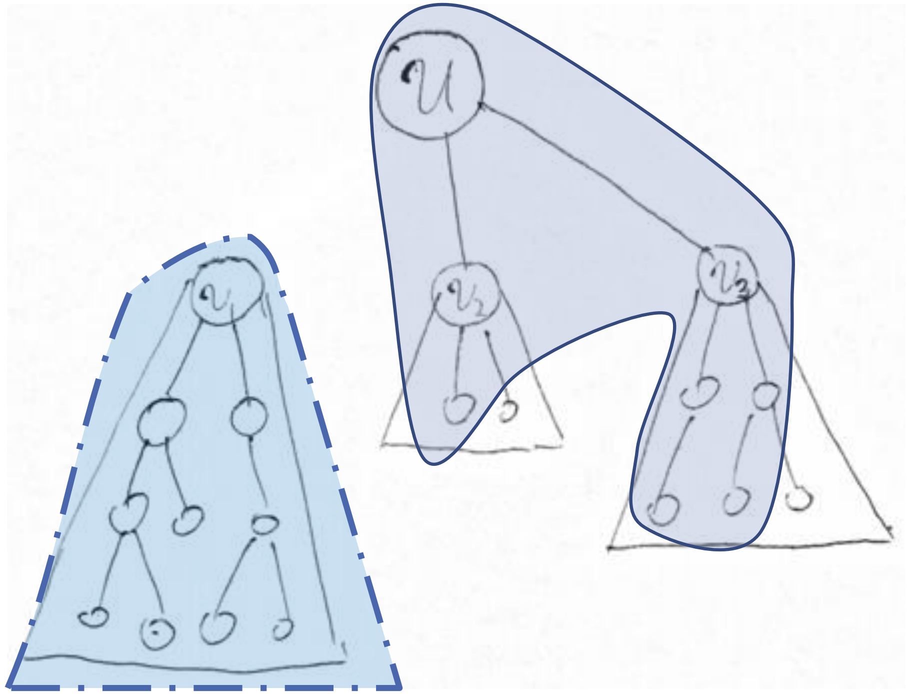
C. 트리의 변화

1. 특정 서브트리를 포함하지 않는 상태로 그룹을 만들면 어떻게 하는가?



C. 트리의 변화

1. 특정 서브트리를 포함하지 않는 상태로 그룹을 만들면 어떻게 하는가?

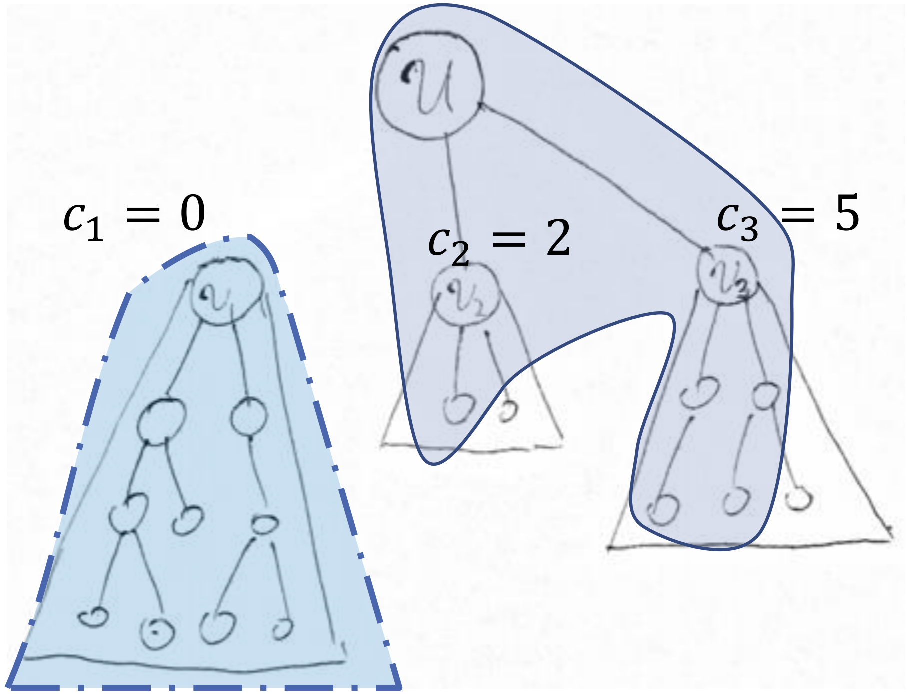


두 그룹을 나눠서 각각 문제를 해결한 후,
경우의 수를 곱해주면 됩니다.

하지만 이는 너무 번거로운 것 같습니다.

C. 트리의 변화

1. 특정 서브트리를 포함하지 않는 상태로 그룹을 만들면 어떻게 하는가?



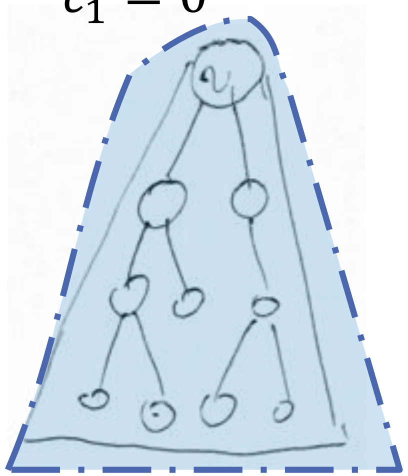
그래서 이러한 경우에는 $\underline{c_i = 0}$으로 둡니다.
예외 없이 생각하려면,
c_i를 **"v_i 그룹이 u 그룹에 기여하는 노드 수"**로 생각해도 좋습니다.

그러면 여전히 $i = c_1 + c_2 + \cdots + c_s + 1$이 성립하여, 점화식을 그대로 사용할 수 있습니다.

C. 트리의 변화

1. 특정 서브트리를 포함하지 않는 상태로 그룹을 만들면 어떻게 하는가?

$c_1 = 0$



그러면 새로 $T[u, 0]$을 정의할 수 있습니다.
노드 u가 루트인 서브트리에서 문제를 해결했을 때의 답 (즉, 최종 답)을 저장해두면 되겠죠.

$$\therefore T[u, 0] = T[u, 1] + T[u, 2] + \cdots + T[u, 2^k] + \cdots$$

u가 속한 그룹의 크기가 2의 거듭제곱 꼴인 경우를 모두 고려하여 합친 것입니다. 이렇게 문제가 해결되었습니다.

C. 트리의 변화

2. 순서쌍을 어떻게 더하고 곱하는가?

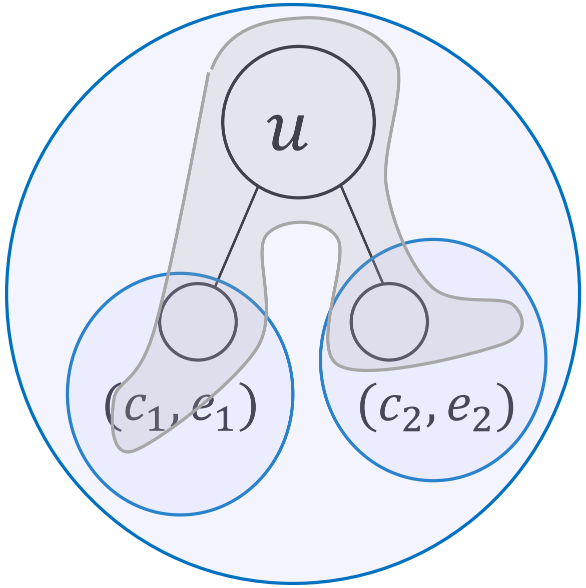
분명히 $T[u,i]$의 정의에는 순서쌍이라고 강조해 놓았는데, 그저 경우의 수를 더하고 곱하듯이 + 기호와 × 기호를 쓰고 있습니다. 이는 생각의 편의를 위해 그렇게 표기한 것입니다.

(경우의 수, 자르는 간선의 수) 순서쌍 (c,e) 두 개를 ‘곱하고’ ‘더하는’ 것에 대해 생각해 봅시다.

C. 트리의 변화

2. 순서쌍을 어떻게 더하고 곱하는가?

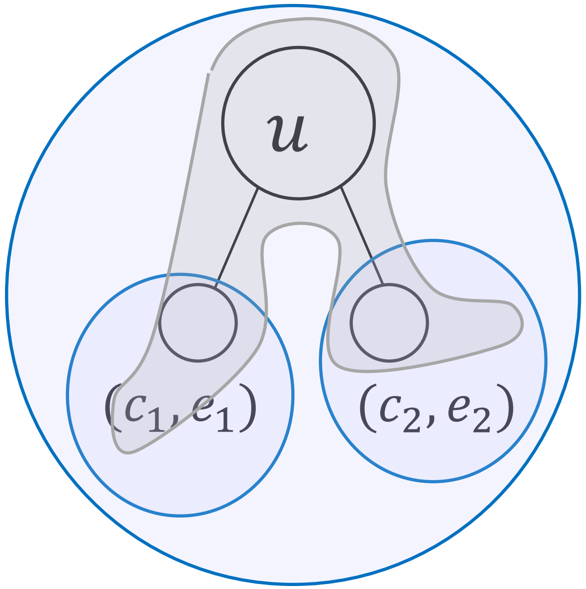
먼저 '곱하는' 경우입니다. 앞에서 언급했듯이, 곱하는 것은 u를 포함하는 그룹을 만들 때 일어납니다. 여러 그룹을 '동시에' 하나로 합친다는 겁니다.



C. 트리의 변화

2. 순서쌍을 어떻게 더하고 곱하는가?

따라서 경우의 수는 곱의 법칙에 의해 곱해지고, 간선 수는 둘 다 원래의 배치 상태를 유지하면서 합쳐지므로 더해집니다. 직관적으로 생각하시면 됩니다.

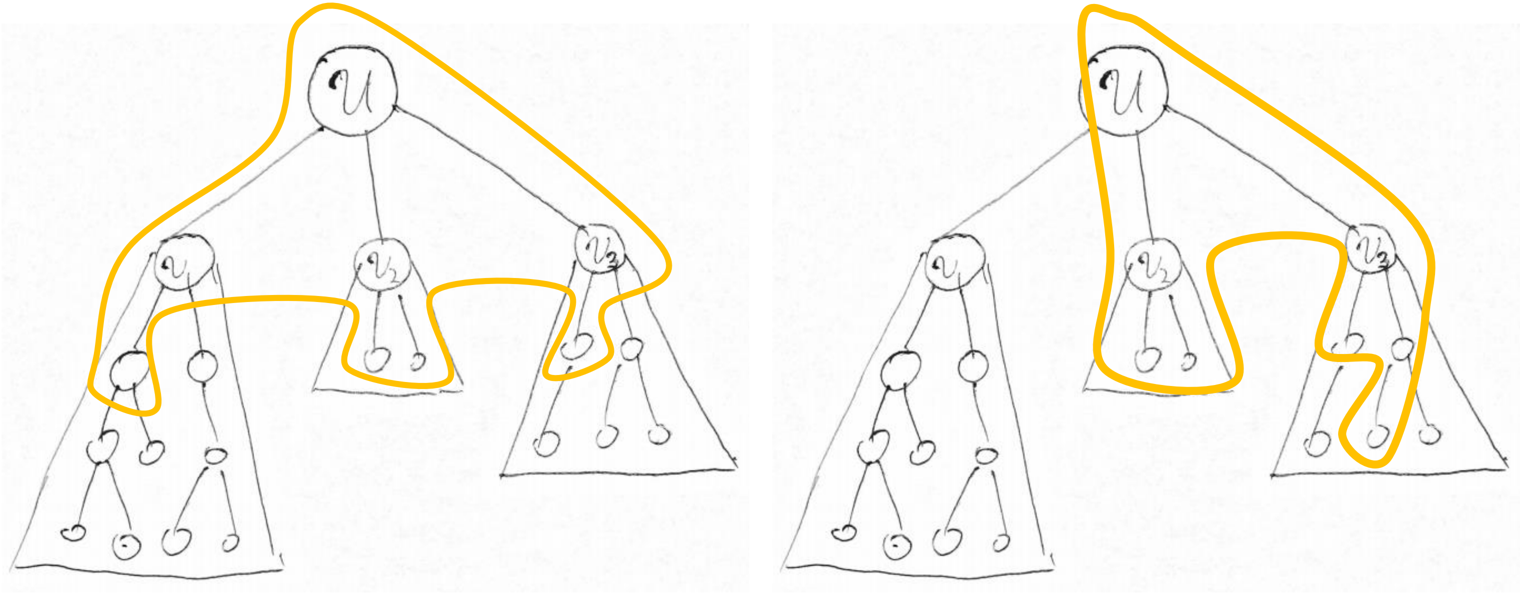


$$(c_1, e_1) \times (c_2, e_2)$$
$$= (c_1 \times c_2, e_1 + e_2)$$

C. 트리의 변화

2. 순서쌍을 어떻게 더하고 곱하는가?

두 번째로 더하는 경우를 생각해봅니다. '더한다'는 것은 가능한 모든 배치(u를 포함하는 그룹)들을 다 고려한다는 겁니다.



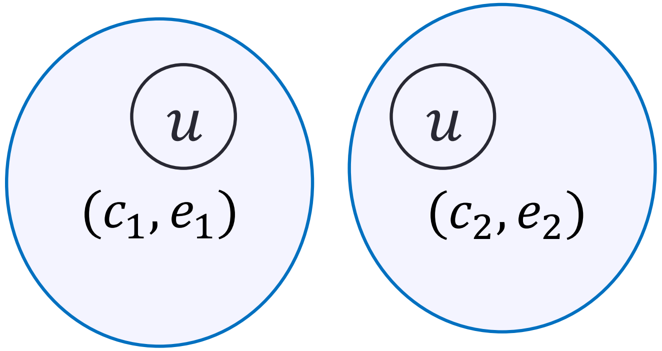
....

C. 트리의 변화

2. 순서쌍을 어떻게 더하고 곱하는가?

우리의 목적은 ‘자르는 간선 수를 최소화’하는 경우의 수를 세는 겁니다.

즉, 두 순서쌍 $(c_1, e_1), (c_2, e_2)$가 있을 때 $e_1 \neq e_2$라면 둘 중 작은 것만 세 주어야 한다는 것입니다. $e_1 = e_2$면 경우의 수도 더할 수 있습니다. 정리하면,



$$(c_1, e_1) + (c_2, e_2)$$
$$= \begin{cases} (c_1 + c_2, e_1) & e_1 = e_2 \\ (c_1, e_1) & e_1 < e_2 \\ (c_2, e_2) & e_1 > e_2 \end{cases}$$

C. 트리의 변화

2. 순서쌍을 어떻게 더하고 곱하는가?

순서쌍의 덧셈과 뺄셈을 정리하면 아래와 같습니다. 순서쌍을 클래스나 구조체에 저장해 두고 연산자 오버로딩 같은 걸 하면 보기에도 편하고 구현도 간단합니다.

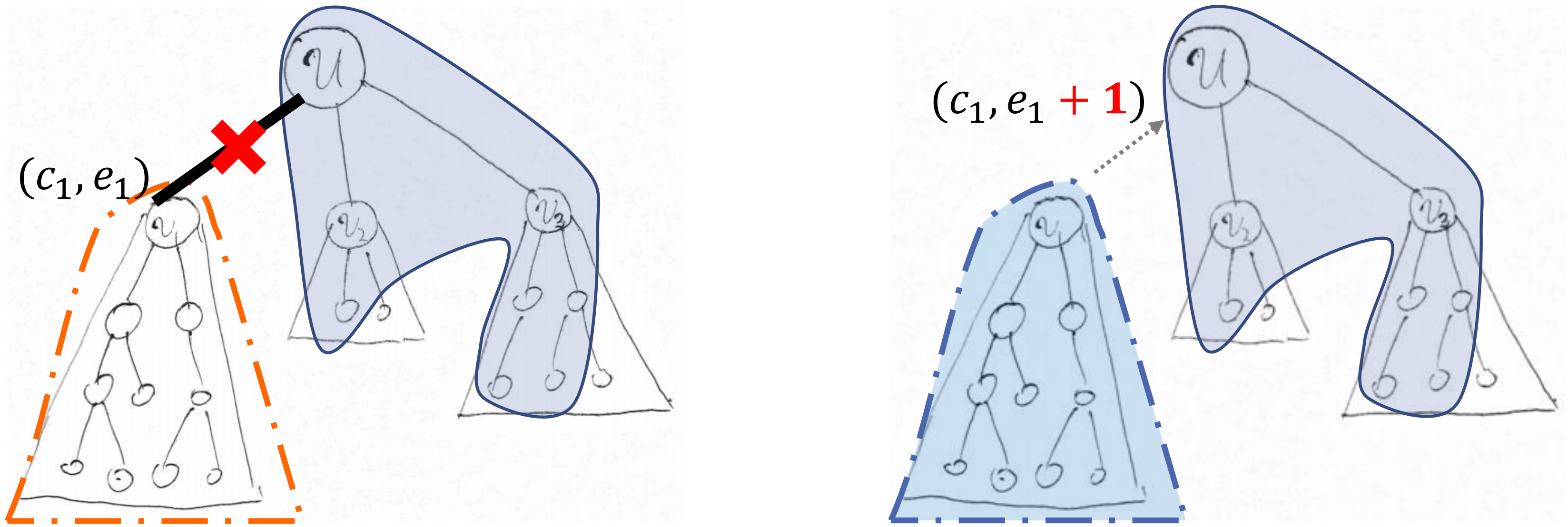
$$(c_1, e_1) \times (c_2, e_2) \\ = (c_1 \times c_2, e_1 + e_2)$$

$$(c_1, e_1) + (c_2, e_2) \\ = \begin{cases} (c_1 + c_2, e_1) & e_1 = e_2 \\ (c_1, e_1) & e_1 < e_2 \\ (c_2, e_2) & e_1 > e_2 \end{cases}$$

C. 트리의 변화

2. 순서쌍을 어떻게 더하고 곱하는가?

잠시 $T[u,0]$ 이야기로 돌아갑니다. $T[u,i]$를 구할 때 $T[v_i,0]$를 참조한다면 <u>반드시</u> 간선 하나가 끊기게 되므로, e의 값을 1 증가시켜야 합니다.



C. 트리의 변화

3. 그래서 이 식을 어떻게 계산하는가?

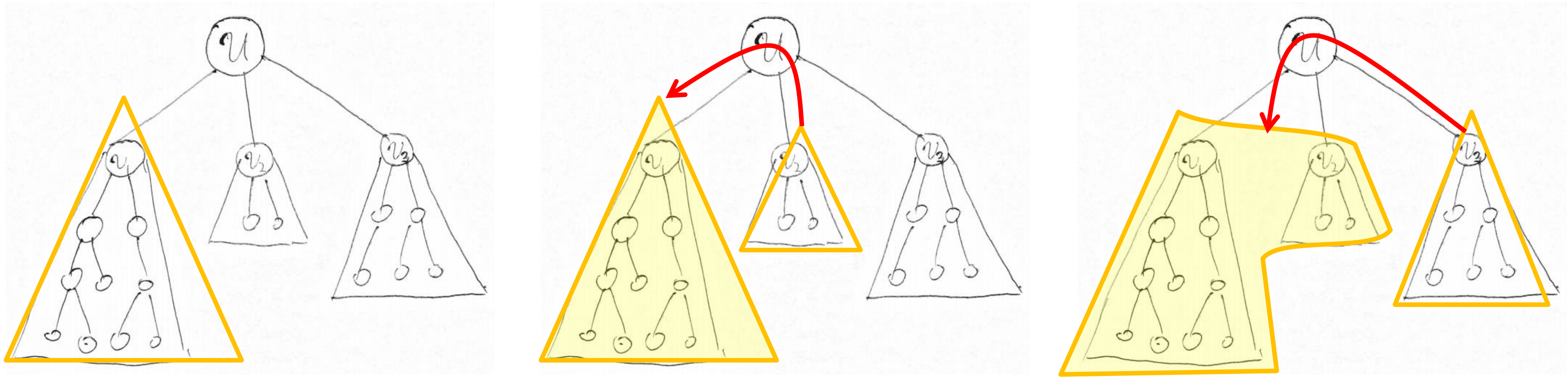
$$\therefore T[u,i] = \sum_{c_1+c_2+\cdots+c_s=i-1} \prod_{j=1}^{s} T[v_j, c_j]$$

이제 남은 것은 저 식을 계산하는 일입니다. 이런 식의 꼴은 어디서 많이 본 것 같습니다. 일종의 0/1 배낭 문제라고도 생각할 수 있습니다.

C. 트리의 변화

3. 그래서 이 식을 어떻게 계산하는가?

식이 복잡해 보이지만 두 개의 서브트리의 결과만 합칠 수 있으면 족합니다. 서브트리가 여러 개 있더라도 아래와 같은 과정으로 전체를 합칠 수 있기 때문입니다.



C. 트리의 변화

3. 그래서 이 식을 어떻게 계산하는가?

즉,

$$T[u,i] = \sum_{c_1+c_2=i-1} \prod_{j=1}^{s} T[v_j, c_j]$$

c_1, c_2를 보기 좋게 k로 바꿔서 표기하면

$$T[u,i] = \sum_{0 \le k \le i-1} T[v_1, k] \times T[v_2, i-1-k]$$

C. 트리의 변화

3. 그래서 이 식을 어떻게 계산하는가?

이 식을 계산하려면 적어도 $O(i)$만큼의 시간이 필요합니다.

$1 \leq i \leq n$ (서브트리의 크기가 최대 n)이므로 $T[u,*]$를 계산하는 데에는 $O(n^2)$의 시간이 필요하고, 노드는 총 n개 존재하므로, $O(n^3)$의 시간복잡도로 문제를 해결할 수 있는 것 같습니다.

느립니다.

C. 트리의 변화

3. 그래서 이 식을 어떻게 계산하는가?

이 식을 계산하려면 적어도 $O(i)$만큼의 시간이 필요합니다.

$1 \le i \le n$ (서브트리의 크기가 최대 n)이므로 $T[u,*]$를 계산하는 데에는 $O(n^2)$의 시간이 필요하고, 노드는 총 n개 존재하므로, $O(n^3)$의 시간복잡도로 문제를 해결할 수 있는 것 같습니다.

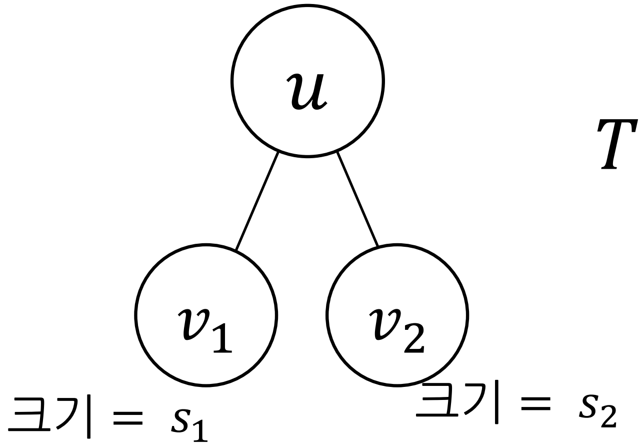
느립니다.

하지만 놀랍게도 이 방법을 '잘' 사용하면 $O(n^2)$의 시간복잡도가 보장됩니다.
어떻게 할까요?

C. 트리의 변화

3. 그래서 이 식을 어떻게 계산하는가?

우선 '막' 계산하는 방법부터 살피겠습니다. 노드 2개만 있는 경우를 생각해 봅니다. $1 \le i \le s_1 + s_2$이므로, $T[u]$를 계산하는 데에 $O((s_1 + s_2)^2)$이 드는 것처럼 보입니다.

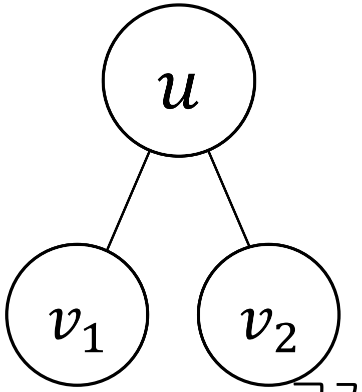


$$T[u,i] = \sum_{0 \le k \le i-1} T[v_1,k] \times T[v_2, i-1-k]$$

C. 트리의 변화

3. 그래서 이 식을 어떻게 계산하는가?

하지만 간과하고 있는 부분이 있는데, $T[v_1, k]$는 $1 \le k \le s_1$일 때만,
$T[v_2, i-1-k]$는 $1 \le i-1-k \le s_2$일 때만 의미를 가진다는 것입니다.
(서브트리 크기가 s_2인데 크기가 s_2+1인 그룹을 만들 수는 없습니다.)



크기 = s_1

크기 = s_2

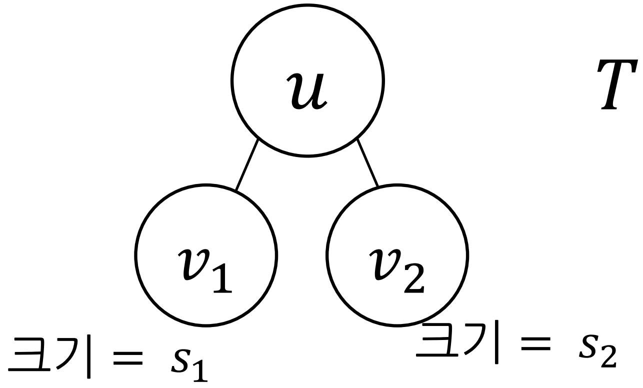
$$T[u,i] = \sum_{\substack{\cancel{0 \le k \le i-1} \\ \max(\mathbf{0}, \boldsymbol{i} - \boldsymbol{s_2} - \mathbf{1}) \le \boldsymbol{k} \le \min(\boldsymbol{i} - \mathbf{1}, \boldsymbol{s_1})}} T[v_1, k] \times \boxed{T[v_2, i-1-k]}$$

$$1 \le i-1-k \le s_2$$
$$-s_2 \le k+1-i \le -1$$
$$i - s_2 - 1 \le k \le i-2$$

C. 트리의 변화

3. 그래서 이 식을 어떻게 계산하는가?

이렇게 k의 범위를 바꿔주면 $O(s_1 s_2)$의 시간복잡도로 $T[u,*]$를 구할 수 있습니다. k의 범위를 가지고 계산해도 되지만, $T[v_1, p]$ $(0 \le p \le s_1)$와 $T[v_2, q]$ $(0 \le q \le s_2)$를 묶쳐서 $T[u, p+q]$에 반영한다고 생각하는 것이 증명과 구현에 있어 상당히 편리합니다.



$$T[u,i] = \sum_{\substack{\cancel{0 \le k \le i-1} \\ \mathbf{\max(0,\, i - s_2 - 1) \le k \le \min(i-1,\, s_1)}}} T[v_1, k] \times \boxed{T[v_2, i-1-k]}$$

$$1 \le i - 1 - k \le s_2$$
$$-s_2 \le k + 1 - i \le -1$$
$$i - s_2 - 1 \le k \le i - 2$$

C. 트리의 변화

3. 그래서 이 식을 어떻게 계산하는가?

그래서 크기가 n일 때 문제를 해결하는 데 드는 시간을 $T(n)$으로 두면,

$$T(n) = T(s_1) + T(s_2) + O(s_1 s_2)\ (s_1 + s_2 = n)$$

의 식을 세울 수 있습니다. 이 식을 잘 풀어보면 $T(n) = O(n^2)$임을 알 수 있습니다.

C. 트리의 변화

3. 그래서 이 식을 어떻게 계산하는가?

옳은 방법은 아니지만, 대강이나마 그 이유를 파악해 보겠습니다. 문제를 좀 바꿔서

$$f(x+y)=f(x)+f(y)+xy, f(1)=1$$

일 때 $f(x)$의 식을 알아봅시다.

$y=1$을 대입하면,

$$f(x+1)=f(x)+f(1)+x=f(x)+x+1$$

입니다. 즉

$$f(x+1)-f(x)=x+1$$

이므로,

C. 트리의 변화

3. 그래서 이 식을 어떻게 계산하는가?

$$f(x) = f(1) + \sum_{i=1}^{x-1} \{f(i+1) - f(i)\}$$

$$= f(1) + \sum_{i=1}^{x-1} (i+1) = f(1) + \frac{1}{2}(x^2 + x - 2)$$

$$= \frac{1}{2}(x^2 + x)$$

입니다. 이 결과를 원래 식에 대입해보면 일치하므로, $f(x) = O(n^2)$라는 사실을 알게 되었습니다.

C. 트리의 변화

3. 그래서 이 식을 어떻게 계산하는가?

하지만 이 설명은 주먹구구식 설명이라서 만족하지 않으신 분들이 계실 거라 생각합니다.

http://koosaga.myungwoo.kr/67 에 관련된 언급이 있으니 한 번 읽어 보시고, 모르는 점이 있다면 저 블로그에 질문해주세요(?)

C. 트리의 변화

이제 모든 것이 마무리되었습니다.

$T[root, 0]$의 값을 구하여 경우의 수를 출력하면 됩니다.

D. 이진 탐색

- 맞은 팀 수 : 10
- 제출 횟수 : 100
- 정답률: 10%
- 처음 맞은 팀: Anti-Q (ch_49, tonyjjw, dotorya, qo) , 42분
- 출제자 : cubelover (윤지학)
- 해설 작성자: cubelover (윤지학)

D. 이진 탐색

		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
00:00	Yes																														
	No																														
00:30	Yes													1																	
	No			1																											
01:00	Yes																														
	No										1					1	1					1				2					1
01:30	Yes		1			1																									
	No		1	1								1	1		1	2	1										1				
02:00	Yes								1																	1					
	No												1	2	1		1	2		1		1					1			2	1

D. 이진 탐색

		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
02:30	Yes			1																											
	No	1																					1				1				
03:00	Yes																	1													
	No			1			2			1			1				1					1			1						
03:30	Yes		1																												
	No														1						3	2						1			
04:00	Yes																														
	No					1				1		1				1	1	2	1					1	2			1	2	3	1
04:30	Yes				1									1																	
	No				1	2			1		1				1			2		2	1		2	1	1	2		2	2	1	2

D. 이진 탐색

풀이부터 이야기하면,

$$\sum_{i=1}^{n} 2^{-a_i} \leq 1$$

이면 가능하고, 이 때 첫 번째로 물어볼 수 있는 정수들은

$$\sum_{i=1}^{k} 2^{-a_i} \leq \frac{1}{2}, \sum_{i=k+1}^{n} 2^{-a_i} \leq \frac{1}{2}$$

인 모든 k $(1 \leq k \leq n)$들입니다.

단 $\sum_{i=1}^{n} 2^{-a_i} \leq \frac{1}{2}$ 이면 “inf”를 출력해야 합니다.

D. 이진 탐색

이는 수학적 귀납법으로 증명할 수 있습니다.

먼저 $\sum_{i-1}^{n} 2^{-a_i} \leq 1$ 이면 그러한 k가 존재한다는 것을 전제로 합니다.
이는 뒤에서 증명할 것입니다.

D. 이진 탐색

그럼 맨 처음 “x가 k 이하입니까?”라고 질문을 하면,

- $x \le k$
  - 이제 x의 범위는 $1 \le x \le k$로 변했고, 질문 횟수 제한은 $a_1 - 1, a_2 - 1, \cdots, a_k - 1$이 됩니다. 전제에서 $\sum_{i=1}^{k} 2^{-a_i} \le 2^{-1}$이라 했으므로 $\sum_{i=1}^{k} 2^{-(a_i - 1)} \le 1$이 됩니다.
- $x > k$
  - 이제 x의 범위는 $k + 1 \le x \le n$로 변했고, 질문 횟수 제한은 $a_{k+1} - 1, a_{k+2} - 1, \cdots, a_n - 1$이 됩니다. 전제에서 $\sum_{i=k+1}^{n} 2^{-a_i} \le 2^{-1}$이라 했으므로 $\sum_{i=k+1}^{n} 2^{-(a_i - 1)} \le 1$이 됩니다.

따라서 $\sum_{i=1}^{n} 2^{-a_i} \le 1$ 이면 이 전략을 사용함으로써 문제를 해결할 수 있다는 걸 알 수 있습니다.

D. 이진 탐색

그러면 $\sum_{i=1}^{n} 2^{-a_i} > 1$ 인 경우에는 왜 안 될까요?

올바른 전략이 존재한다고 가정하고, 첫 번째로 k를 물어본다고 가정해 봅시다. $\sum_{i=1}^{k} 2^{-a_i}$와 $\sum_{i=k+1}^{n} 2^{-a_i}$ 중 하나는 $\frac{1}{2}$보다 클 것입니다.

그럼 해당 구간을 계속 취해서 파고 들어가면, 어느 순간 구간의 길이가 1 (즉 x가 결정됨)임에도 불구하고 $2^{-a_x} > 1$, 즉 $a_x < 0$이 되는데 이는 주어진 횟수 이내로 x를 찾지 못했다는 것을 의미합니다.

따라서 불가능합니다.

D. 이진 탐색

이제 $\sum_{i=1}^{n} 2^{-a_i} \le 1$이면 $\sum_{i=1}^{k} 2^{-a_i} \le \frac{1}{2}$, $\sum_{i=k+1}^{n} 2^{-a_i} \le \frac{1}{2}$을 모두 만족하는 정수 k가 반드시 존재함을 증명합니다.

$s_i = \sum_{k=1}^{i} 2^{-a_k}$($2^{-a_i}$의 누적합)라고 합시다. 그리고 s_i가 $\frac{1}{2}$ 초과인 최소의 i를 j라 합시다. 이러한 j가 존재하지 않는다면 "inf"를 출력하면 되므로 논의하지 않습니다. 이제 $s_{j-1} = \frac{1}{2}$임을 보이고자 합니다.

$s_j - s_{j-1} = 2^{-a_j}$이고 $\{a_i\}$는 단조증가하므로, $2^{-a_1}, 2^{-a_2}, \cdots, 2^{-a_j}$ 모두 2^{-a_j}의 정수배입니다. 그러면 1 이상 j 이하의 i에 대해서 $s_i = x_i \times 2^{-a_j}$라고 쓸 수 있겠죠 ($x_i$는 정수)

D. 이진 탐색

$s_j > \frac{1}{2}$이므로 $x_j > 2^{a_j-1}$이고, $s_{j-1} \le \frac{1}{2}$이므로 $x_{j-1} \le 2^{a_j-1}$입니다.

그런데 x_j는 정수이므로 $x_j = 2^{a_j-1} + 1$입니다.

즉 $x_j - 1 = x_{j-1} = 2^{a_j-1}$이고, 따라서 $s_{j-1} = \frac{1}{2}$를 만족합니다.

그러므로 k는 무조건 존재합니다.

E. 별빛이 내린다

- 맞은 팀 수 : 3
- 제출 횟수 : 42
- 정답률: 7.143%
- 처음 맞은 팀: Anti-Q (ch_49, tonyjjw, dotorya, qo) , 282분
- 출제자 : tae (류현종)
- 해설 작성자 : tncks0121 (박수찬)

E. 별빛이 내린다

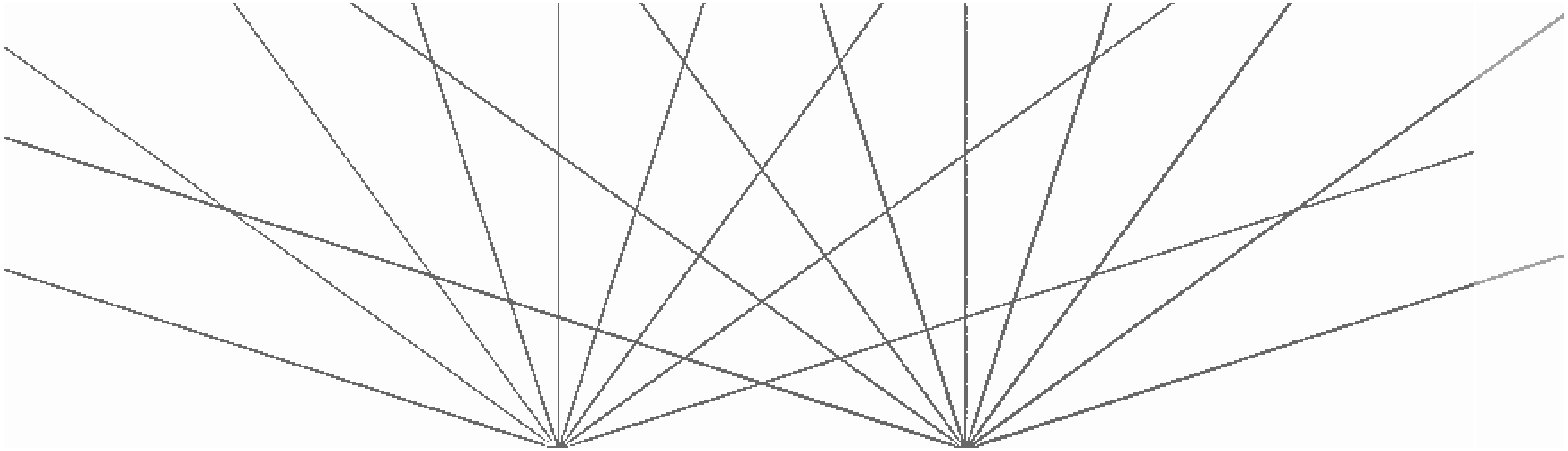
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
00:00	Yes																														
	No																														
00:30	Yes																														
	No																														
01:00	Yes																														
	No																														
01:30	Yes																														
	No										1																				
02:00	Yes																														
	No									1																					

E. 별빛이 내린다

		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
02:30	Yes																														
	No																		1			1									
03:00	Yes																														
	No											1																			
03:30	Yes																														
	No																						4	1							
04:00	Yes																														
	No	1																									1				
04:30	Yes													1														1	1		
	No														1			1			1				1	1		3			

E. 별빛이 내린다

문제를 요약하면, 대강 아래 그래프 위에 있는 몇 개의 선분이 주어졌을 때, 교점의 개수를 구하라는 것입니다.



E. 별빛이 내린다

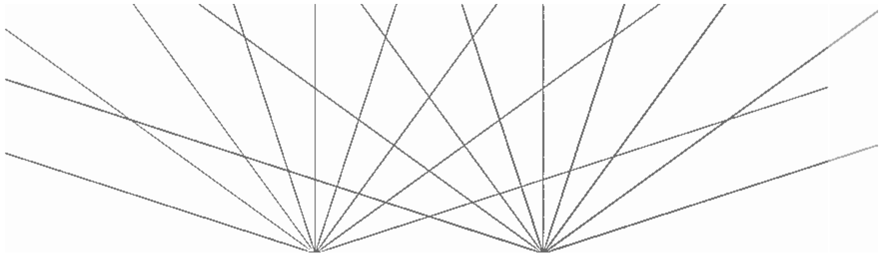
교점의 개수가 되는 이유는,

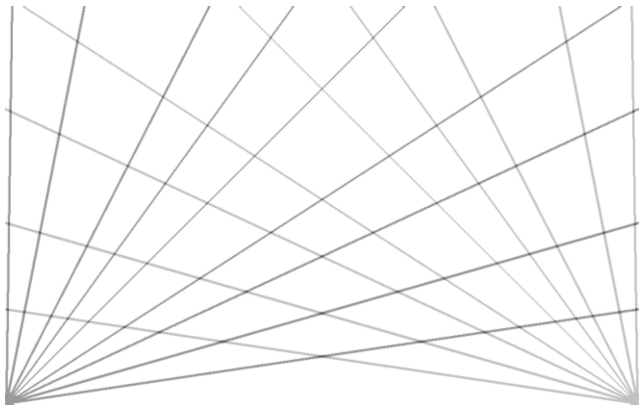
- “모든 관측”이 입력으로 주어지고,
- 각 관측자가 별을 바라본 각이 0° 초과 180° 미만이므로, 두 관측자가 한 관측을 하나씩 잡을 때 교선이 생기는 경우가 없으며,
- 각 관측자가 관측한 결과끼리는 서로 겹치지 않고,
- 같은 좌표에 별은 최대 한 개 존재하며,
- ‘별이 있을 수도 있고 없을 수도 있는 위치에는 별이 떠 있다고 가정’

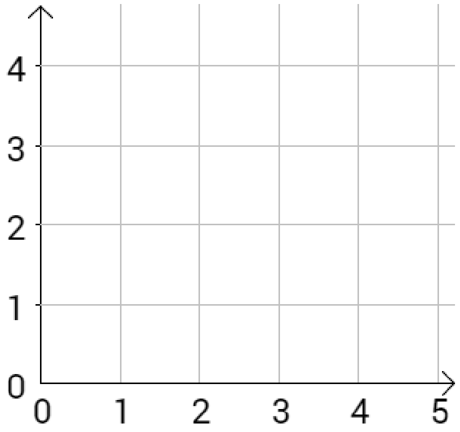
하기 때문입니다.

E. 별빛이 내린다

이 특징을 잘 활용해서, 좌표계를 잘 조작하면 결국 직교 좌표계 비슷하게 됩니다. 코더스의 위치를 살짝 바꿔서 코더스의 관측을 모두 평행하게 하고, 비슷하게 하이의 관측도 모두 평행하게 만들면 됩니다.

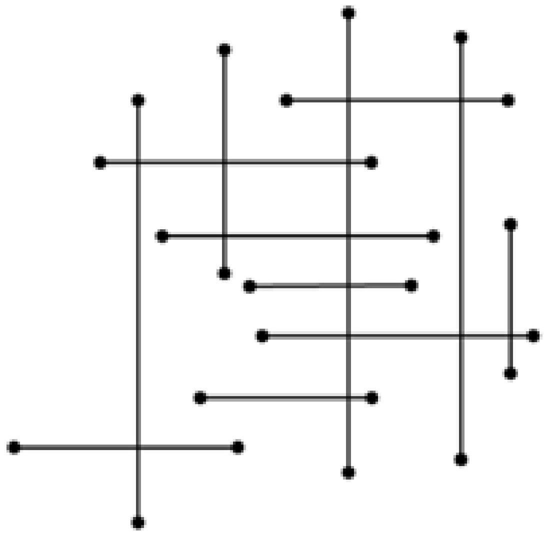






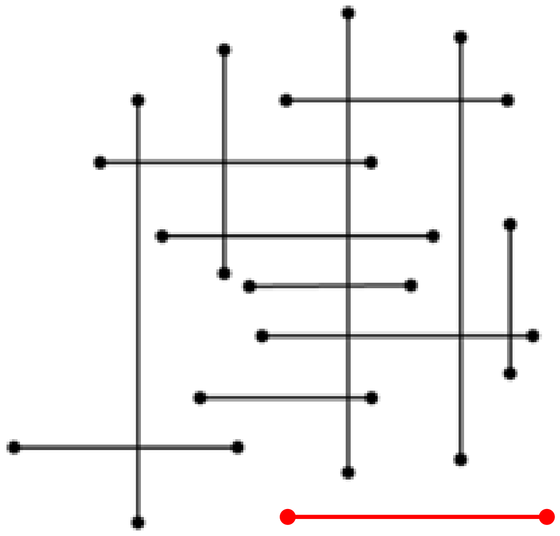
E. 별빛이 내린다

좌표계를 바꿈으로써, 각 관측을 x축/y축에 평행한 선분들로 바꿔버릴 수 있고, 이러한 상황에서 교점의 개수를 구하는 문제가 됩니다. 이는 Segment Tree나 Fenwick Tree와 같은 자료구조를 활용하여 구할 수 있습니다.



E. 별빛이 내린다

이제 남은 것은 '불가능한 관측 결과'를 판정하는 것입니다. 불가능한 경우는, 다른 선분과 교차하지 않는 어떤 선분이 존재하는 경우입니다. 이는 교점의 개수를 구하는 과정에서 처리할 수 있습니다. (결국 총 교점의 개수 = 각 선분의 교점의 개수의 합이기 때문입니다.)



F. 계산 실수

- 맞은 팀 수 : 3
- 제출 횟수 : 31
- 정답률: 9.677%
- 처음 맞은 팀: Andromeda Express (Feat. kriii) (ch_46, Astein, ainu7, kriii) , 114분
- 출제자: tncks0121 (박수찬)
- 해설 작성자 : tncks0121 (박수찬)

F. 계산 실수

		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
00:00	Yes																														
	No																														
00:30	Yes																														
	No																														
01:00	Yes																														
	No															1															
01:30	Yes																									1					
	No								1			1	1								1										
02:00	Yes																														
	No														1					1		1									

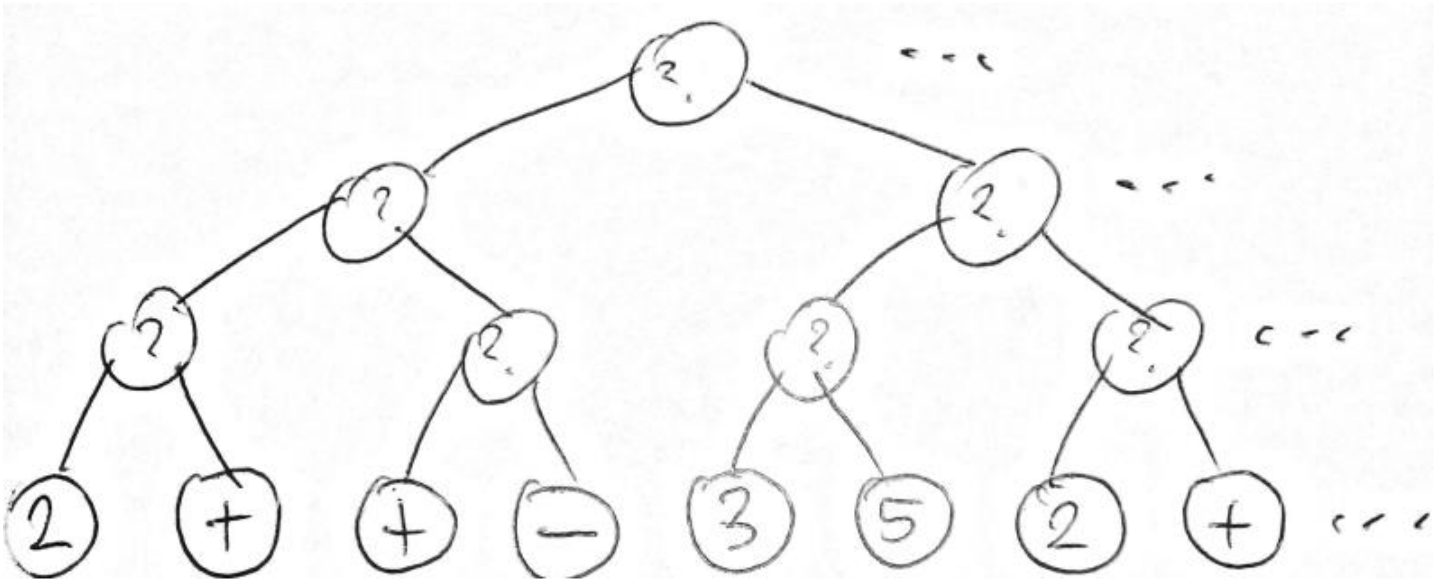
F. 계산 실수

		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
02:30	Yes																														
	No					1	1							1																	
03:00	Yes																														
	No																						1								
03:30	Yes																											1			
	No																									4	2				
04:00	Yes																														
	No																			1											
04:30	Yes												1																		
	No					1	1																1	2	2						1
05:00	No	1																													

F. 계산 실수

수식의 각 글자를 리프 노드로 하는, 전형적인 세그먼트 트리 문제입니다.

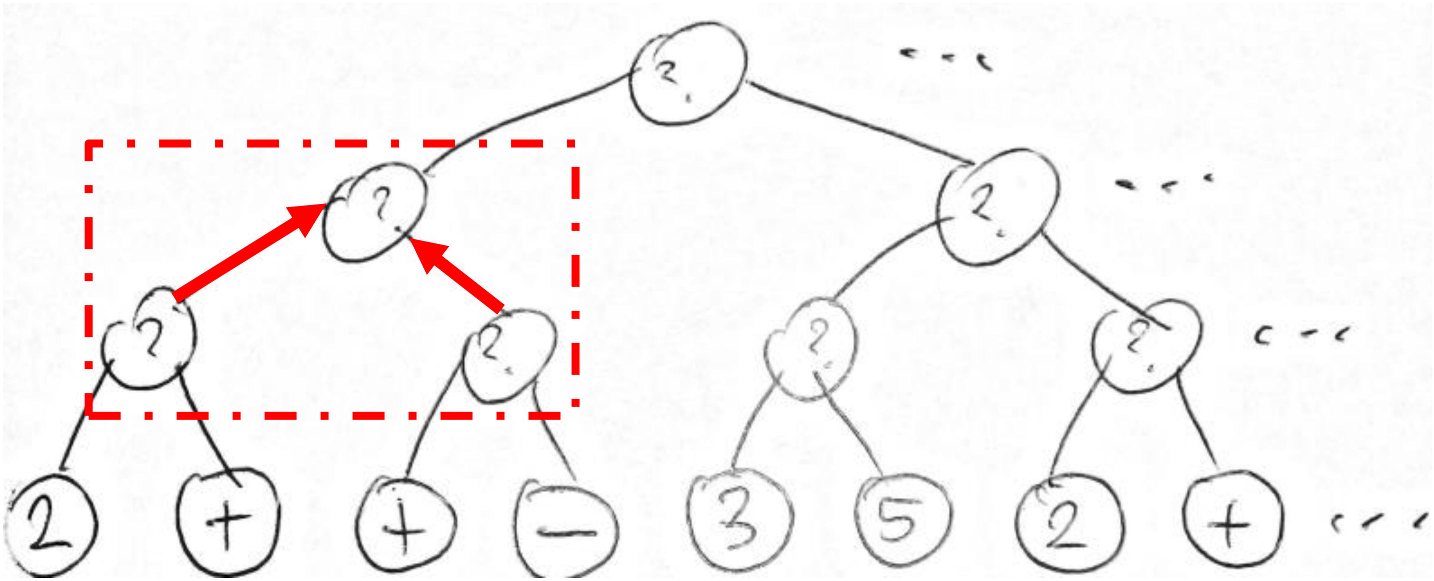
~~예선에 낸 문제가 너무 어려웠던 것 같아 쉽게 냈습니다.~~



F. 계산 실수

그렇다면, 두 노드가 대표하는 구간을 빠른 시간 안에 합칠 수 있어야 합니다.

어떻게 하면 합칠 수 있을지 알아보기 위해 두 구간을 놓고 생각해 봅시다.



F. 계산 실수

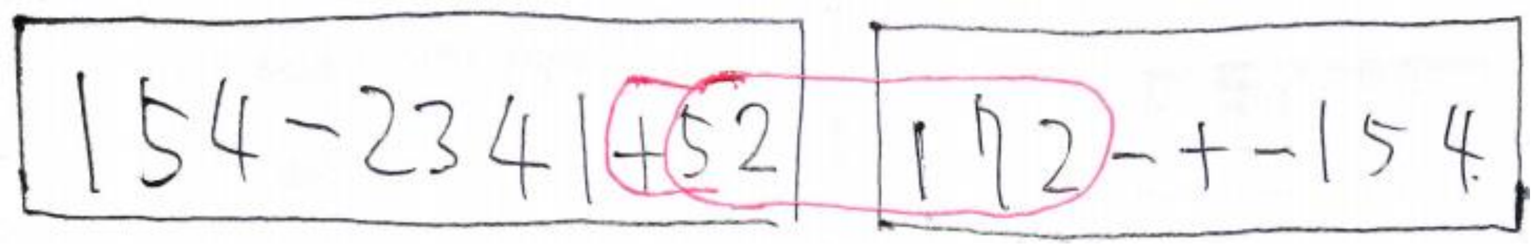
자 이 두 개의 구간을 합쳐봅시다.

154-2341+52

172-+-154

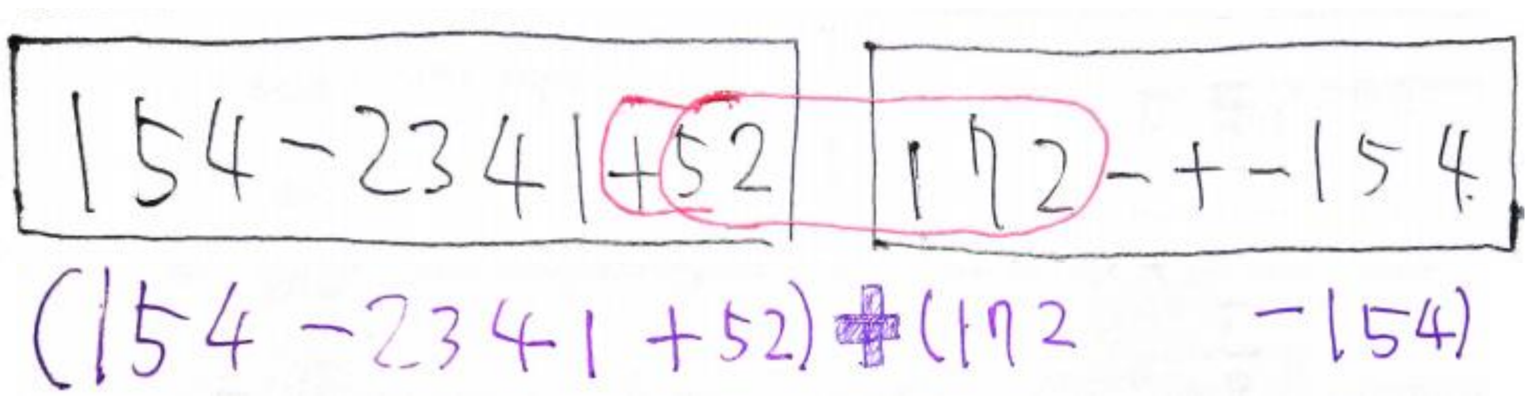
F. 계산 실수

저 +52172 부분이 겹쳐 있습니다. 이 겹친 부분을 제외하면 원래 작은 구간에서 계산하나, 합친 구간에서 계산하나 그 결과가 같습니다. 즉, 전체 값을 계산하기 위해..



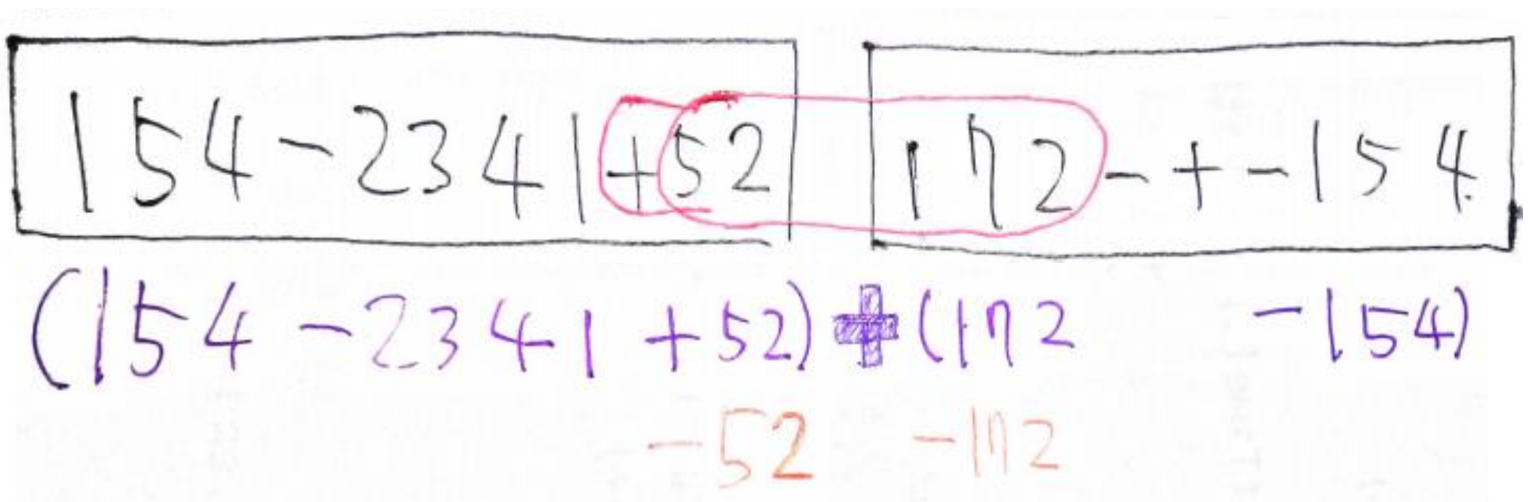
F. 계산 실수

먼저 작은 구간에 대해, 계산 결과를 미리 구해주고 합칩니다.



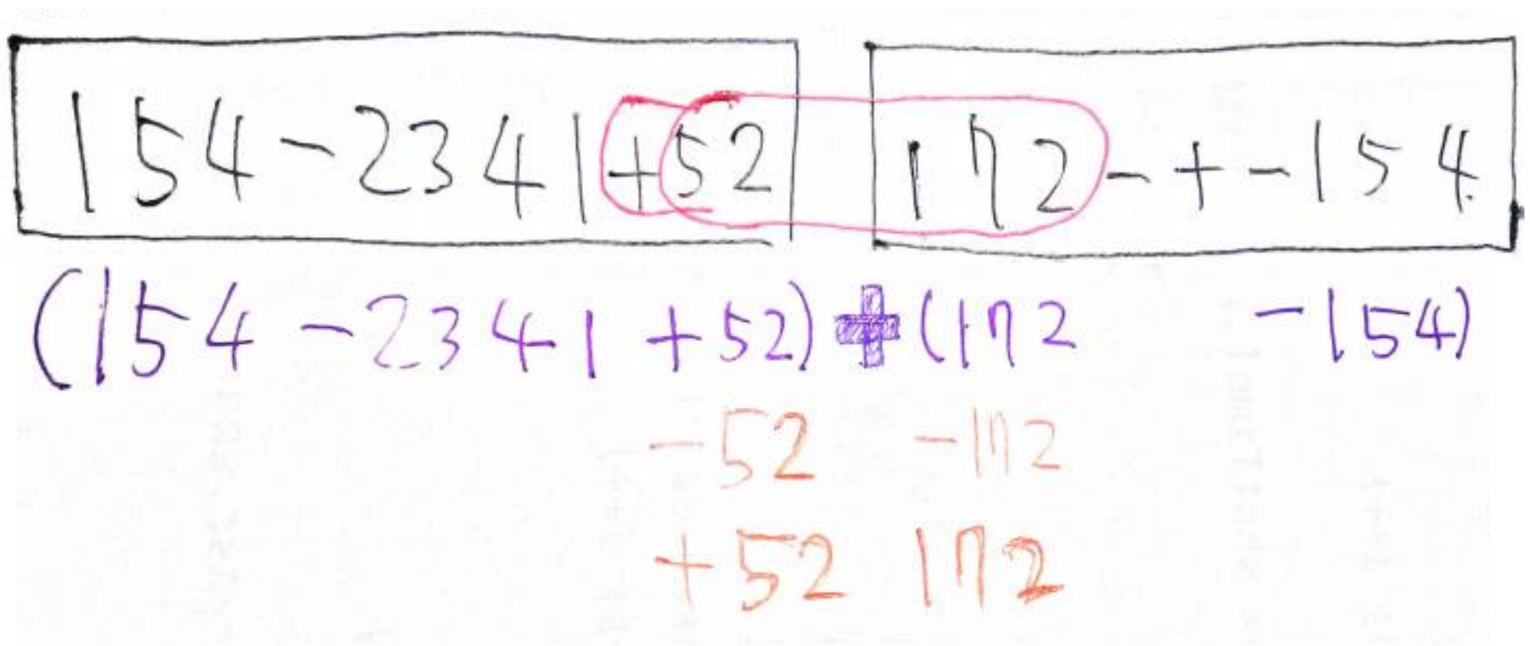
F. 계산 실수

겹치는 부분의 계산 결과를 빼 줌으로써 “154-2341”, “-154”만 계산한 효과를 만들고,



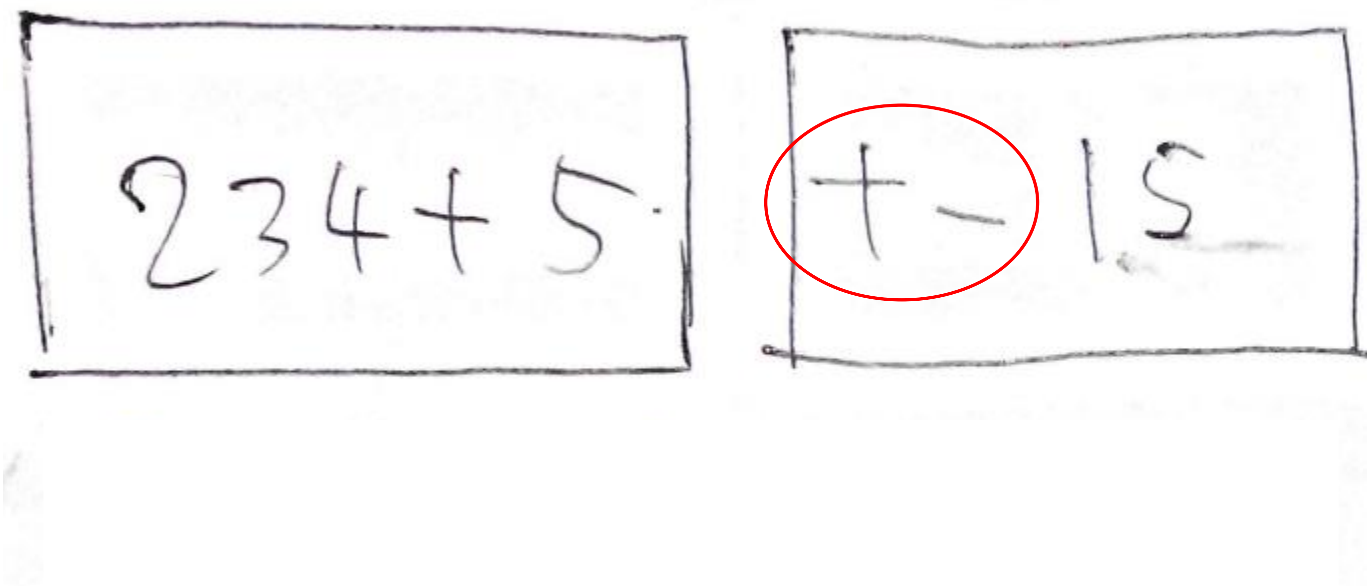
F. 계산 실수

마지막으로 겹치는 수인 +52172를 더해주면 되겠습니다.



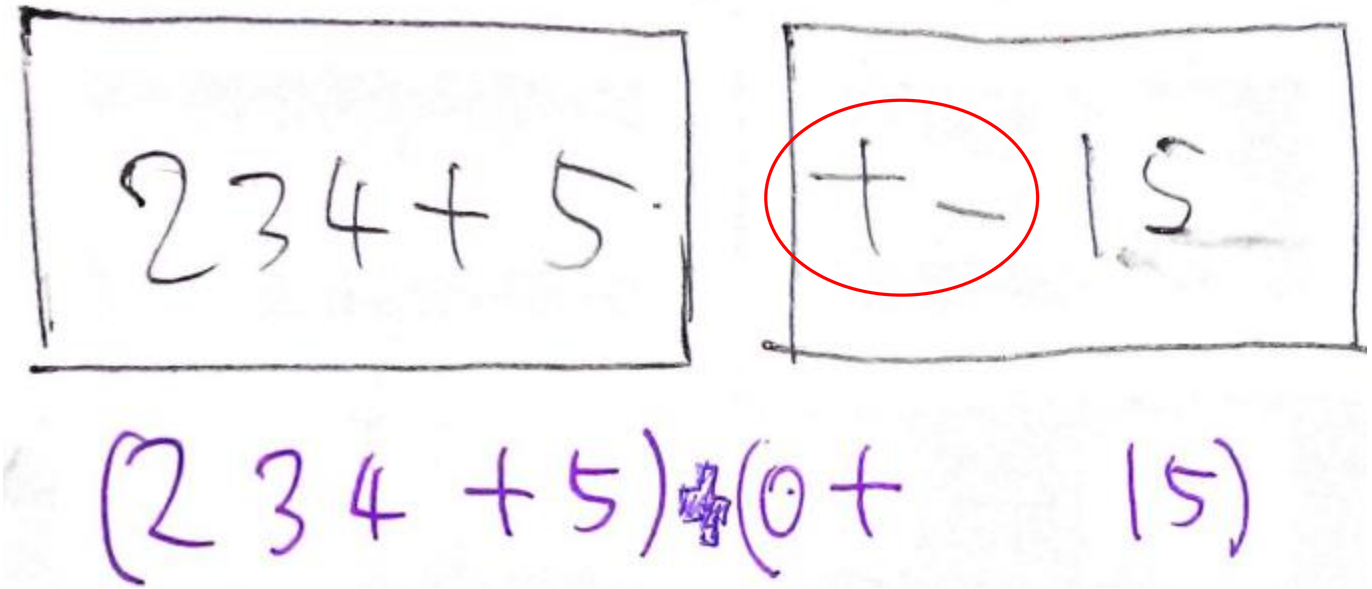
F. 계산 실수

다른 예제를 하나 살펴봅니다. 오른쪽 구간이 부호로 시작하는 경우,



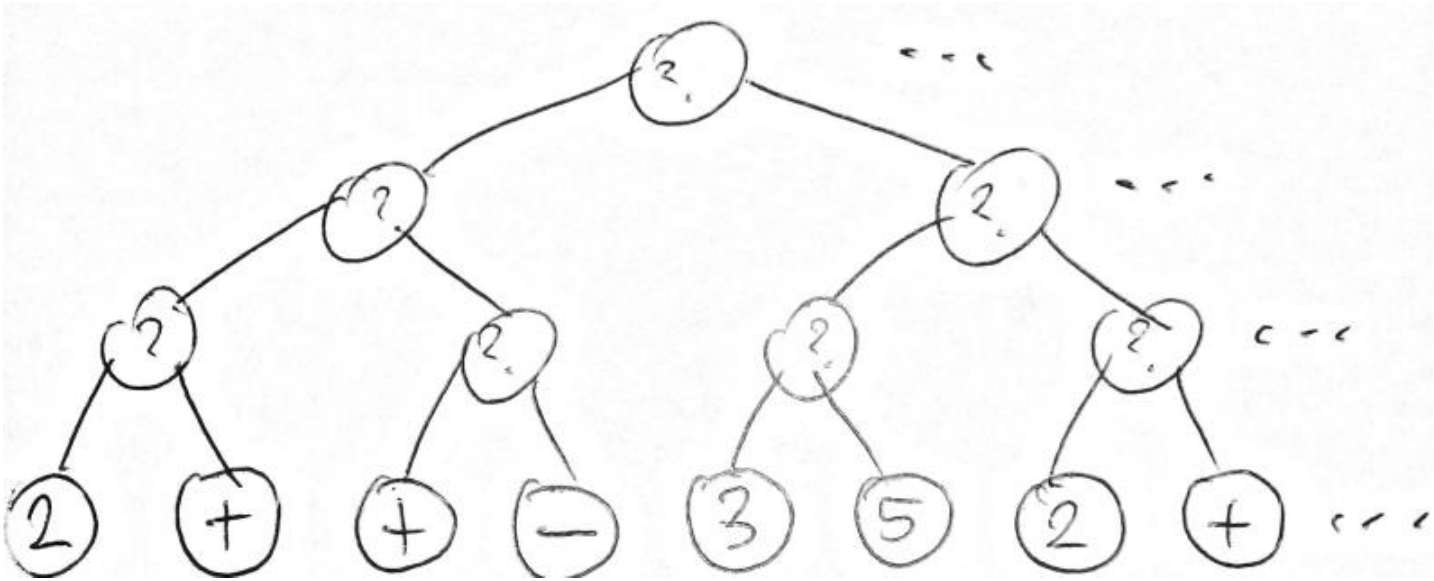
F. 계산 실수

그냥 각 구간의 계산 결과를 더해주면 된다는 것도 알 수 있습니다.



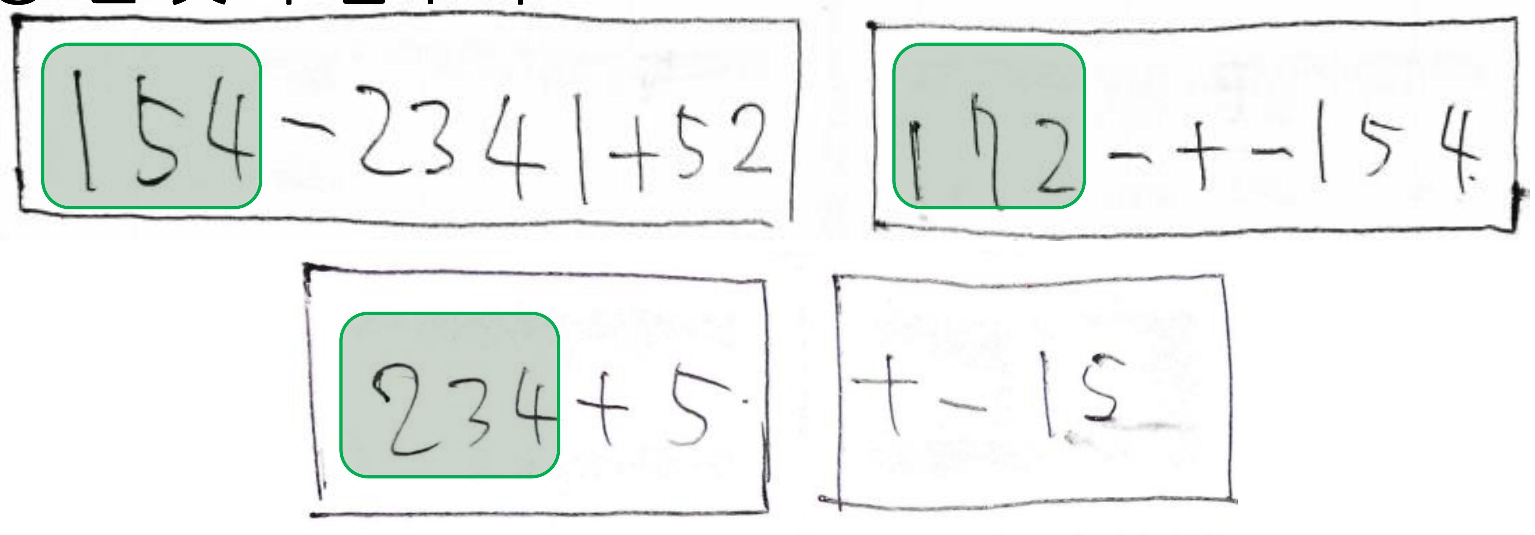
F. 계산 실수

이제 이 결과들을 종합하여, 세그먼트 트리의 각 노드에 어떤 값들을 저장해야 할 지를 생각해 봅시다. 크게 다음의 세 가지 값이 있습니다.



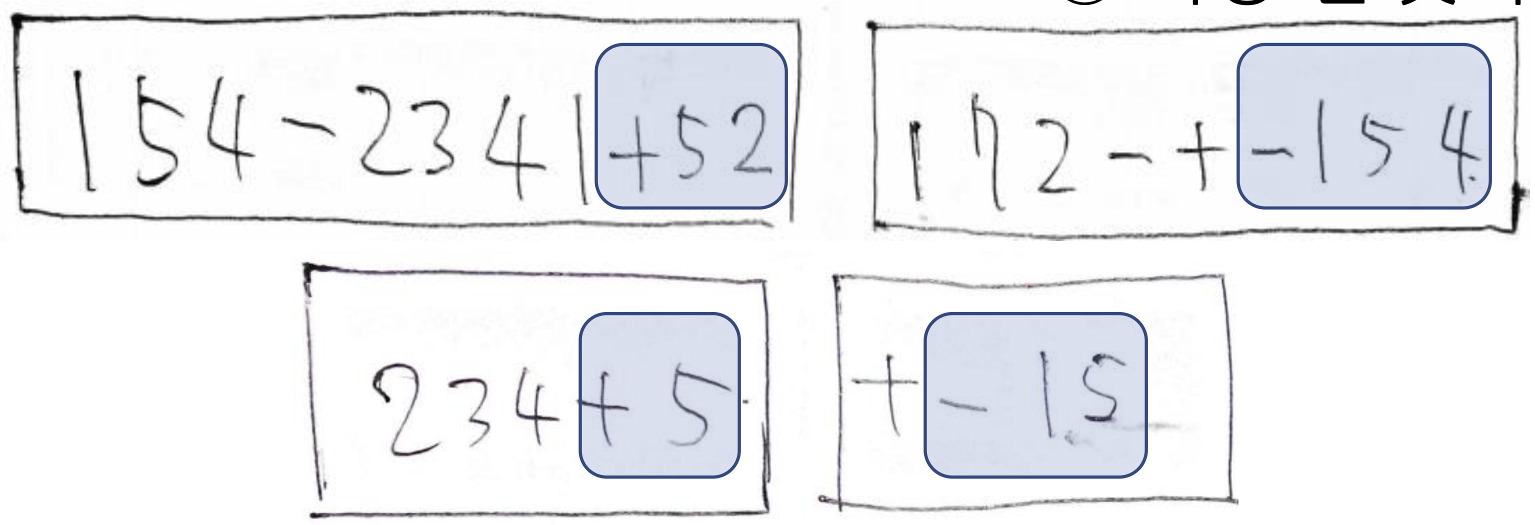
F. 계산 실수

① 가장 긴 숫자 접두사

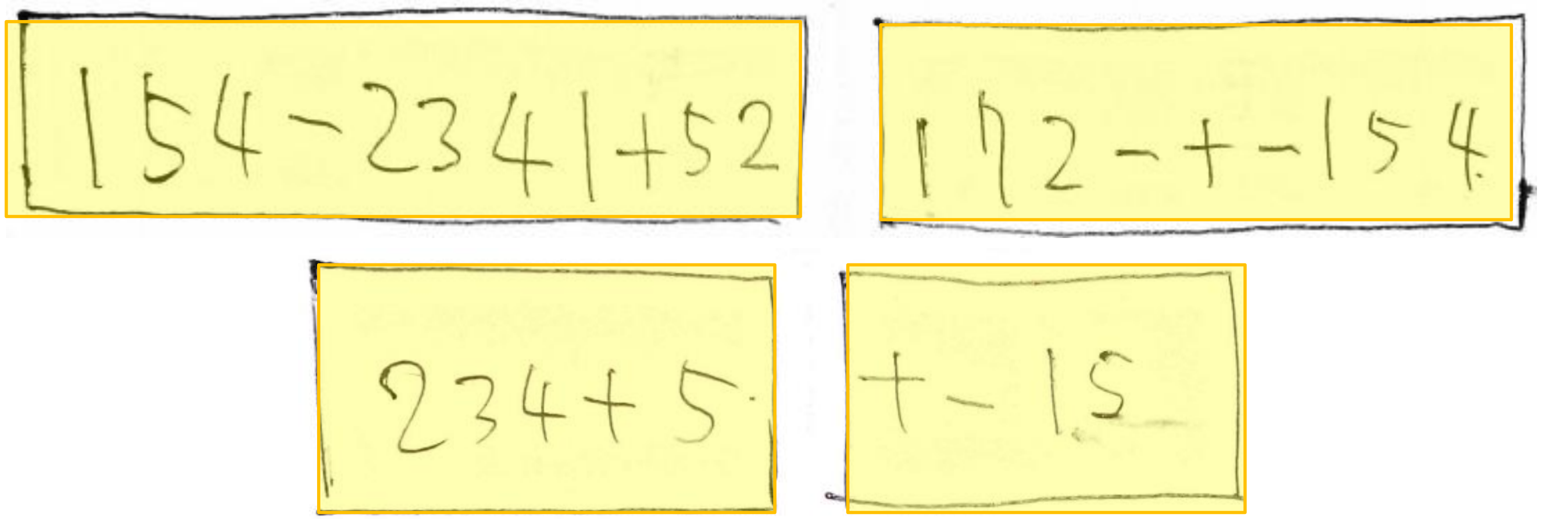


F. 계산 실수

② 가장 긴 숫자 접미사



F. 계산 실수

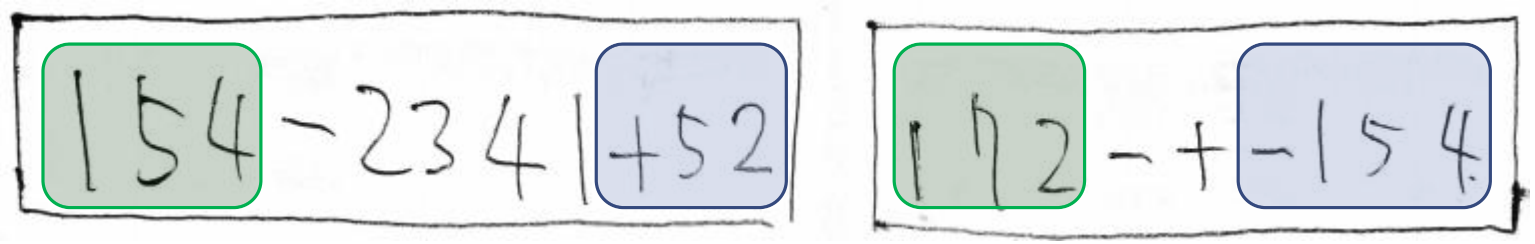


③ 각 구간의 계산 결과

F. 계산 실수

접두사/접미사라는 말이 애매할 수 있는데, 다음 세 가지 정보를 저장해 두면 됩니다. 사실 이 모든 정보를 저장할 필요는 없으나, 시간 제한을 넉넉하게 주어 가능하게 했습니다.

1. 글자 수 (문자열의 길이) 또는 $10^{\text{글자 수}}$
2. 맨 앞 글자가 ‘+’인가, ‘-’인가, 숫자인가?
3. 해당 수를 10억 7로 나눈 나머지



F. 계산 실수

합치는 것은 앞서 보여드린 방법 그대로 구현하시면 됩니다. 두 노드의 계산 결과를 합친 뒤, 겹치는 부분만 재조정하는 것이 핵심입니다.

앞 슬라이드에 제시된 모든 값을 저장하면 예외 처리를 크게 할 필요가 없으나, 메모리를 절약하려면 예외처리를 하는 코드가 필요할 수 있습니다.

10억 7로 나눈 나머지를 계산하는 과정에서 실수를 할 수 있으니 유의하시기 바랍니다.

추가 문제: 곱셈이 있다면 어떻게 할까요?

G. 순열 그래프의 전갈성 판별

- 맞은 팀 수 : 37
- 제출 횟수 : 92
- 정답률: 40.217%
- 처음 맞은 팀: Haha Hoho (ch_160, ㅎㅎ, Dongle) , 36분
- 출제자 : koosaga (구재현)
- 해설 작성자 : koosaga (구재현) + tncks0121(박수찬)

G. 순열 그래프의 전갈성 판별

		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
00:00	Yes																														
	No																														
00:30	Yes							1																							2
	No																		1												
01:00	Yes																1		1		1	1						1			
	No									1			1													1					1
01:30	Yes	2										1					1					1		1					1		1
	No					1					1	2	1				1		1			2	1		1				1		1
02:00	Yes				1						1	1				1				1	1		1								
	No	3	1		1																										

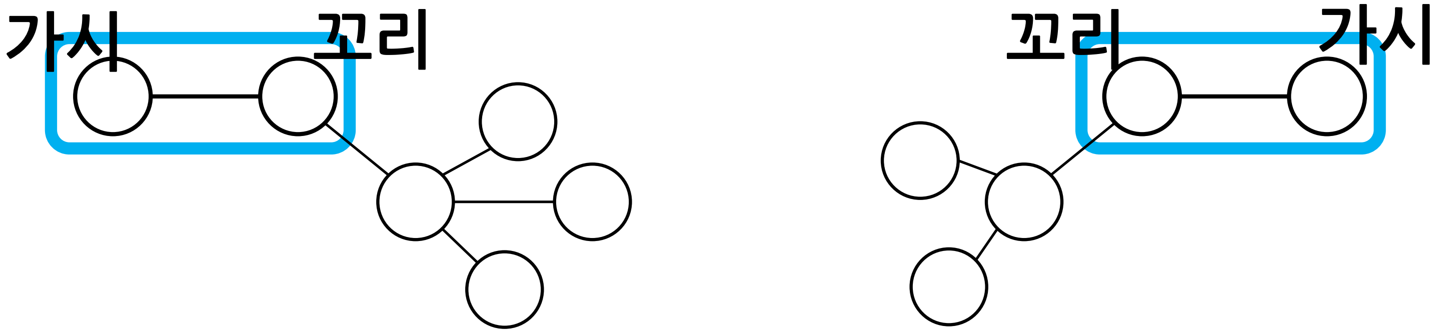
G. 순열 그래프의 전갈성 판별

		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
02:30	Yes	1	1																					1							
	No				1		1			1							1		1			1	1	1			1	2	1		1
03:00	Yes						1		1			1	1													1					
	No	1		1		1					1											1									
03:30	Yes					1				1				1																	
	No																														
04:00	Yes																												1	1	
	No														1			1						1	1	1		1	2	2	3
04:30	Yes										1																				
	No		1																												

G. 순열 그래프의 전갈성 판별

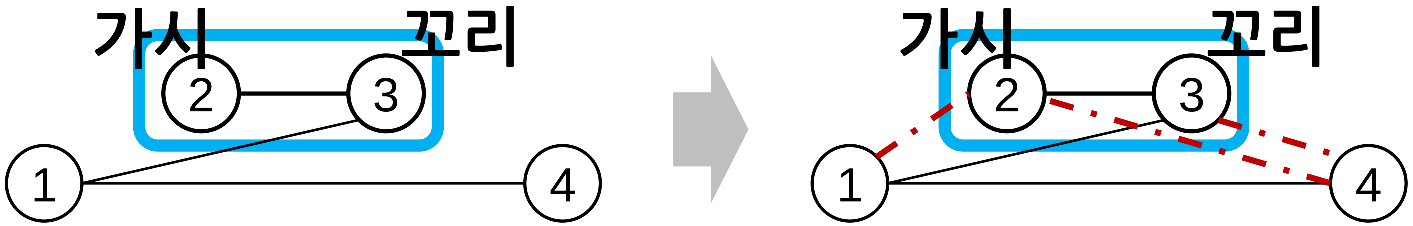
‘순열 그래프’이자 ‘전갈 그래프’인 어떤 그래프를 고려해봅시다.

가시 정점과 꼬리 정점을 묶어서 생각합니다. 가시 정점과 꼬리 정점 양 옆에 있는 정점들을 생각했을 때, 모든 정점은 이 묶인 두 정점보다 왼쪽에 있거나, 오른쪽에 있습니다. (편집자 주: 번호가 커질수록 오른쪽이라고 생각하세요)



G. 순열 그래프의 전갈성 판별

그래프를 연결 그래프로 유지하면서, 조건을 만족하게 그래프 양 옆에 모두 1개 이상의 정점을 배치하는 것이 불가능하기 때문입니다.

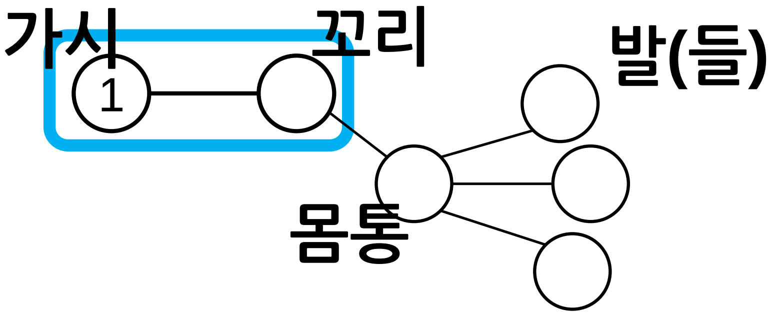


$a_2 > a_3, a_1 > a_3, a_1 > a_4$ $a_2 > a_3, a_1 > a_3, a_1 > a_4$

고로, 모든 정점은 가시 정점과 꼬리 정점의 왼쪽이나 오른쪽에 나열되어 있습니다.

G. 순열 그래프의 전갈성 판별

편의상 모두 오른쪽에 있다고 가정했을 때, 가시 정점과 꼬리 정점 중 하나는 1번 정점일 것입니다. 가시 정점이 1번 정점이라고 가정합시다.



G. 순열 그래프의 전갈성 판별

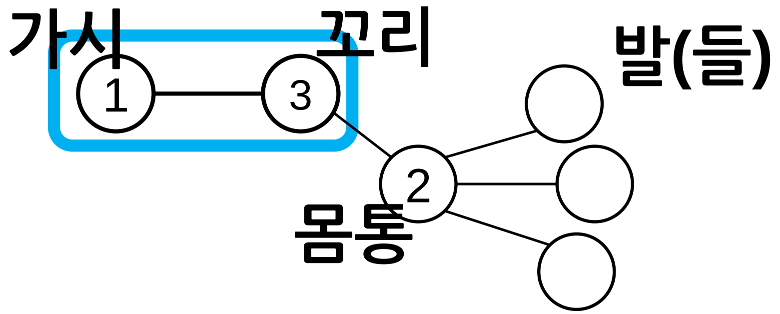
꼬리 정점은 유일하게 가시 정점과 연결된 정점이므로,

$$a_1 = 2, a_{\text{꼬리}} = 1$$

입니다. 이 때 $a_{\text{꼬리}} = 1$이므로 모든 $u <$ 꼬리에 대해 $a_u > a_{\text{꼬리}}$가 성립하여 $(u, \text{꼬리})$ 간선이 존재합니다. 따라서 꼬리 정점과 유일하게 연결된 몸통 정점을 만들기 위해서는

꼬리 = 3, 몸통 = 2

이어야 합니다.



G. 순열 그래프의 전갈성 판별

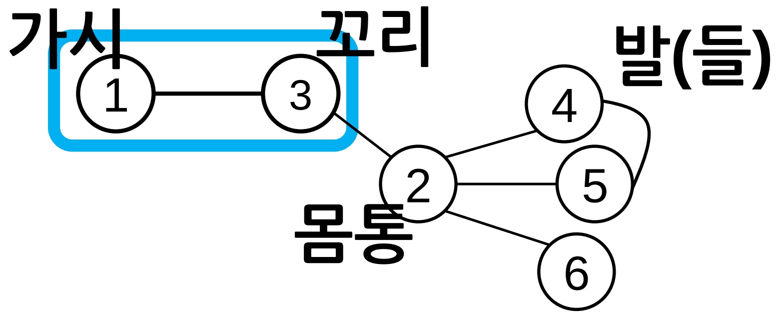
또한 3, 4, 5, 6, .., 번 정점(꼬리 + 발)과 2번 정점(몸통)이 연결되어 있으므로,

$$a_2 > \max(a_3, a_4, \cdots, a_N)$$

이어야 하고, $a_1 = 2$임이 이미 결정되어 있기에

$$a_2 = N$$

이어야 합니다. 이제 순열 a로 순열 그래프를 그려 보면 전갈 그래프입니다.

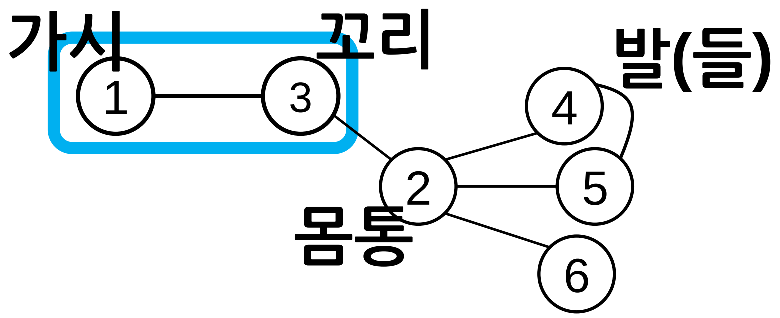


G. 순열 그래프의 전갈성 판별

정리하면, 모든 정점이 (가시 정점, 꼬리 정점) 묶음의 오른쪽에 있고, 가시 정점의 번호가 1인 경우에는

$$a_1 = 2, a_2 = N, a_3 = 1$$

을 만족해야만 전갈 그래프가 됩니다. 또한 $a_1 = 2, a_2 = N, a_3 = 1$을 만족하는 수열 a로 순열 그래프를 만들면 항상 전갈 그래프입니다. 즉, 필요충분조건입니다.

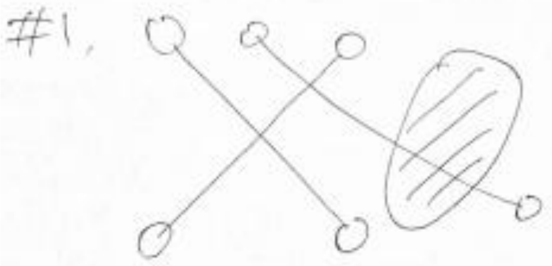


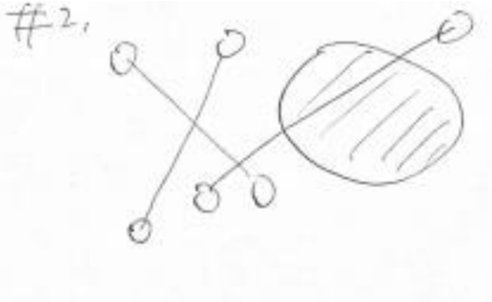
G. 순열 그래프의 전갈성 판별

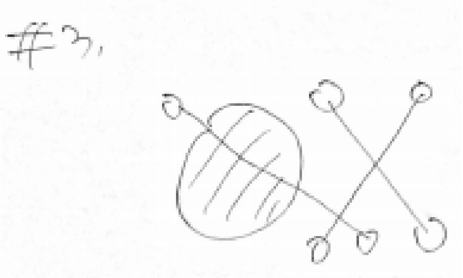
이와 같은 방식으로, 모든 정점이 (가시 정점, 꼬리 정점) 묶음의 왼쪽/오른쪽에 있는지, 가시/꼬리 중 어느 정점이 1번/N번 정점인지에 따라 정확히 $2^2 = 4$개의 패턴을 만들 수 있습니다.

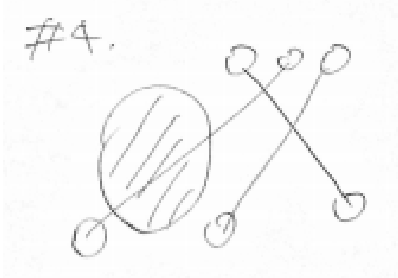
이 패턴을 만족하는 그래프만이 전갈 그래프일 수 있으며, 이 패턴을 만족하는 그래프는 모두 전갈 그래프입니다.

패턴을 알면, 자명히 $O(1)$에 순열이 패턴을 만족하는지 판단할 수 있습니다.









G. 순열 그래프의 전갈성 판별

문제의 아이디어는 이론전산학의 난제 중 하나인 Aanderaa-Karp-Rosenberg 추측에서 따 왔습니다.

두 정점 i, j를 잇는 무방향 간선이 있는지를 검사하는 $has\_edge(i, j)$ 라는 함수를 생각해 봅시다. Aanderaa-Karp-Rosenberg 추측을 개략적으로 요약하자면, 대부분의 비자명한 그래프 성질 (트리인지, 이분 그래프인지 등..)을 판단하기 위해서는 이 함수를 $n(n-1)/2$ 번 호출해야 한다는 내용의 추측입니다.

"모든"이 아니라 "대부분"인 이유는, "전갈스러움"이라는 성질이 이 추측의 반례로 밝혀져, 이후 이 문제에 추가적인 조건이 부여되었기 때문입니다.

자세한 내용에 대해서는 영문 위키백과를 참고하는 것을 추천드립니다.

H. Checkpoint

- 맞은 팀 수 : 9
- 제출 횟수 : 70
- 정답률: 12.857%
- 처음 맞은 팀: 넥슨은 다람쥐를 뿌려라 (ch_434, ntopia, unused, kaze) , 58분
- 출제자 : kcm1700 (김찬민)
- 해설 작성자 :

H. Checkpoint

		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
00:00	Yes																														
	No																														
00:30	Yes																													1	
	No				1								1							1											
01:00	Yes															1															
	No																														
01:30	Yes	1	1																				1								
	No			1																											1
02:00	Yes													1																	
	No													1						1											

H. Checkpoint

		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
02:30	Yes																														
	No										1								1								2				
03:00	Yes																														
	No								1					1	1				1										1		
03:30	Yes																														
	No									1							1					1					1		1		
04:00	Yes												1						1		1										
	No		1	1					1		1		1	2						1					1			2	1		
04:30	Yes																														
	No	4	5		1	1				1	1					2	3	1	1	1	2		1				1	2			1

H. Checkpoint

아직 풀이가 준비되지 않았습니다.

조속히 공개하도록 하겠습니다.

I. The Show Must Go On

- 맞은 팀 수 : 1
- 제출 횟수 : 15
- 정답률: 6.67%
- 처음 맞은 팀: Andromeda Express (Feat. kriii) (ch_46, Astein, ainu7, kriii) , 208분
- 출제자 : koosaga (구재현)
- 해설 작성자 : koosaga (구재현) + tncks0121 (박수찬)

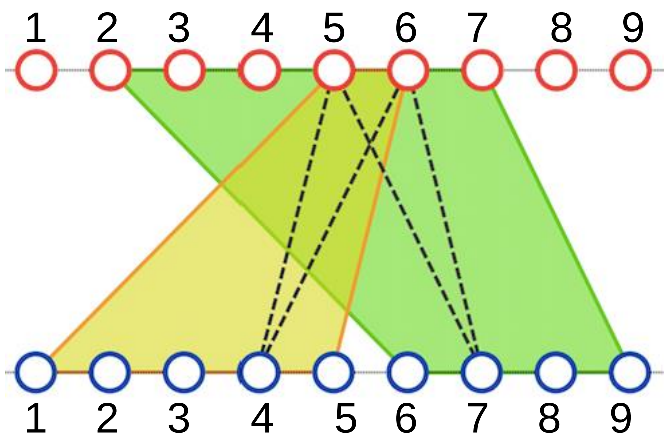
I. The Show Must Go On

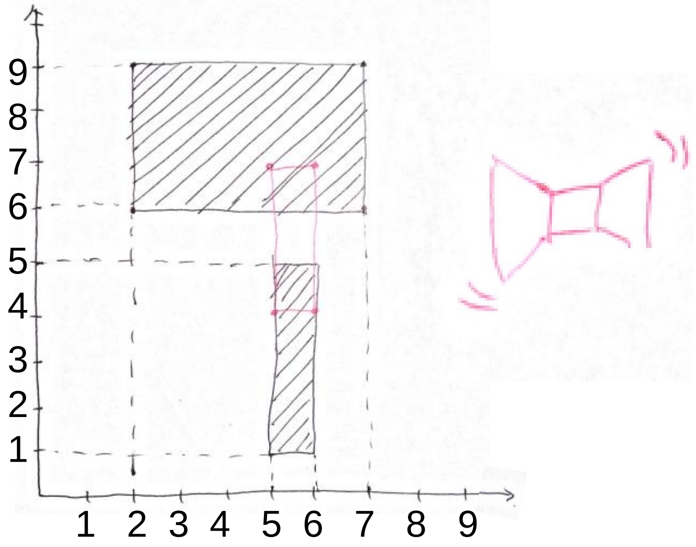
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
02:30	Yes																														
	No																														
03:00	Yes																													1	
	No																											1			
03:30	Yes																														
	No																							1							
04:00	Yes																														
	No																														
04:30	Yes																														
	No								2	5	2															1				1	1

I. The Show Must Go On

그래프 이론 문제처럼 보이지만, 기하 문제로 해석하면 깔끔하게 문제를 변형할 수 있습니다.

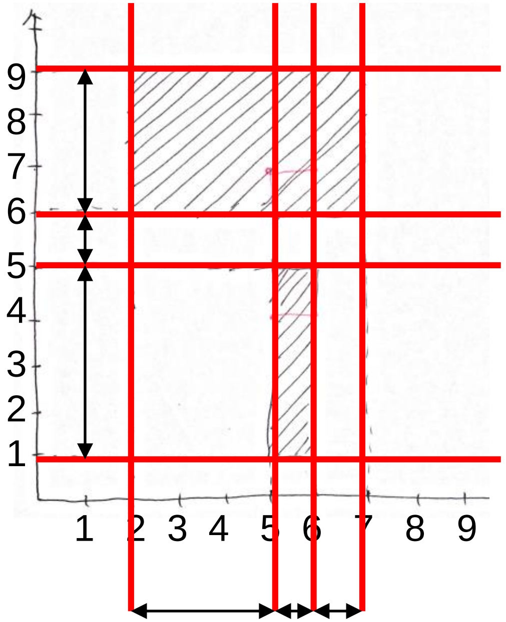
천장에 있는 점을 x축, 바닥에 있는 점을 y축에 사영시키면, 주어진 사다리꼴은 직사각형의 모습이고, '나비넥타이'는, 네 꼭짓점이 주어진 직사각형 속에 속한, 직사각형을 뜻함을 알 수 있습니다.





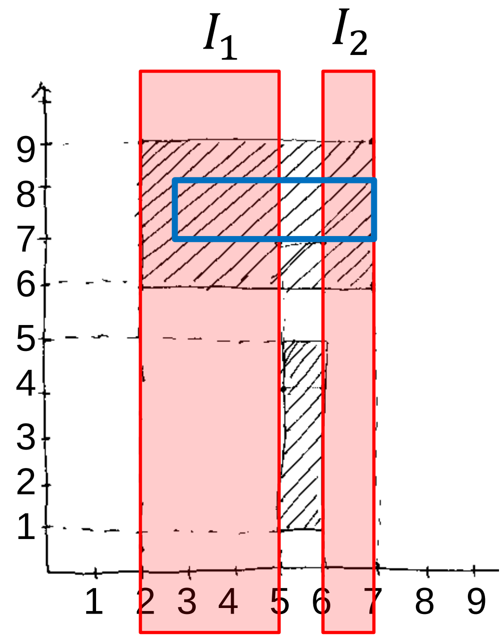
I. The Show Must Go On

이렇게 기하 문제로 바꾸고 나면, 비록 $N \leq 10^9$이지만, 직사각형의 서로 다른 x좌표/y좌표는 $2M$개이기에, 각 축마다 대략 $2M$개의 구간만이 의미를 가지게 된다고 생각할 수 있습니다.



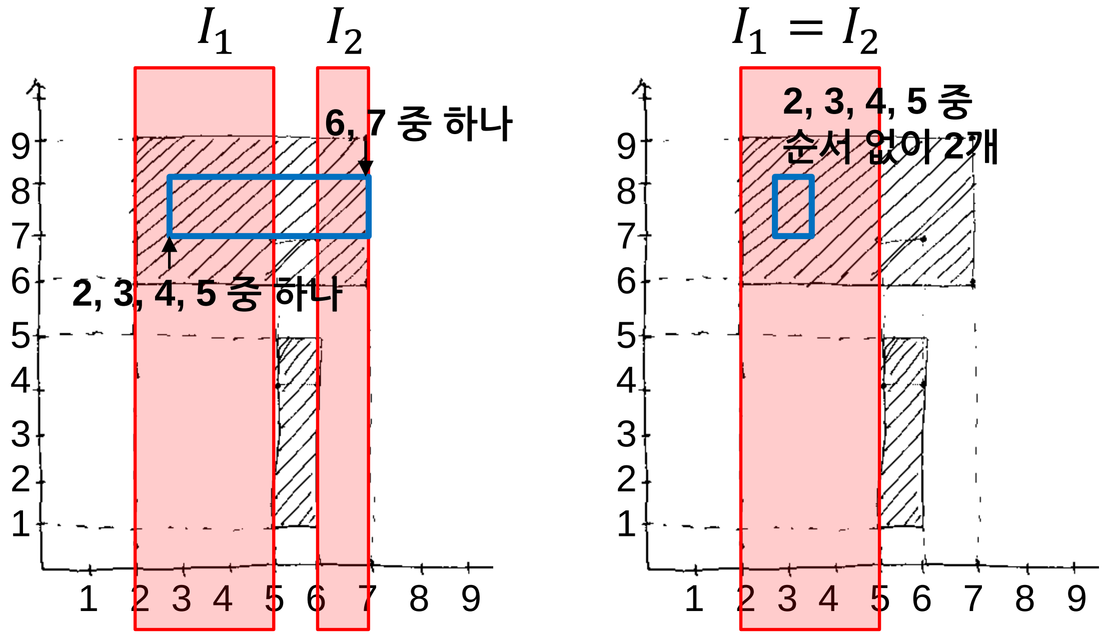
I. The Show Must Go On

이를 바탕으로 ‘나비넥타이’의 수를 세어 보도록 합시다. 나비넥타이의 x좌표가 x좌표 구간 I_1, I_2에 속한 모든 경우의 수를 세어 보도록 합시다.



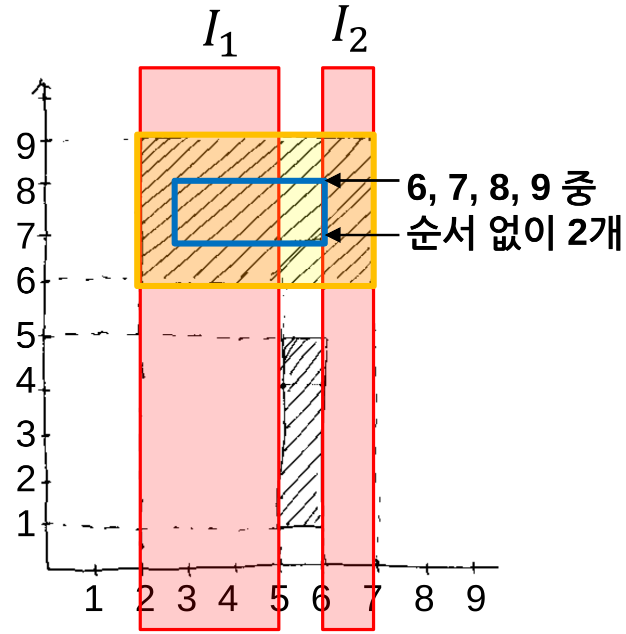
I. The Show Must Go On

직사각형의 x좌표만 고려해 보면, 가능한 경우의 수가 $\mathrm{len}(I_1) \times \mathrm{len}(I_2)$가지임을 알 수 있습니다. 단 $I_1 = I_2$일 때에는 $\binom{\mathrm{len}(I_1)}{2}$가지입니다. 여기서 $\mathrm{len}(I)$는 구간 I에 속한 x좌표의 수입니다.



I. The Show Must Go On

직사각형의 y좌표만 고려해 보면, 가능한 경우의 수는 I_1과 I_2에서 동시에 존재하는 y좌표의 개수를 $\mathrm{Intersection}(I_1, I_2)$ 라고 할 때, $\binom{\mathrm{Intersection}(I_1,I_2)}{2}$가지입니다.



I. The Show Must Go On

정리하면, 나비넥타이의 x좌표가 x좌표 구간 I_1, I_2에 속한 모든 경우의 수는

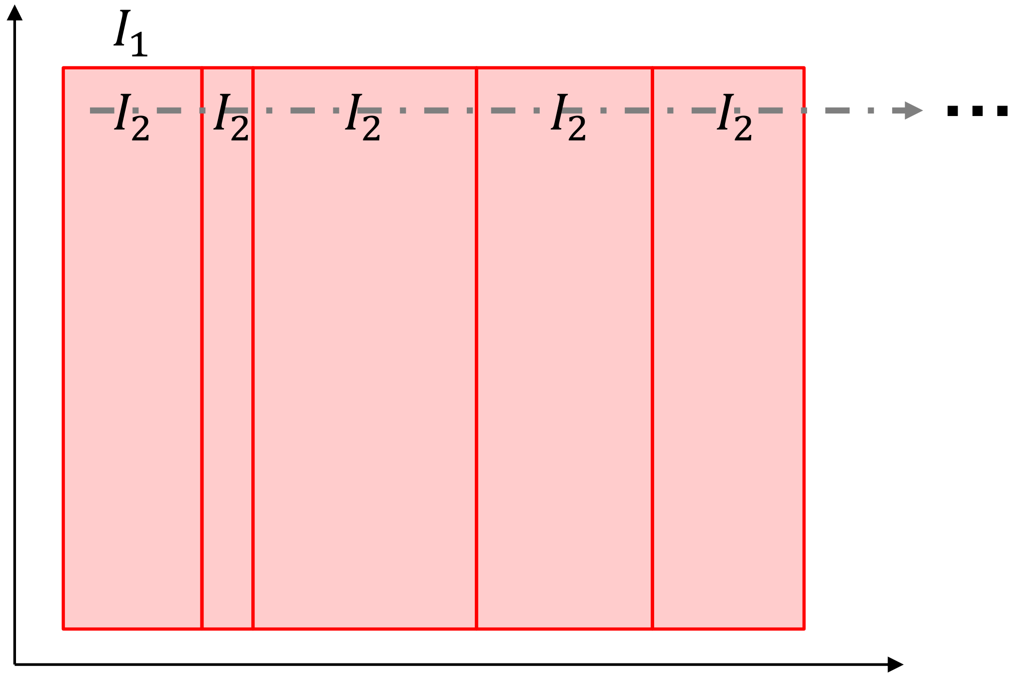
$$\begin{cases} \operatorname{len}(I_1) \times \operatorname{len}(I_2) \times \binom{\operatorname{Intersection}(I_1, I_2)}{2} & I_1 \neq I_2 \\ \binom{\operatorname{len}(I_1)}{2} \times \binom{\operatorname{Intersection}(I_1, I_2)}{2} & I_1 = I_2 \end{cases}$$

입니다. 이를 모든 (I_1, I_2) 쌍에 대해 계산한 후 합치면 답을 구할 수 있습니다.

(I_1, I_2) 쌍의 수는 $O(M^2)$개입니다. $\operatorname{len}(I)$는 $O(1)$만에 구할 수 있으므로, 전체 시간복잡도는 $\operatorname{Intersection}(I_1, I_2)$에 의해 좌우됩니다. 단순히 모든 y좌표 구간을 순회하는 방식으로 구하면 $\operatorname{Intersection}(I_1, I_2)$의 시간복잡도가 $O(M)$이고, 따라서 전체 시간복잡도가 $O(M^3)$입니다. 느립니다.

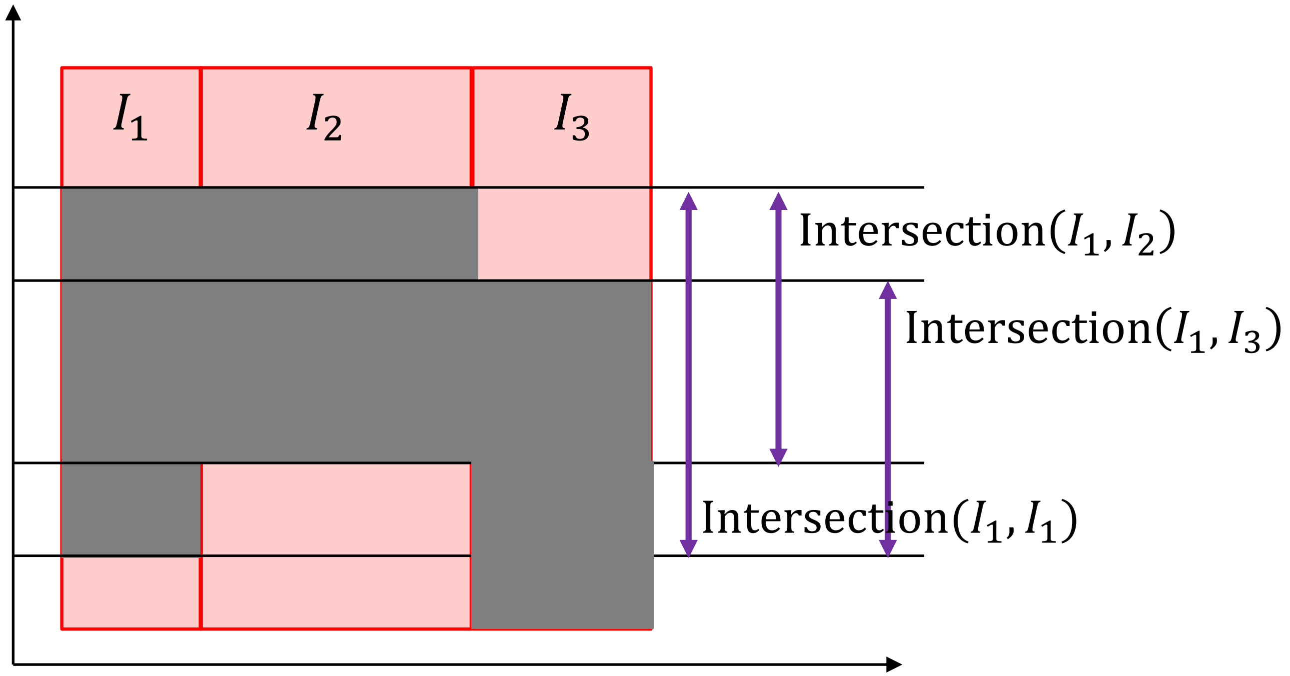
I. The Show Must Go On

따라서 Intersection(I_1, I_2)의 시간복잡도를 줄여야 합니다. 단순히 줄일 수는 없고, (I_1, I_2) 쌍이 이미 정해져 있다는 사실을 이용하여 Plane Sweeping을 할 수 있습니다. I_1을 고정시켜놓고, I_2를 점점 오른쪽으로 옮겨가는 방식으로 시행합니다.



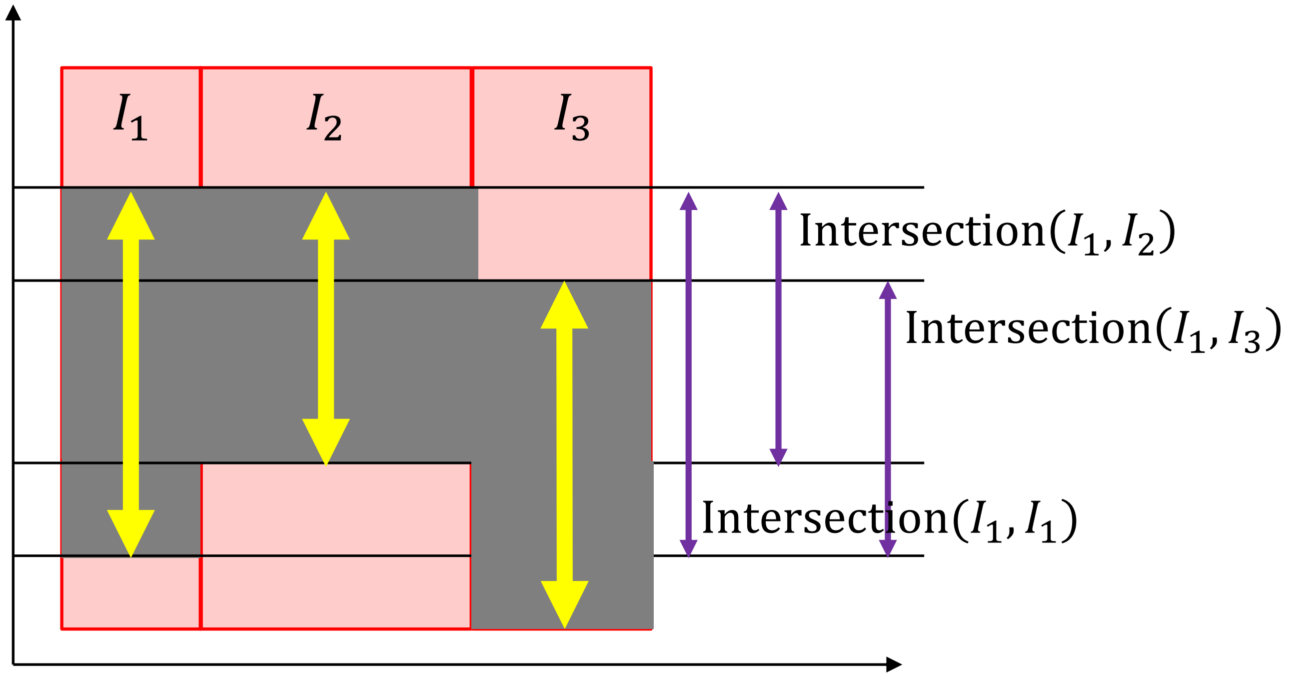
I. The Show Must Go On

방법은 다음과 같습니다. 우선 모든 직사각형이 I_1 오른쪽(경계 포함)에 있도록 범위를 잘 조정합니다. 그 후 직사각형의 합집합의 넓이를 구하는 방법과 비슷한 요령으로 구합니다.



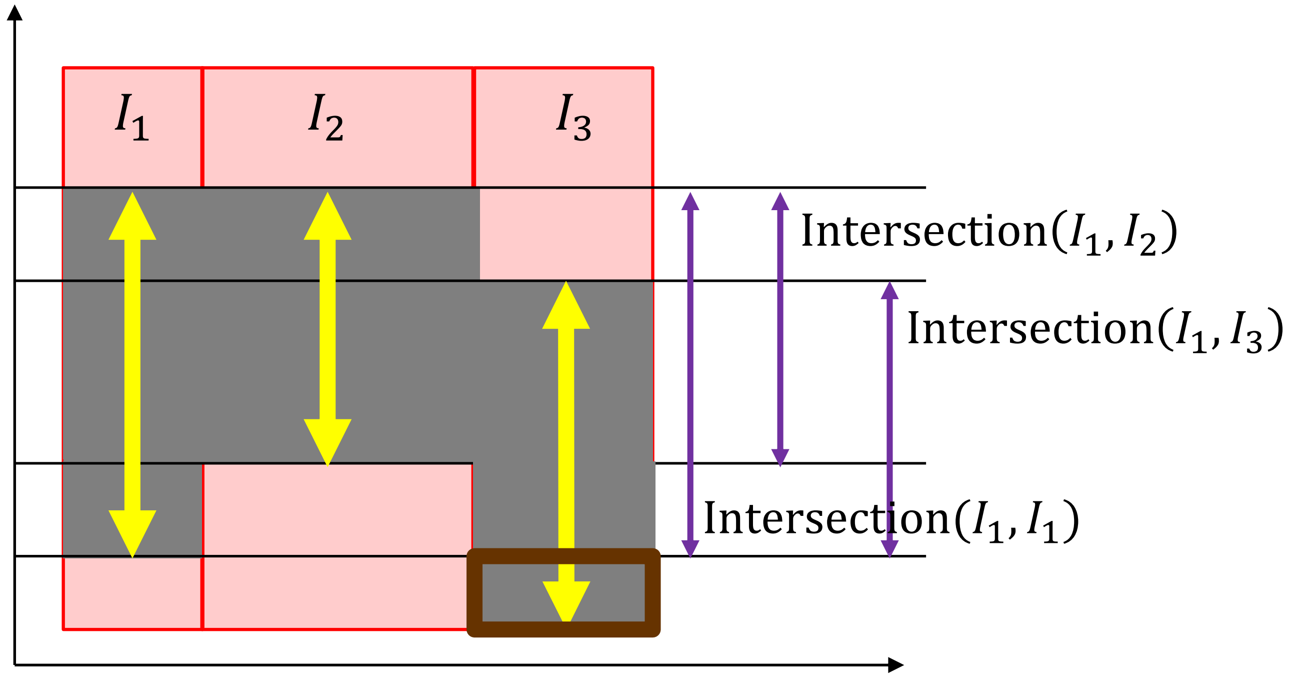
I. The Show Must Go On

해당 방법을 잘 살펴보면, Segment tree와 같은 자료구조를 활용하여, 각 x축 구간에 대해 합집합의 y좌표의 개수를 구합니다. 이 문제에서 해야 할 일과 상당히 유사하지만, 살짝 다른 점이 있습니다.



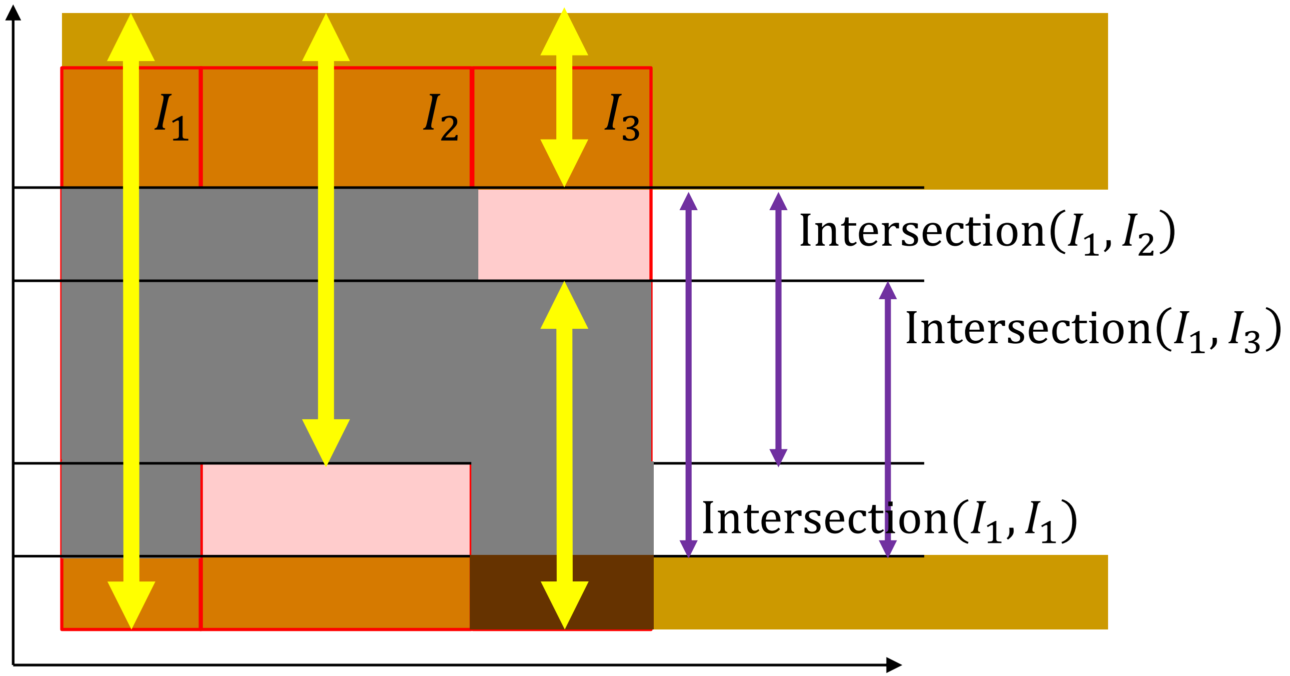
I. The Show Must Go On

바로 I_1과의 “교집합”을 구해야 한다는 것입니다. 합집합 구하는 방법에서는 아래의 강조된 부분과 같이 교집합이 아닌 부분도 고려해 줍니다.



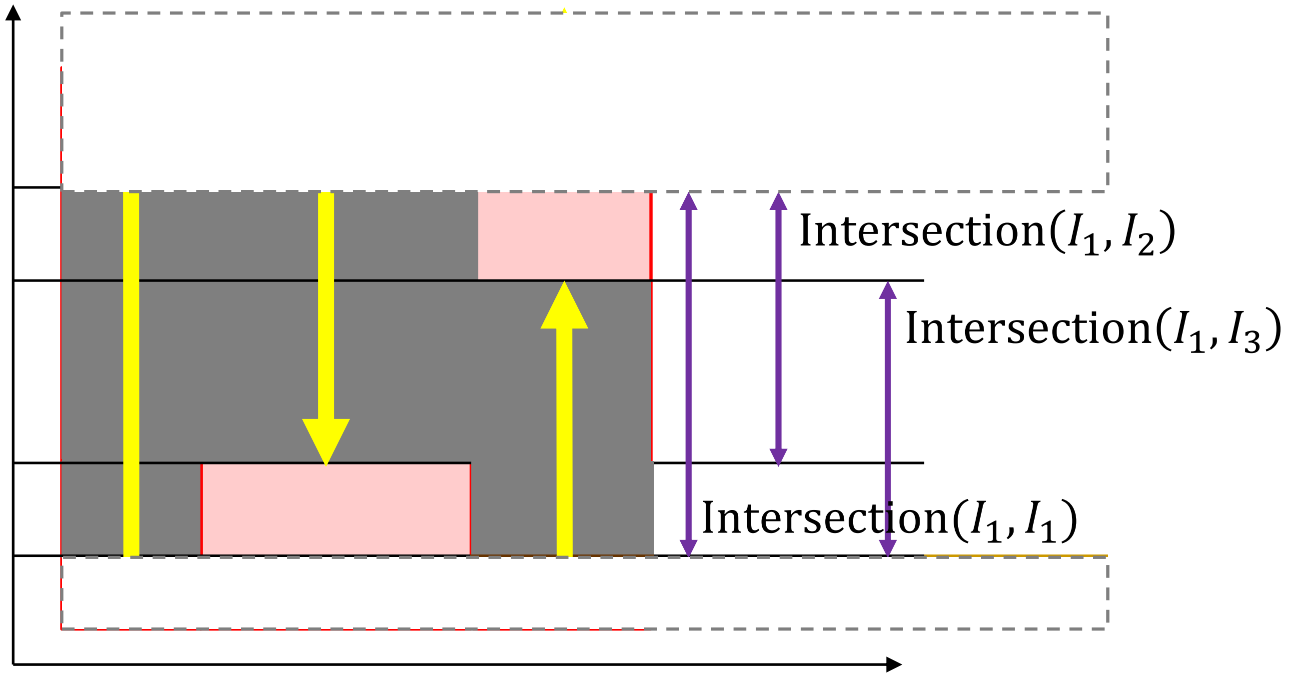
I. The Show Must Go On

이를 해결하기 위해서, y축 구간들 가운데 I_1의 직사각형이 덮지 않는 구간들에는 미리 긴 직사각형을 만들어둡니다. 그런 후 각 x축 구간에 대해 합집합의 y좌표의 개수를 구하면, 새로 만든 직사각형의 구간들은 항상 세어집니다.



I. The Show Must Go On

그러면 전체에서 항상 세어지는, 교집합이 아닌 부분을 제외하면, 원하는 답을 알 수 있게 됩니다.



I. The Show Must Go On

이 풀이를 사용하면 각 I_1마다 $\mathrm{Intersection}(I_1, *)$의 값을 $O(M\log M)$의 시간복잡도로 구할 수 있습니다. 구간 I_1은 총 $O(M)$개 있으므로, 시간복잡도는 $O(M^2 \log M)$이 되어 문제를 해결할 수 있습니다.

J. 도박과 사각형

- 맞은 팀 수 : 7
- 제출 횟수 : 68
- 정답률: 10.294%
- 처음 맞은 팀: Anti-Q (ch_49, tonyjjw, dotorya, qo) , 61분
- 출제자 : RiKang (이승재)
- 해설 작성자 : RiKang (이승재)

J. 도박과 사각형

		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
00:00	Yes																														
	No																														
00:30	Yes																														
	No																														
01:00	Yes		1			1								1																	
	No																														
01:30	Yes																														
	No																														
02:00	Yes																						1								
	No		1					1																							

J. 도박과 사각형

		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
02:30	Yes																														
	No															1			1												
03:00	Yes																	1													
	No									1	1					1			1									1			
03:30	Yes																														
	No									1				1							1	2	3	4	4	1	1	2	1	1	
04:00	Yes				1																										
	No					2	3	3		1			3		2	1			1	2	2										
04:30	Yes																1														
	No																		2		1			1	1			3	1	1	

J. 도박과 사각형

사각형에 적힌 수가 1, 2, 3, 4, 5밖에 없으므로, 각각의 경우에 대해 기댓값을 구해서 더하는 방법을 사용합니다.

$$E(S) = E\left((c_1)^2\right) + 2E\left((c_2)^2\right) + 3E\left((c_3)^2\right) + 4E\left((c_4)^2\right) + 5E\left((c_5)^2\right)$$

숫자 2가 좋으므로 $E\left((c_2)^2\right)$를 구해보겠습니다.

먼저,

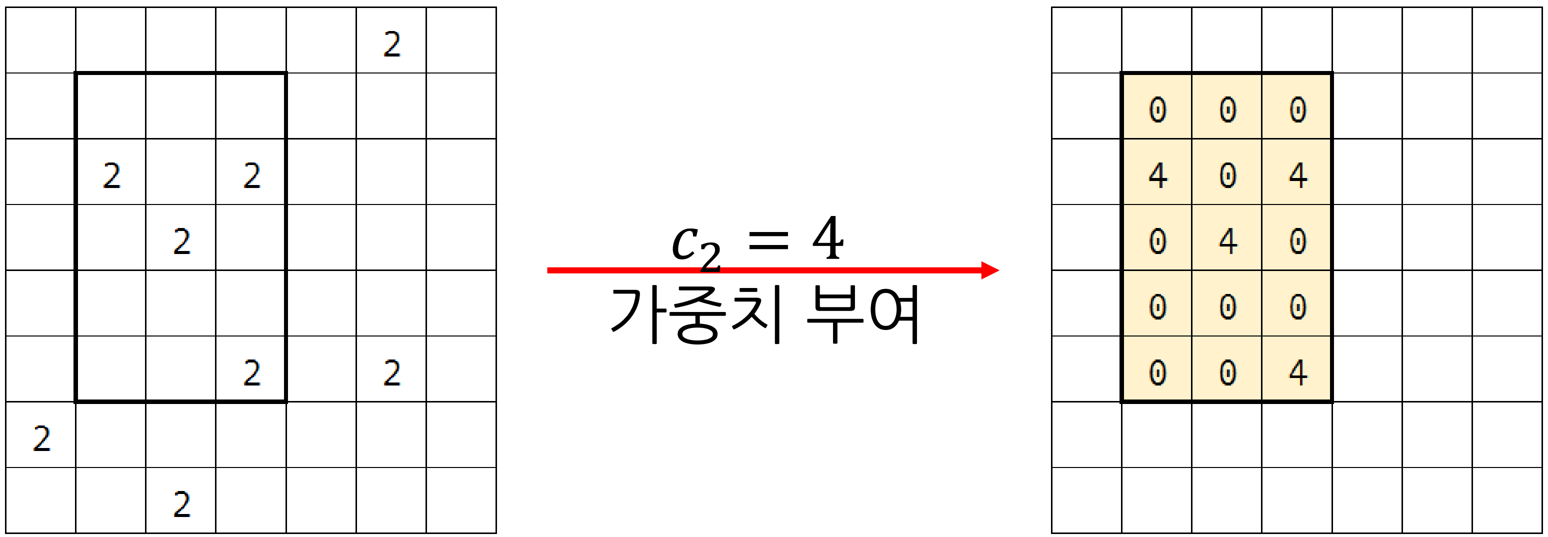
$$(c_2)^2 = \underbrace{c_2 + c_2 + c_2 + \cdots + c_2}_{c_2\text{번}}$$

임을 이용합니다.

J. 도박과 사각형

그래서, 어떤 직사각형에 대해, 각 격자에 2가 적힌 칸에는 가중치 c_2를, 2가 적히지 않은 칸에는 가중치 0을 부여하면, 직사각형에 2가 c_2개 적혀 있으므로

$$(c_2)^2 = \sum_{\text{모든 격자}} (\text{직사각형에서 격자의 점수})$$



J. 도박과 사각형

이 식이 항상 성립하므로,

$$\sum_{\text{모든 직사각형}} (c_2)^2 = \sum_{\text{모든 직사각형}} \sum_{\text{모든 격자}} (\text{직사각형에서 격자의 점수})$$

역시 성립합니다.

이 식을 다시 쓴,

$$\sum_{\text{모든 직사각형}} (c_2)^2 = \sum_{\text{모든 격자}} \sum_{\text{모든 직사각형}} (\text{직사각형에서 격자의 점수})$$

도 성립합니다.

J. 도박과 사각형

그래서,

$$\mathrm{Point}(i,j) = \sum_{\text{모든 직사각형}} (i,j)\text{ 격자의 점수}$$

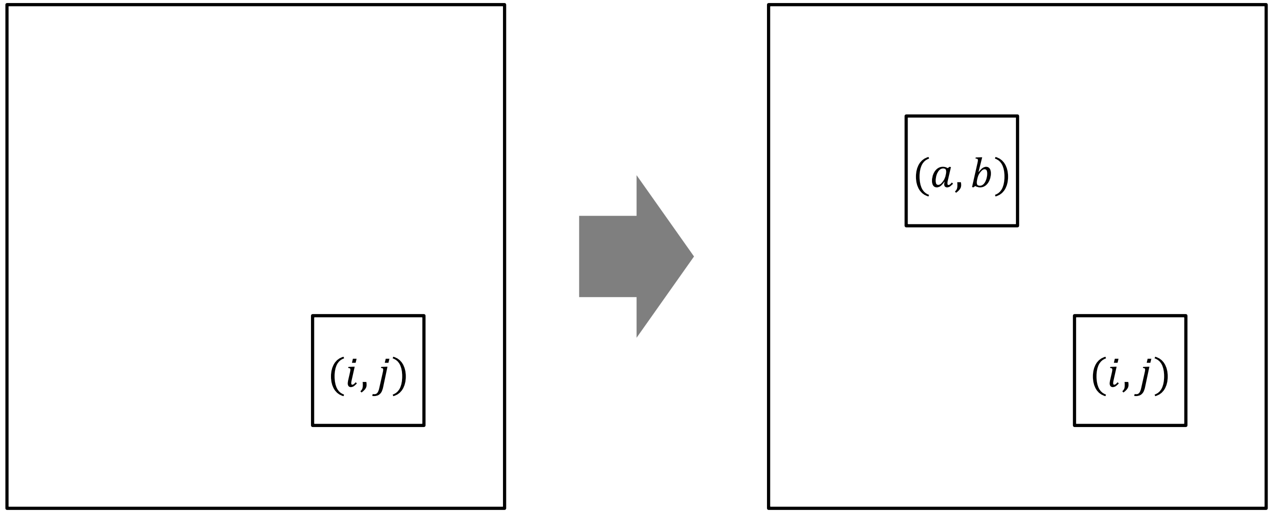
로 정의하면,

$$\sum_{\text{모든 직사각형}} (c_2)^2 = \sum_{i,j} \mathrm{Point}(i,j)$$

가 성립합니다. $\mathrm{Point}(i,j)$를 모든 (i,j)에 대해 구하면 문제가 해결됩니다.

J. 도박과 사각형

$\mathrm{Point}(i,j)$의 변화에 대해 생각합니다. 물론 (i,j)에는 2가 적혀 있다고 합시다. $1 \le a \le i, 1 \le b \le j$인 어떤 (a,b) 격자에 갑자기 2가 적혔을 때, $\mathrm{Point}(i,j)$는 얼마나 변할까요?

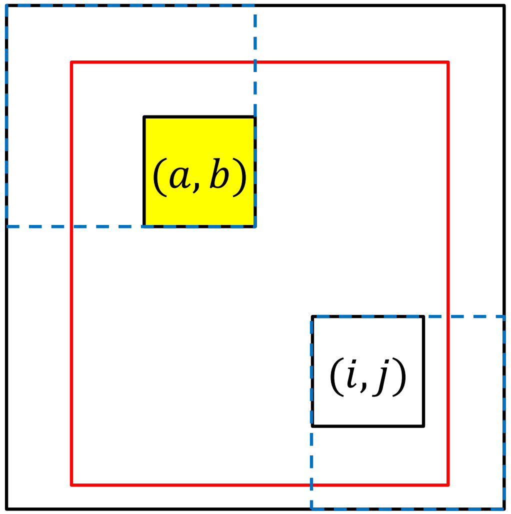


J. 도박과 사각형

Point(i, j)의 정의를 따라서, (a, b)와 (i, j)를 포함하는 직사각형들은 2의 빈도수가 1만큼 증가합니다. 그러한 직사각형의 수는

$$a \times b \times (n - i + 1) \times (n - j + 1)$$

입니다. 따라서 Point(i, j)의 변화량이 위와 같다는 것을 알 수 있습니다.

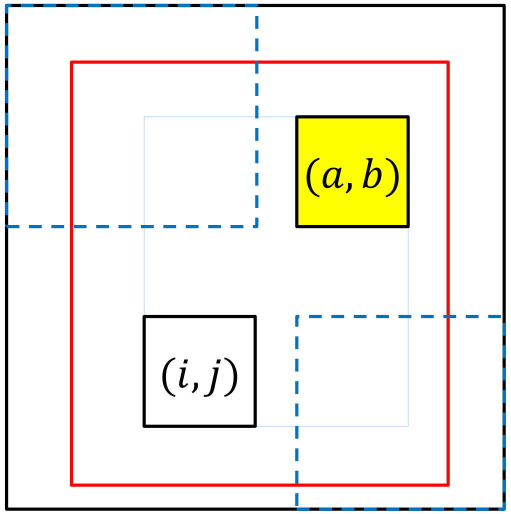


J. 도박과 사각형

같은 원리로, $1 \le a \le i, j \le b \le n$인 어떤 (a, b) 격자에 2가 적혔을 때, Point(i, j)의 변화량은

$$a \times j \times (n - i + 1) \times (n - b + 1)$$

입니다.

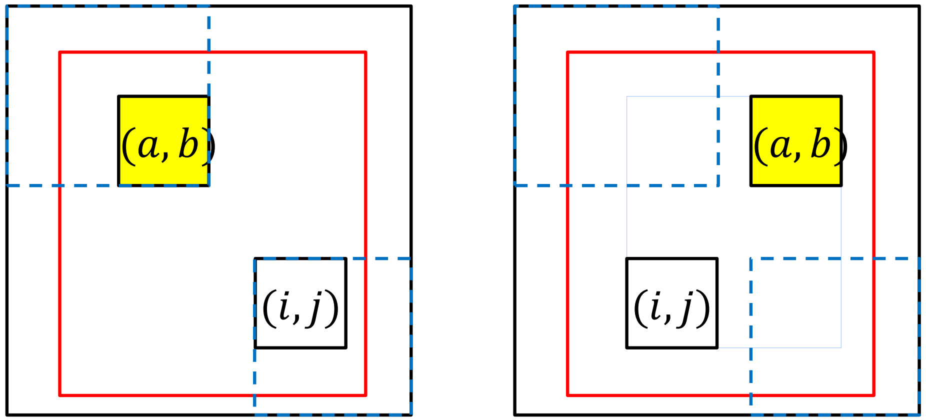


J. 도박과 사각형

이를 이용해서, $\mathrm{Point}(i,j)$의 식을 써 보면

$$\mathrm{Point}(i,j) = (n-i+1)\times(n-j+1)\times \sum_{\substack{1\le a\le i,1\le b\le j\\(a,b):\,2}} a\times b$$

$$+\,(n-i+1)\times j\times \sum_{\substack{1\le a\le i,j<b\le n\\(a,b):\,2}} a\times(n-b+1)$$



J. 도박과 사각형

이를 이용해서, $\mathrm{Point}(i,j)$의 식을 써 보면

$$\mathrm{Point}(i,j) = (n-i+1)\times(n-j+1)\times \boxed{\sum_{\substack{1\le a\le i,1\le b\le j \\ (a,b):\,2}} a\times b}$$

$$+ (n-i+1)\times j\times \boxed{\sum_{\substack{1\le a\le i,j<b\le n \\ (a,b):\,2}} a\times(n-b+1)}$$

2차원 부분합을 활용하여
미리 구해놓으면,
전처리 $O(n^2)$, 값을 구하는 데에 $O(1)$.

K. 특수부대 CH

- 맞은 팀 수 : 3
- 제출 횟수 : 96
- 정답률: 3.226%
- 처음 맞은 팀: 넥슨은 다람쥐를 뿌려라 (ch_434, ntopia, unused, kaze) , 206분
- 출제자 : tae(류현종)
- 해설 작성자 : tae(류현종)

K. 특수부대 CH

		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
00:00	Yes																														
	No																														
00:30	Yes																														
	No																														
01:00	Yes																														
	No											1																			
01:30	Yes																														
	No																									1					
02:00	Yes																														
	No								1			1								1			1				1			1	1
02:30	Yes																														
	No		1	1	1	1		1			1	2		1		1				2								1		2	

K. 특수부대 CH

		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
02:30	Yes																														
	No		1	1	1	1		1			1	2		1		1				2								1		2	
03:00	Yes																											1			
	No	1			1			1													1										
03:30	Yes																														
	No			1										1		1					1	1						1			1
04:00	Yes																		1												
	No	1		1	1		2							1			2				1	3	1						1		
04:30	Yes															1															
	No						2	2			2	4	5	6								2			1	1		1	1	3	5

K. 특수부대 CH

동일한 상태가 유지되는 시간에 대한 이해

→ 가장 빠른 상태 변화까지의 일정 상태 시뮬레이션

K. 특수부대 CH

가장 많이 틀린 요소:

- 체력이 0이면 죽습니다. 음수가 아니어도 죽어요.
- 죽어 있는 동안은 총 안 맞아요.
- 수레와 내 이동방향이 같은 경우에는 못 만나기도 해요.
- 근데 만나기도 해요. 조건 잘 봐야..

K. 특수부대 CH

명확한 이벤트 상태 전이 정의가 가장 중요합니다.

Special thanks to

후원

NEXON GT

시스템 지원

STARTLINK

Special thanks to

출제

류현종(xhae)	류원하(beingryu)
박수찬(tncks0121)	이태현(etaehyun4)
구재현(koosaga)	윤지학(cubelover)
김찬민(kcm1700)	???(zigui)

스태프

김예린	박구삼
박서현	하승표

Coder'shigh

감사합니다!

내년에 봐요~