

UOSPC 2021 후기 및 해설

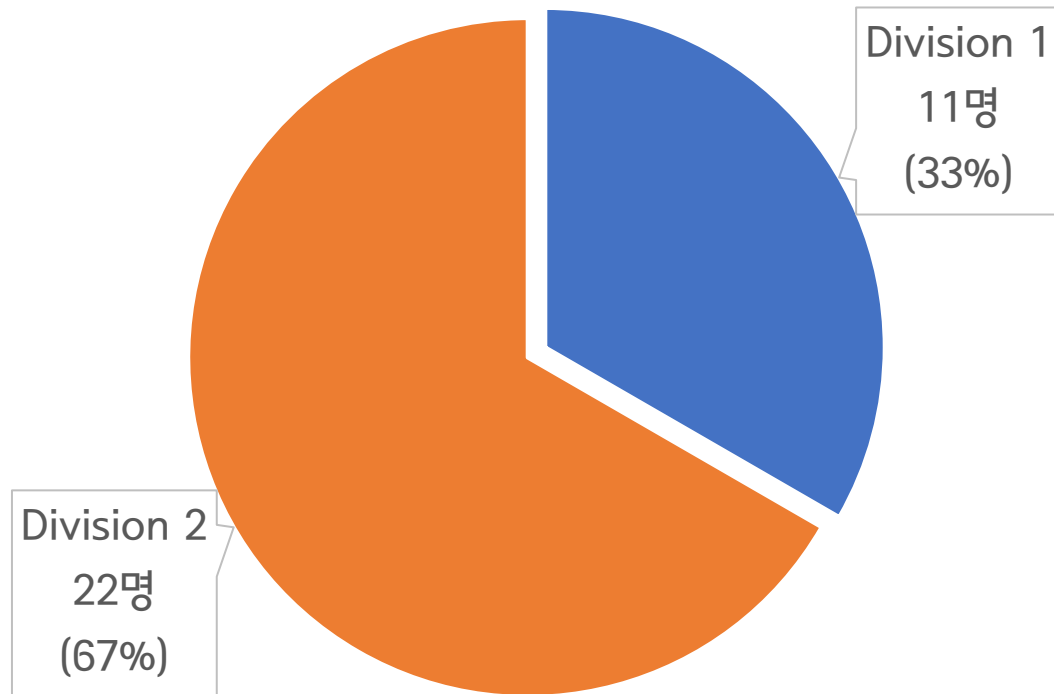
2021년 12월 4일

서울시립대 알고리즘 소모임 **신림**

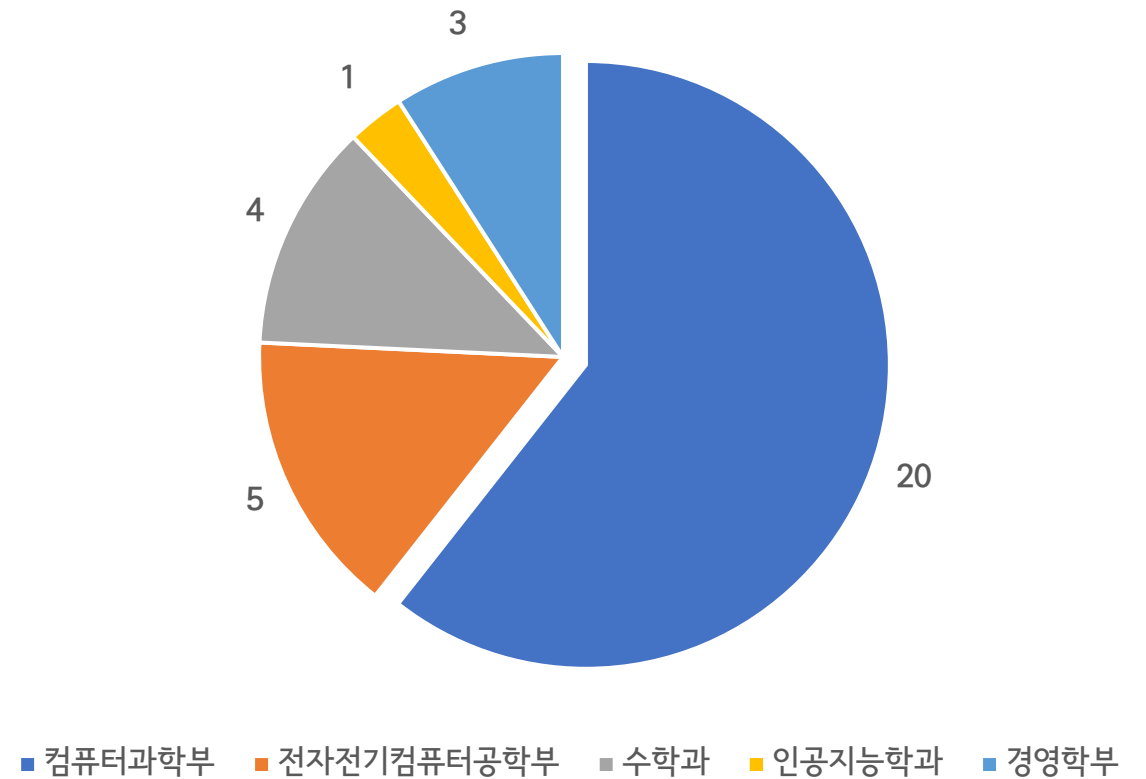
참가자 수 통계

■ ■ 참가자 수 통계

Division 별 참가자 수

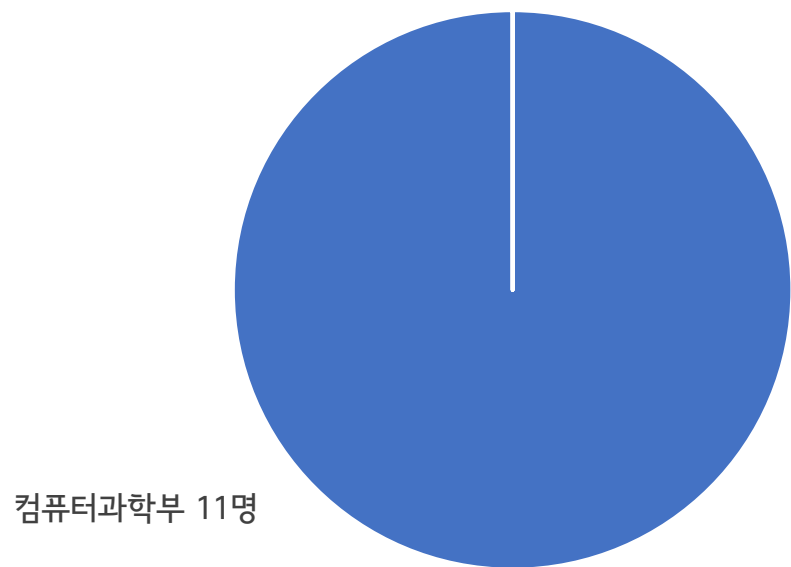


학과별 참가자 수

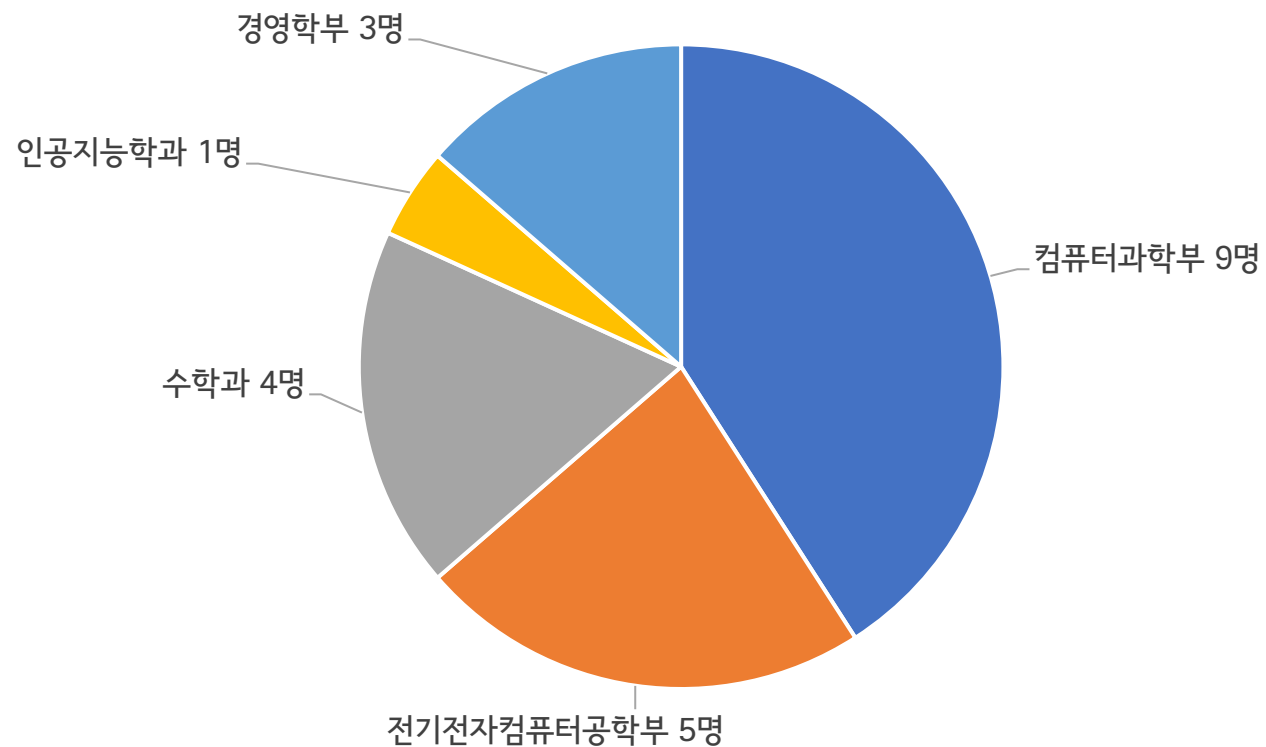


■ Division 별 통계

Division 1 참가자 수



Division 2 참가자 수



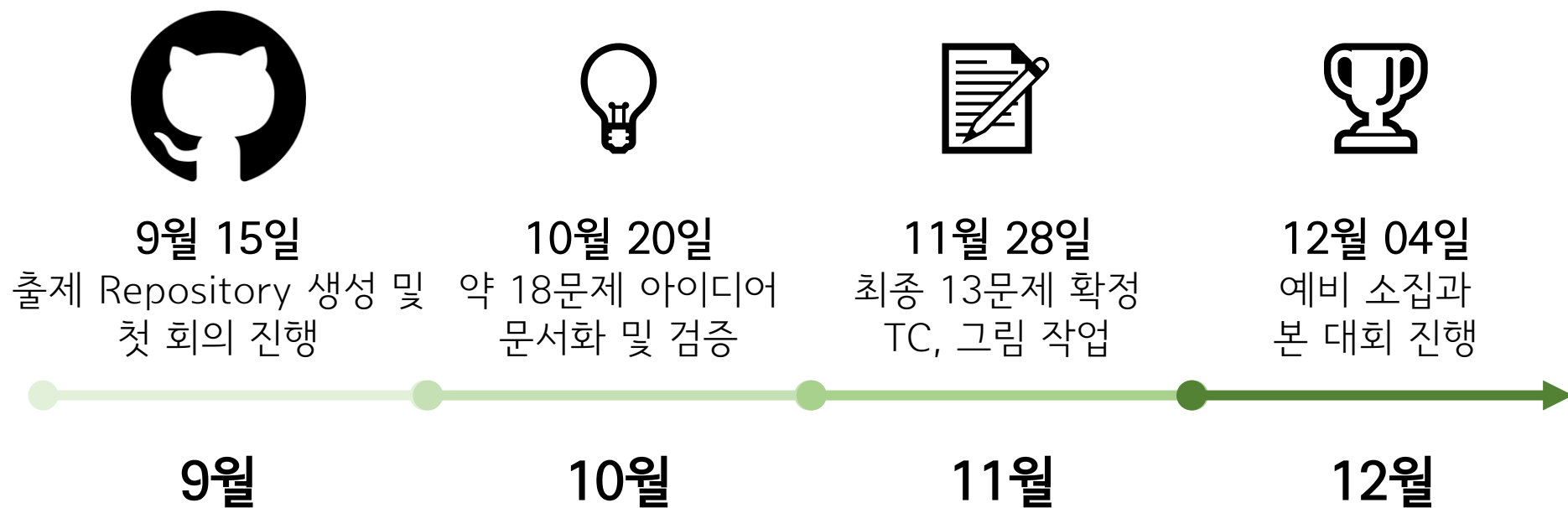
대회 준비 과정

■ 출제와 검수를 도와 주신 분들

총 5명의 출제자 분들과 3명의 검수자 분들께서 도와 주셨습니다!

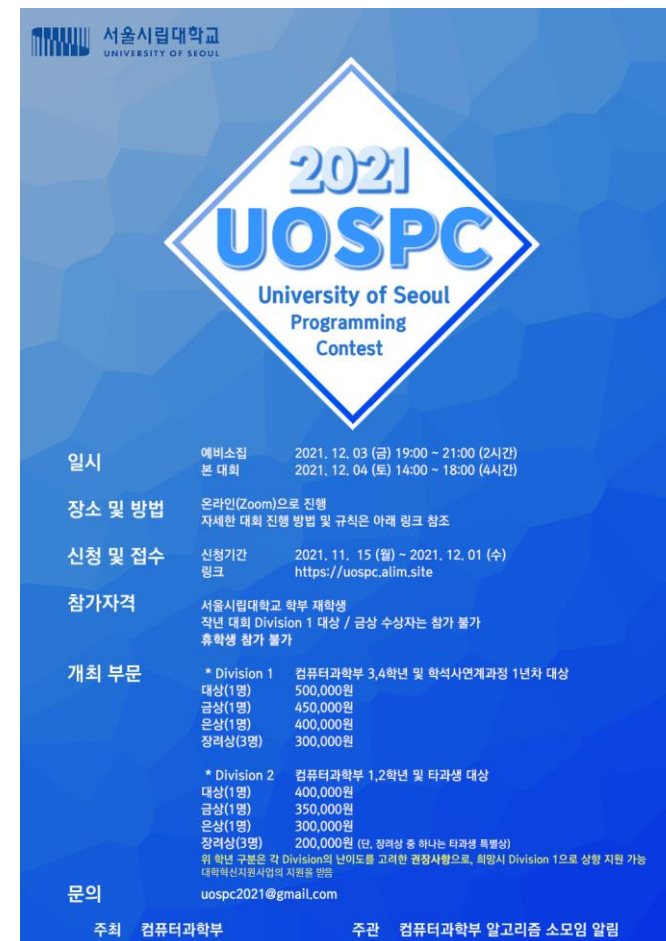
	이름	문제 출제	전체 문제 검수
출제자	김정현 (컴퓨터과학부)		✓
	오규민 (컴퓨터과학부)		✓
	이현석 (수학과)		✓
	정상윤 (경제학부)		✓
	최문기 (컴퓨터과학부)		✓
검수자	최연웅 (수학과)		✓
	정인우 (컴퓨터과학부)		✓
	주현도 (수학과)		✓

■ ■ 타임라인



Special Thanks

- 정상윤 님께서 포스터 작업을 맡아주셨습니다! ☺
- 컴퓨터과학부 김민호 교수님께서 대회 운영에 대한 값진 피드백을 주셨습니다!



서울시립대학교
UNIVERSITY OF SEOUL

2021 UOSPC
University of Seoul
Programming
Contest

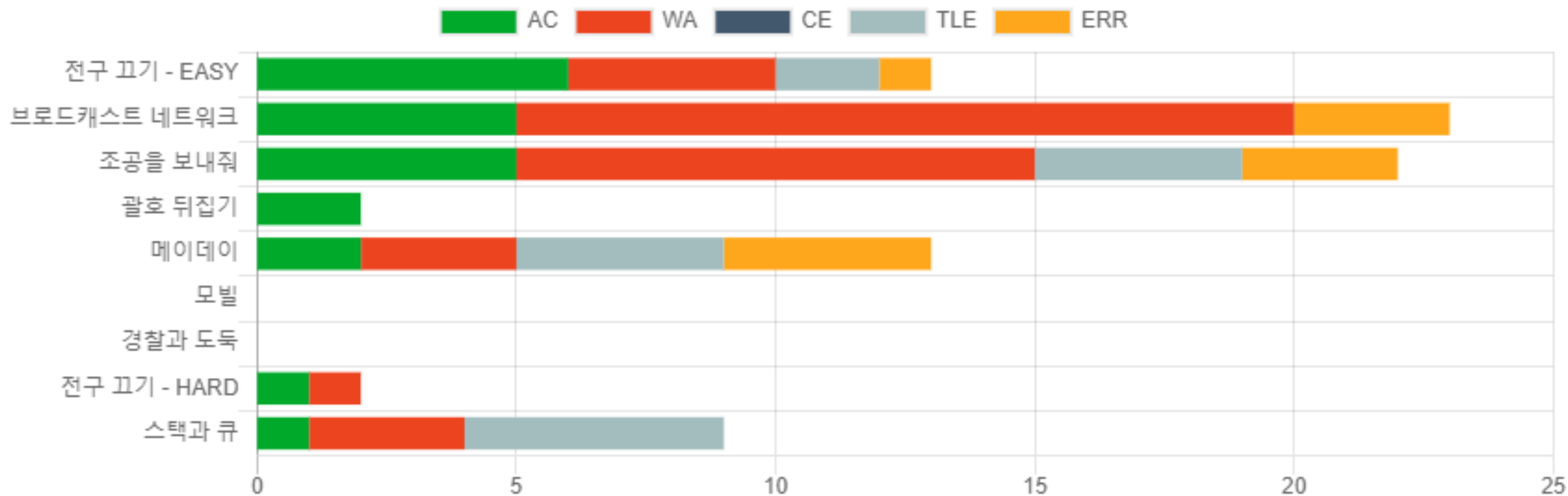
일시	예비소집 본 대회	2021. 12. 03 (금) 19:00 ~ 21:00 (2시간) 2021. 12. 04 (토) 14:00 ~ 18:00 (4시간)																
장소 및 방법	온라인(Zoom)으로 진행 자세한 대회 진행 방법 및 규칙은 아래 링크 참조																	
신청 및 접수	신청기간 링크	2021. 11. 15 (월) ~ 2021. 12. 01 (수) https://uospc.alim.site																
참가자격	서울시립대학교 학부 재학생 작년 대회 Division 1 대상 / 금상 수상자는 참가 불가 휴학생 참가 불가																	
개최 부문	<p>* Division 1 컴퓨터과학부 3,4학년 및 학석사연계과정 1년차 대상</p> <table><tbody><tr><td>대상(1명)</td><td>500,000원</td></tr><tr><td>금상(1명)</td><td>450,000원</td></tr><tr><td>은상(1명)</td><td>400,000원</td></tr><tr><td>장려상(3명)</td><td>300,000원</td></tr></tbody></table> <p>* Division 2 컴퓨터과학부 1,2학년 및 타과생 대상</p> <table><tbody><tr><td>대상(1명)</td><td>400,000원</td></tr><tr><td>금상(1명)</td><td>350,000원</td></tr><tr><td>은상(1명)</td><td>300,000원</td></tr><tr><td>장려상(3명)</td><td>200,000원 (단, 장려상 중 하나는 타과생 특별상)</td></tr></tbody></table> <p>위 학년 구분은 각 Division의 난이도를 고려한 권장사항으로, 희망시 Division 1으로 상향 지원 가능 대학혁신지원사업의 지원을 받음</p>		대상(1명)	500,000원	금상(1명)	450,000원	은상(1명)	400,000원	장려상(3명)	300,000원	대상(1명)	400,000원	금상(1명)	350,000원	은상(1명)	300,000원	장려상(3명)	200,000원 (단, 장려상 중 하나는 타과생 특별상)
대상(1명)	500,000원																	
금상(1명)	450,000원																	
은상(1명)	400,000원																	
장려상(3명)	300,000원																	
대상(1명)	400,000원																	
금상(1명)	350,000원																	
은상(1명)	300,000원																	
장려상(3명)	200,000원 (단, 장려상 중 하나는 타과생 특별상)																	
문의	uospc2021@gmail.com																	
주최	컴퓨터과학부	주관 컴퓨터과학부 알고리즘 소모임 알림																

대회 리뷰 및 총평

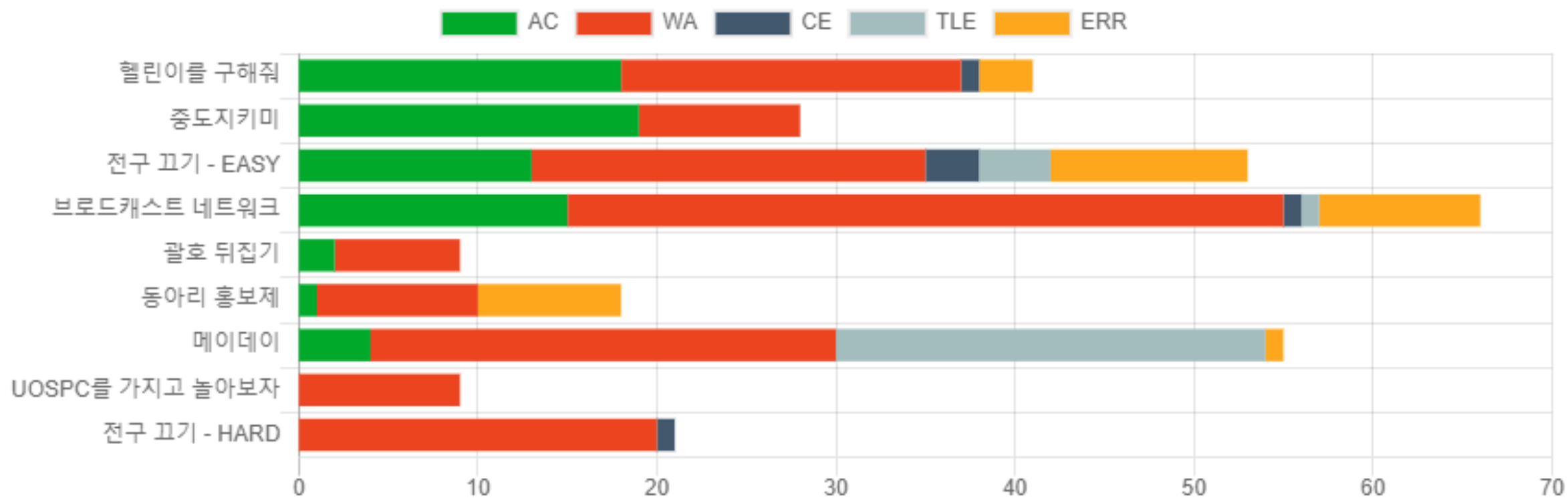
문제 목록

	Division 1		Division 2	
번호	문제 이름	의도한 난이도	문제 이름	의도한 난이도
A	전구 끄기 - EASY	Easy	헬린이를 구해줘	Easy
B	브로드캐스트 네트워크	Medium	중도지킴이	Easy
C	조공을 보내줘	Medium	전구 끄기 - EASY	Easy
D	괄호 뒤집기	Medium	브로드캐스트 네트워크	Medium
E	메이데이	Medium	괄호 뒤집기	Medium
F	모빌	Medium	동아리 홍보제	Medium
G	전구 끄기 - HARD	Hard	메이데이	Medium
H	경찰과 도둑	Hard	UOSPC를 가지고 놀아보자	Hard
I	스택과 큐	Hard	전구 끄기 - HARD	Hard

■ Division 1 문제 리뷰



■ Division 2 문제 리뷰



문제 풀이

출제자별 발표 순서

1. 김정현		
문제 이름	Division	의도한 난이도
브로드캐스트 네트워크	1, 2	Medium
UOSPC를 가지고 놀아보자	2	Hard

2. 이현석		
문제 이름	Division	의도한 난이도
헬린이를 구해줘	2	Easy
중도지키미	2	Easy
동아리 홍보제	2	Medium
조공을 보내줘	1	Medium

3. 정상윤		
문제 이름	Division	의도한 난이도
괄호 뒤집기	1, 2	Medium
경찰과 도둑	1	Hard

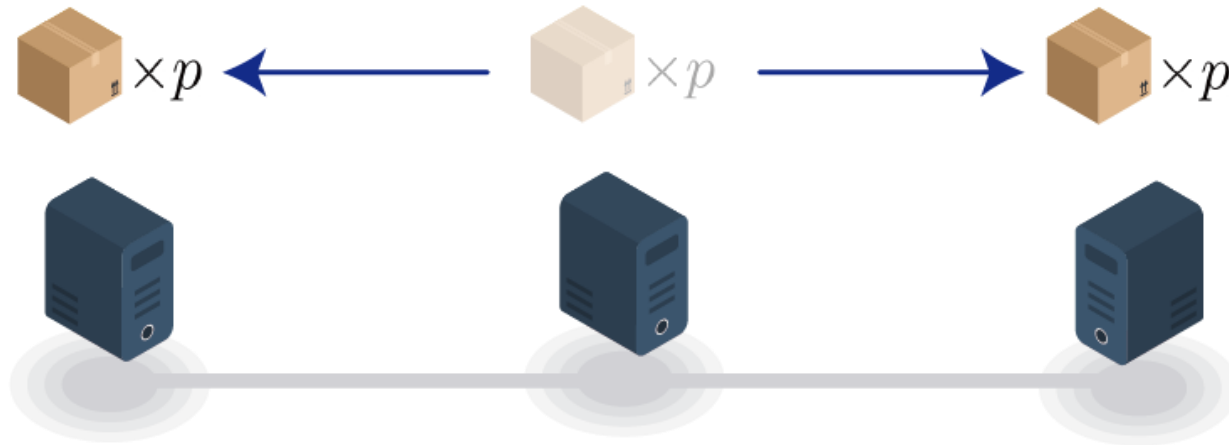
4. 오규민		
문제 이름	Division	의도한 난이도
전구 끄기 - EASY	1, 2	Easy
전구 끄기 - HARD	1, 2	Hard
메이데이	1, 2	Medium
모빌	1	Medium
스택과 큐	1	Hard

브로드캐스트 네트워크

- 종류 - Simulation
 - 출제진 의도 - **Medium**
-
- First solve
 - Division 1: 전예준 (Newbieeee)
 - Division 2: 신한범 (shb115)
 - 출제자: 김정현

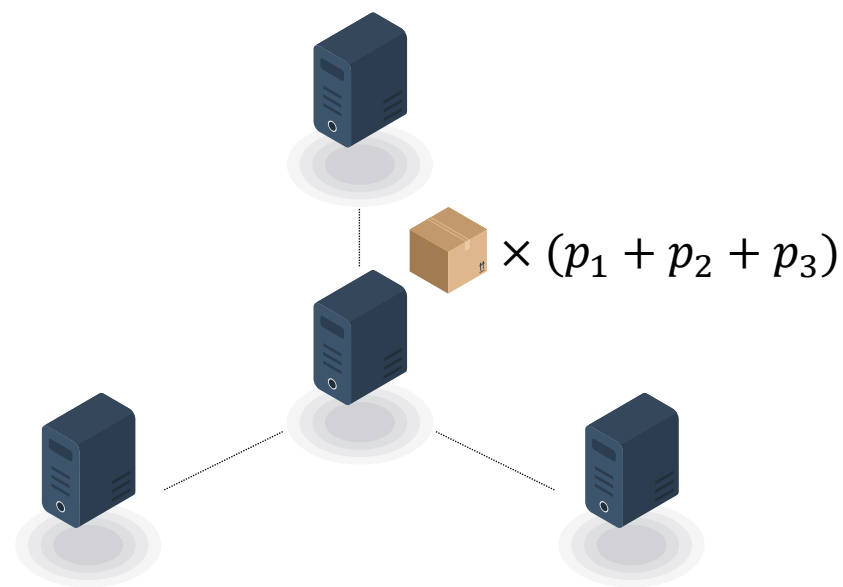
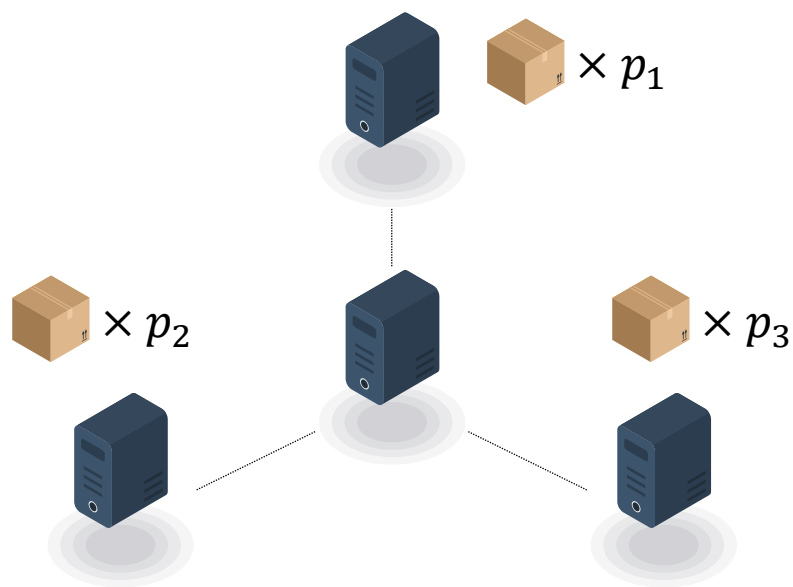
■ ■ 브로드캐스트 네트워크

- 간단한 시뮬레이션을 의도하고 출제한 문제입니다.
- 시간이 t 일 때의 패킷 수로부터 $t + 1$ 일 때의 패킷 수를 어떻게 구할 수 있을지 생각해봅시다.



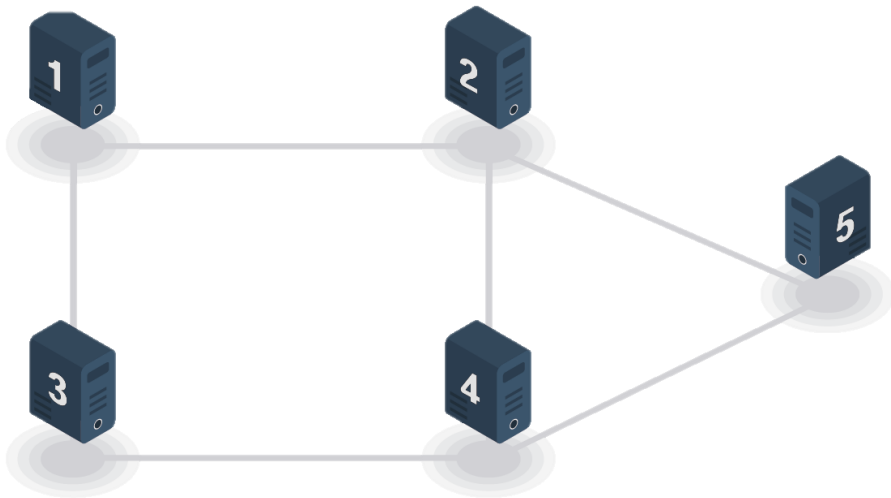
■ ■ 브로드캐스트 네트워크

- 특정 컴퓨터로 도착하는 패킷의 수는 이웃한 컴퓨터들이 보내는 패킷의 수를 합한 것과 같습니다.



■ ■ 브로드캐스트 네트워크

- 아래 예제에서 t 시간에 i 번째 컴퓨터에 존재하는 패킷의 수를 $p(t, i)$ 라고 두면, t 에 대한 점화식을 세울 수 있습니다.



$$p(t + 1, 1) = p(t, 2) + p(t, 3)$$

$$p(t + 1, 2) = p(t, 1) + p(t, 4) + p(t, 5)$$

$$p(t + 1, 3) = p(t, 1) + p(t, 4)$$

$$p(t + 1, 4) = p(t, 2) + p(t, 3) + p(t, 5)$$

$$p(t + 1, 5) = p(t, 2) + p(t, 4)$$

■ ■ 브로드캐스트 네트워크

- $t = 0$ 일 때부터 입력으로 주어지는 시간까지 점화식을 전개하면 되므로, $O(n^2t)$ 의 시간복잡도로 해결할 수 있습니다.

$$p(0, 1) = 1$$

$$p(0, 2) = 0$$

$$p(0, 3) = 0$$

$$p(0, 4) = 0$$

$$p(0, 5) = 0$$

$$p(t + 1, 1) = p(t, 2) + p(t, 3)$$

$$p(t + 1, 2) = p(t, 1) + p(t, 4) + p(t, 5)$$

$$p(t + 1, 3) = p(t, 1) + p(t, 4)$$

$$p(t + 1, 4) = p(t, 2) + p(t, 3) + p(t, 5)$$

$$p(t + 1, 5) = p(t, 2) + p(t, 4)$$

■ ■ 브로드캐스트 네트워크

- 이 문제는 쉬운 문제로 출제되었으나, 분할정복과 선형대수를 이용하면 $O(n^3 \log T)$ 로도 해결할 수 있습니다.
- 더 어려운 문제를 원하신다면 한번쯤 도전 해보시기 바랍니다!

$$\begin{pmatrix} p(t+1, 1) \\ p(t+1, 2) \\ p(t+1, 3) \\ p(t+1, 4) \\ p(t+1, 5) \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} p(t, 1) \\ p(t, 2) \\ p(t, 3) \\ p(t, 4) \\ p(t, 5) \end{pmatrix}$$

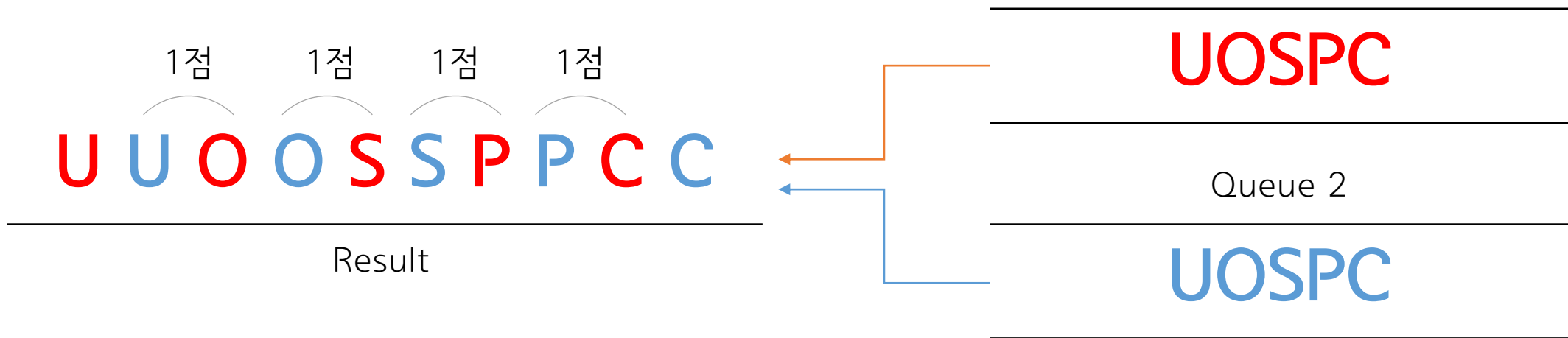
$$P_T = A^T P_0$$

UOSPC를 가지고 놀아보자

- 종류 - Dynamic Programming
 - 출제진 의도 - Hard
-
- First Solve: -
 - 출제자: 김정현

UOSPC를 가지고 놀아보자

- 두 개의 큐를 어떤 순서로 사용해야 최적의 문자열을 얻을 수 있을까요?
- 두 큐를 어떤 순서로 사용할지를 생각하는 것은 조금 어려워 보입니다.



■ UOSPC를 가지고 놀아보자

- 모든 경우의 수를 탐색한다면 어떨까요?
 - 두 큐 각각에 문자가 n_1 개, n_2 개 있으므로, 큐에서 문자를 $n_1 + n_2$ 번 뽑아야 합니다.
 - $n_1 + n_2$ 번 중 n_1 번은 첫 번째 큐를 택할 것이므로 큐를 택하는 경우의 수는 $n_1 + n_2$ 개에서 n_1 개를 택하는 조합의 수와 같습니다.
- 모든 경우의 수를 탐색하는 것은 너무 오래 걸릴 듯 합니다.

UOSPC를 가지고 놀아보자

- 문자의 가짓수가 단 5개 **뿐**이라는 점을 이용해, DP(Dynamic Programming)을 적용해보겠습니다.
- 현재 두 큐에서 각각 i, j 번째 문자가 Front에 있고, 만들어진 문자열의 끝 문자가 X 일 때, 만들 수 있는 최소 점수를 $f(i, j, X)$ 라고 하겠습니다.



UOSPC를 가지고 놀아보자

- $f(i, j, X)$ 의 형태가 복잡해 보이지만, 쉽게 점화식을 세울 수 있습니다.

① Q_1 에서 문자를 뽑아 X 뒤에 붙인다면?

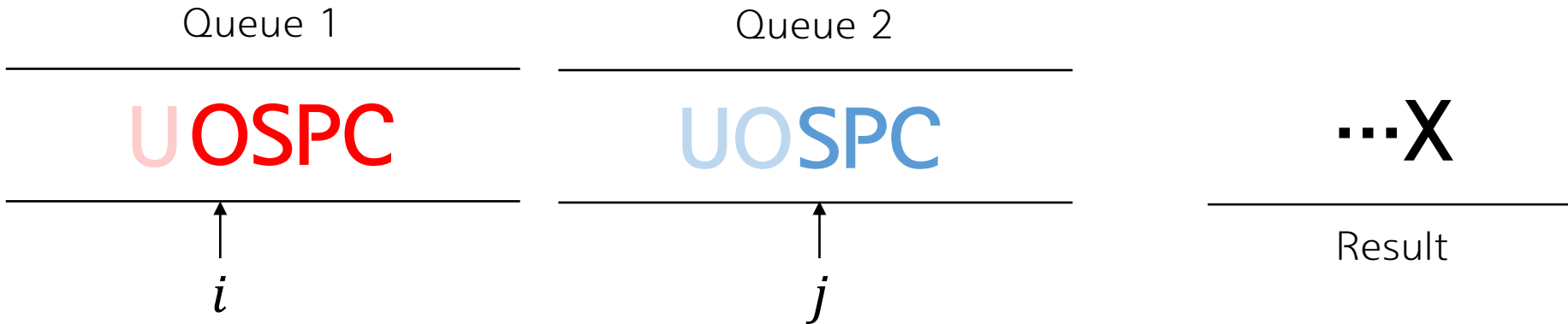
$$\begin{aligned} & (Q_1[i] \text{가 } X \text{와 다르다면 1점}) \\ & \quad + \\ & f(i + 1, j, Q_1[i]) \end{aligned}$$

② Q_2 에서 문자를 뽑아 X 뒤에 붙인다면?

$$\begin{aligned} & (Q_2[j] \text{가 } X \text{와 다르다면 1점}) \\ & \quad + \\ & f(i, j + 1, Q_2[j]) \end{aligned}$$

UOSPC를 가지고 놀아보자

- 두 큐에 있는 문자의 수가 각각 최대 2,000개이고, 문자는 5가지가 있으므로, $f(i, j, X)$ 의 인자의 개수는 2×10^7 개입니다.
- 이는 일반적인 컴퓨터에서 1초 내에 충분히 탐색해낼 수 있는 양입니다.
- 따라서 최종 시간 복잡도는 $O(n_1 n_2)$ 입니다.



헬린이를 구해줘

- 종류 - 구현
- 출제진 의도 - **Easy**

- First Solve: 이명규 (kaki1013)
- 출제자: 이현석

■ 헬린이를 구해줘

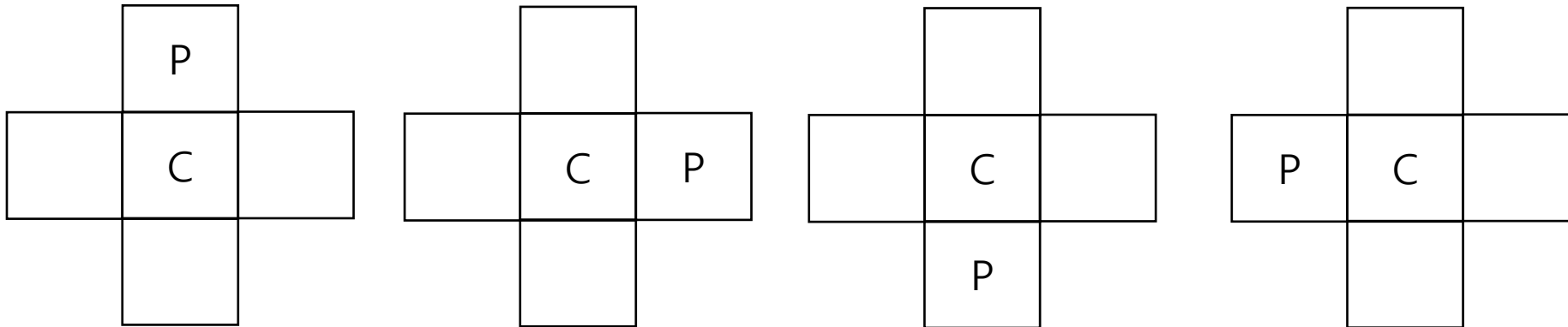
- 단순 구현 문제입니다.
- ‘모든 케이스에서 답을 구할 수 있다.’ → 적어도 2개의 값은 정상으로 입력된다.
- ‘-1’이 아닌 2개의 정상 입력을 찾고 등차수열의 첫 항과 공차를 구할 수 있습니다.
- i, j 번째 정상 입력을 찾으면 공차를 $(A_j - A_i) / (j - i)$ 을 통해 공차를 구할 수 있습니다.
($i \neq j, i < j$)
- 시간 복잡도 : $O(N)$

중도지키미

- 종류 - 구현
 - 출제진 의도 - **Easy**
-
- First Solve: 이상민 (poiu694)
 - 출제자: 이현석

중도지키미

- 단순 구현 문제입니다.
- 사람이 앉을 수 있는 빈 자리는 'C'로 입력됩니다.
- 빈 자리 'C' 상하좌우에 사람이 앉아있는 자리 'P' 가 없다면 앉을 수 있는 자리입니다.



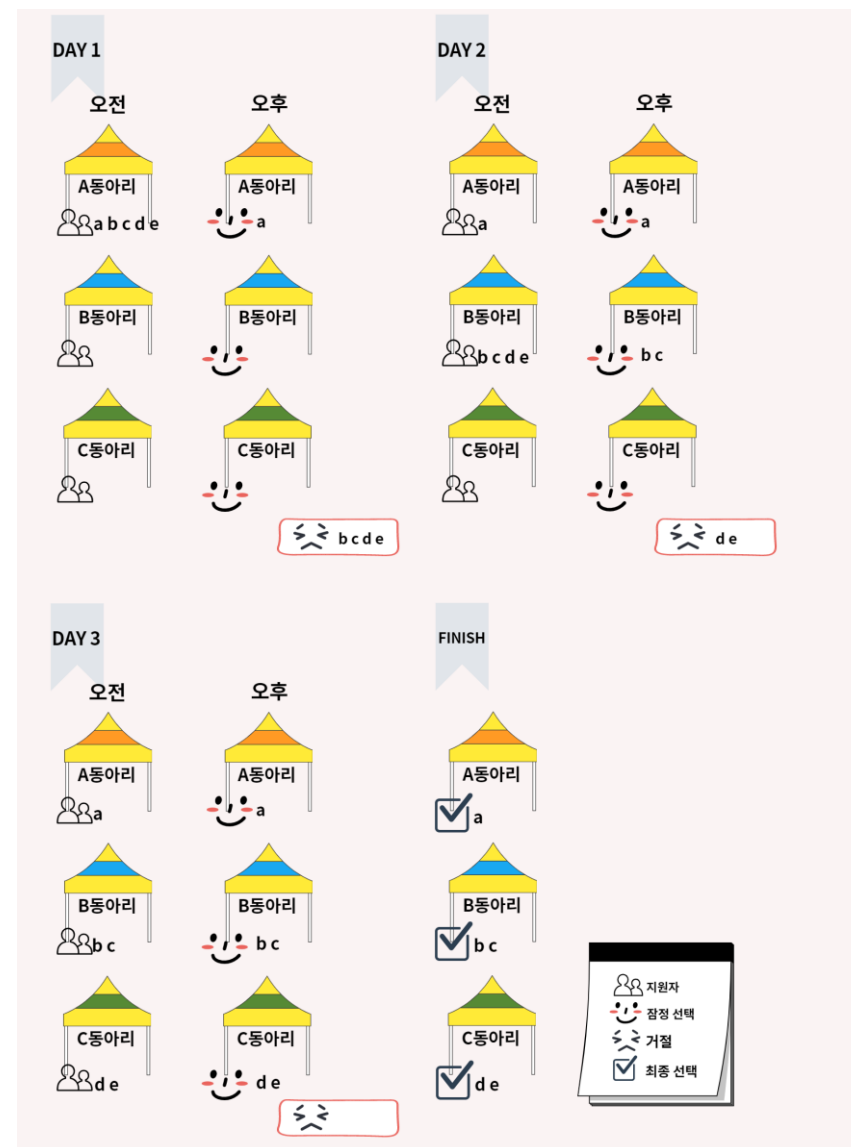
- 위의 4가지 경우를 제외한 모든 빈 자리는 앉을 수 있는 자리입니다.
- 시간 복잡도 : $O(N*M)$

동아리홍보제

- 종류 - 구현
 - 출제진 의도 - Medium
-
- First Solve: 김민호 (vmfoslkim)
 - 출제자: 이현석

동아리홍보제

- 구현 문제입니다.
- T일 동안 다음을 반복합니다.
 - 잠정 선택되지 않은 지원자를, 우선순위를 고려하여 각 동아리별로 배정한 목록을 구합니다.
 - 각 동아리별로 지원자 중 우선순위가 높은 지원자를 잠정 선택하고, 낮은 지원자를 탈락시킵니다.
- 시간 복잡도 : $O(TMN)$



조공을 보내줘

- 종류 - 자료구조(우선순위큐)
 - 출제진 의도 - Medium
-
- First Solve: 유시온 (kir3i)
 - 출제자: 이현석

■ 조공을 보내줘

- 자료구조 우선순위 큐를 이용해 해결할 수 있습니다.
 - 1) K도시에서 필요한 정보는 1번 도시부터 K-1번 도시까지 문이 열려있는지 정보
→ 1번 도시부터 문이 열려있는 연속된 도시 정보가 필요합니다.
 - 2) 우선순위 큐를 이용해 현재 조공을 보내는(문이 열린) 도시 번호를 저장합니다.
이때, 낮은 번호에게 높은 우선 순위를 부여합니다.
즉, 우선순위 큐를 이용해 열려있는 도시 중 가장 작은 도시 번호를 참조할 수 있습니다.

조공을 보내줘

- 자료구조 우선순위 큐를 이용해 해결할 수 있습니다. ($O(n \log n)$)

3) 우선순위 큐를 이용해 1번 도시부터 열려있는 연속된 도시 정보를 갱신합니다.

Step 1) 우선순위 큐에 현재 조공을 보내는 도시 정보를 입력합니다.

Step 2) 1번 도시부터 문이 열려있는 연속된 도시 + 1 == 우선순위 큐.top()

→ 연속된 도시 정보 갱신 && 우선순위 큐.pop()

```
for (int i = 0; i < N; i++) {
    pq.push(-list[i]);
    while(!pq.empty() && openContiCityNum + 1 == -pq.top()){
        openContiCityNum++;
        pq.pop();
        if(openContiCityNum == list[i]) ans++;
    }
}
```

괄호 뒤집기

- 종류 - Greedy Algorithm
 - 출제진 의도 - **Medium**
-
- First Solve
 - Division 1: 유시온 (kir3i)
 - Division 2: 박성우 (ece2017440048)
 - 출제자: 정상윤

■ ■ 괄호 뒤집기

- Greedy Algorithm(탐욕법)으로 풀 수 있는 문제입니다.
- 먼저 괄호를 최소한으로 뒤집는 경우를 살펴봅시다.
- ‘(’ → ‘1’ ‘)’ → ‘-1’로 각각 변환해 수열을 만듭니다.
- Ex) ‘()) (((’이라는 문자열이 주어졌다면 [1,-1,-1,1,1,1]로 변환
- 이제 수열의 누적 합을 구합니다. [1,0,-1,0,1,2]
- 누적 합이 음수가 되는 구간의 괄호는 반드시 뒤집혀야 합니다.
→ 여는 괄호보다 닫는 괄호가 더 많이 존재하기 때문입니다.

■ ■ 괄호 뒤집기

- 누적 합이 -1 이 되는 순간마다 뒤집어 주었다면,
- 누적 합을 구한 수열의 마지막 index는 0 이상의 짝수인 정수 $2k$ 가 될 것입니다.
- 이는 ‘(‘가 ‘)’보다 $2k$ 만큼 많이 존재한다는 뜻입니다.
- 문자열의 뒤에서부터 k 개의 ‘(’를 ‘)’로 뒤집어 주면 됩니다.
- 이 때 뒤집은 총 괄호의 수는, 누적 합이 -1 이 되는 모든 구간의 수 + k 이며, 이 경우가 최솟값이 되는 것은 자명합니다.

■ ■ 괄호 뒤집기

- 그렇다면 **최댓값**은 어떻게 구해야 할까요?
- 방법은 생각보다 간단합니다, **모든 괄호를 뒤집**은 후, 뒤집힌 문자열에서 **최솟값**을 구하면 됩니다!!

‘(((())))’ → ‘)))) ((((‘ → ‘() () (())’

[**최댓값** : N - 뒤집은 문자열에서의 최솟값!]
시간복잡도는 선형 탐색만 진행하기 때문에 **$O(N)$** 입니다.

경찰과 도둑

- 종류 - Graph traversal, DFS, BFS, Game Theory
 - 출제진 의도 - Hard
-
- First Solve: -
 - 출제자: 정상윤

경찰과 도둑

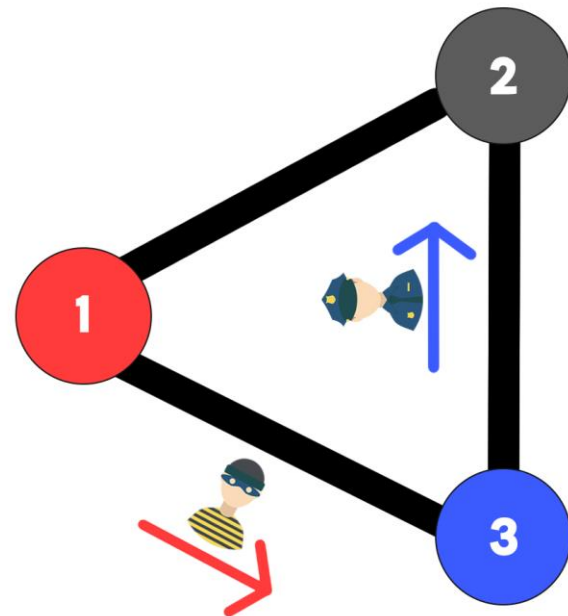
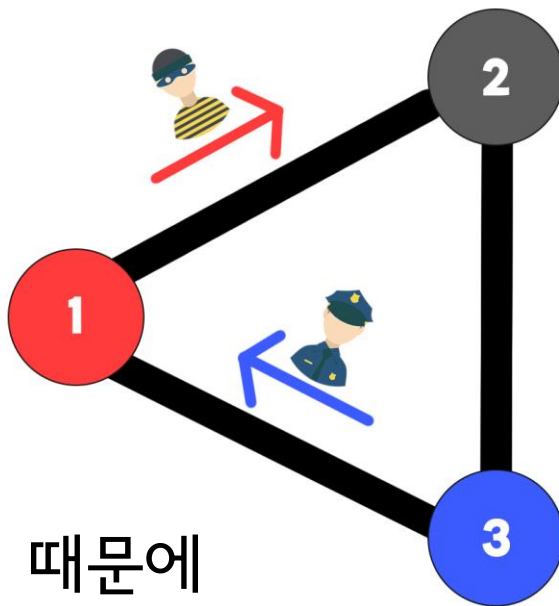
- 경찰이 도둑을 영원히 잡을 수 없는 조건?

경찰과 도둑이 같은 **cycle** 위에 존재!

도둑이 경찰이 움직인 방향으로 움직인다면
영원히 거리를 좁힐 수 없습니다!

문제에서 **n**개의 정점과 도로가 주어졌기 때문에
반드시 **1**개의 **cycle**이 생기게 됩니다

→ Tree graph에 1개의 간선이 추가되어 사이클이 1개 생깁니다



경찰과 도둑

Step 1) 사이클 판별

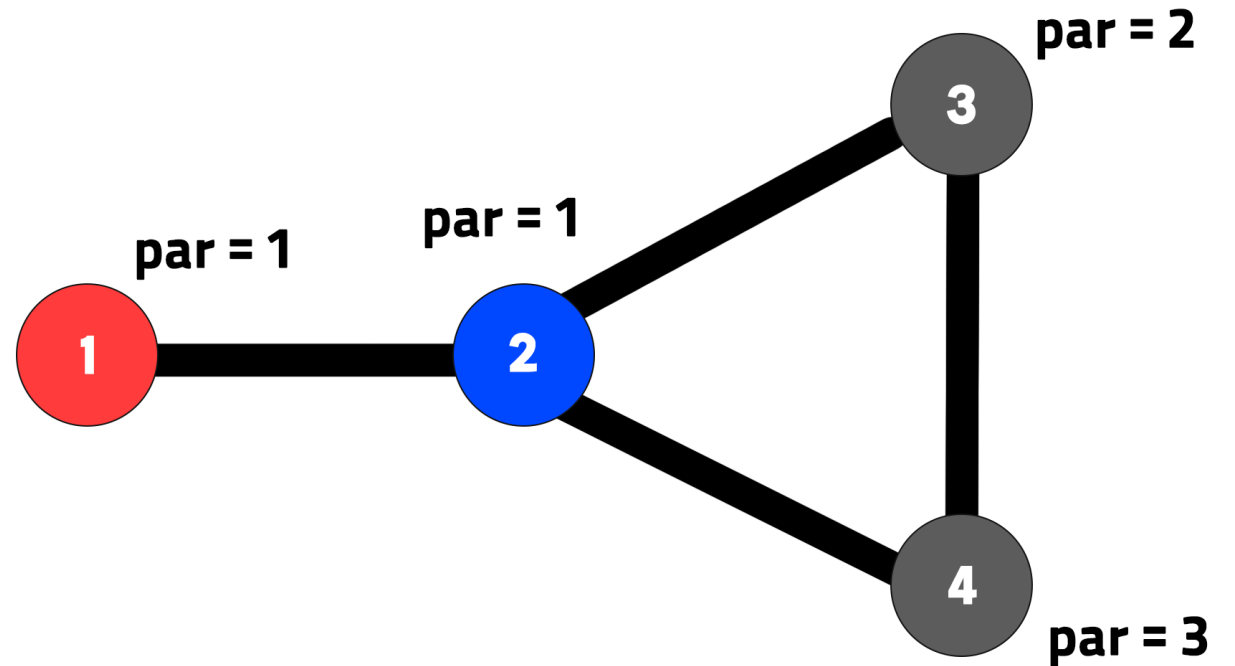
- 주어진 그래프에서 사이클을 이루는 모든 정점을 찾아야 합니다.

Dfs(depth first search):

1번 정점부터 깊이 우선 탐색으로
각 정점의 부모 정점을 찾습니다!

정점 방문 순서 : 1 - 2 - 3 - 4 - 2

par = [0,1,1,2,3] # 부모 배열



경찰과 도둑

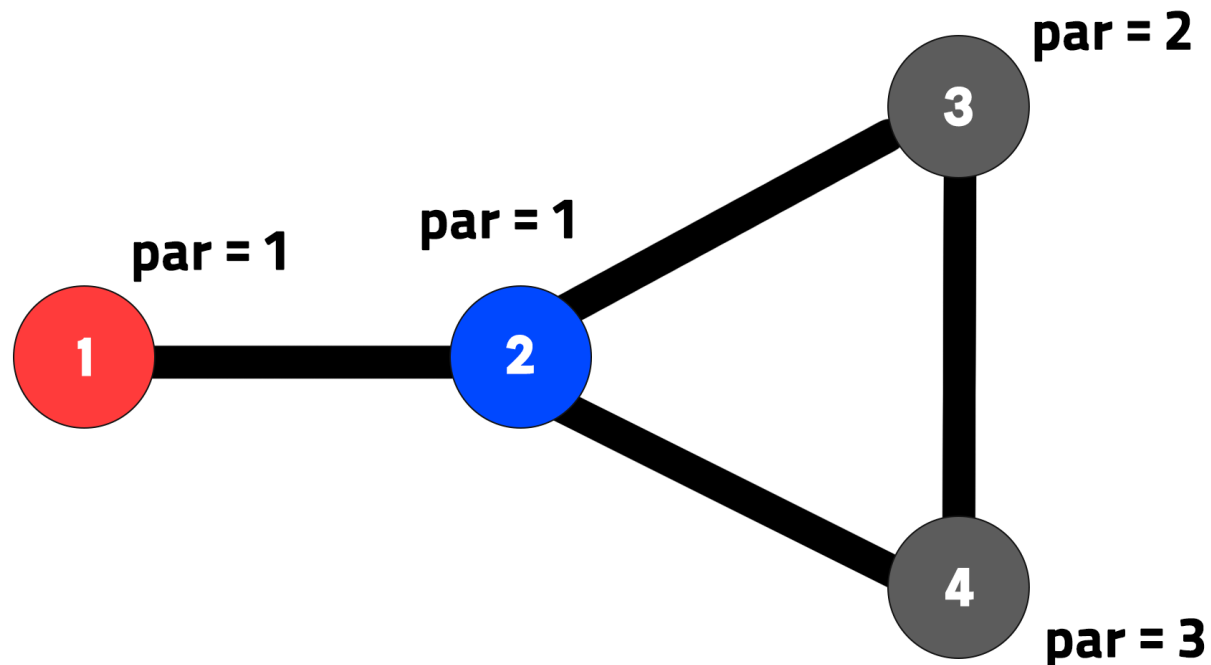
Step 1) 사이클 판별

정점 방문 순서 : 1 - 2 - 3 - 4 - 2

이미 방문한 정점인 2를 다시 방문,

정점 2와 4가 사이클을 이루는 것을 확인합니다.

이제 다시 방문한 정점(2)를 정점 **i**, 정점 **i**로 이어지는 정점(4)을 정점 **j**로 두고,
나머지 정점들을 탐색해 봅시다.



■ 경찰과 도둑

Step 1) 사이클 판별

부모 배열 $par = [0, 1, 1, 2, 3]$ 을 활용해서 사이클을 이루는 정점을 모두 찾을 수 있습니다.

$par[j] = k_1, par[k_1] = k_2, par[k_2] = k_3 \dots, par[k_m] = i$ 일 때,
 $[k_1, k_2, \dots, k_m]$ 번 정점들은 모두 사이클을 이루고 있습니다.

수식으로 쓰면 복잡하지만, 백트래킹을 활용하면 쉽게 구할 수 있는 부분입니다.

■ 경찰과 도둑

Step 2) 사이클까지의 거리 확인

경찰은 반드시 사이클을 이루는 모든 정점에

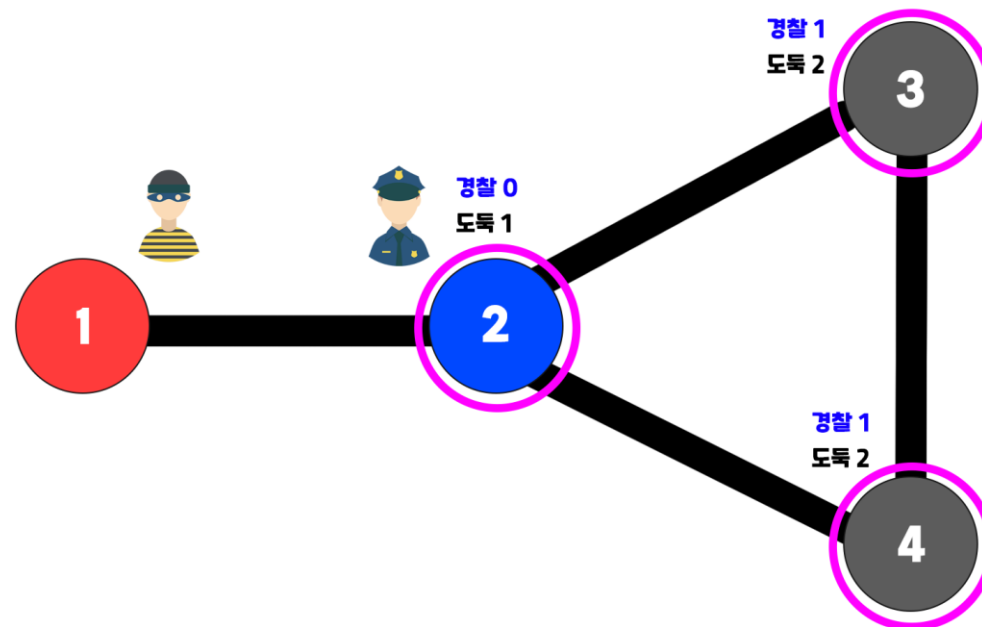
도둑보다 먼저 도착해야만 도둑을 검거할 수 있습니다!!!

거리 확인을 위해 BFS(Breadth First Search)를 활용할 수 있습니다

경찰과 도둑

Step 2) 사이클까지의 거리 확인

1. 경찰과 도둑의 최초 정점에서 다른 정점까지의 거리 BFS로 확인
2. 사이클을 이루는 정점들에서 경찰과 도둑의 거리를 각각 비교
3. 도둑이 하나라도 가까우면 검거 X
4. 시간복잡도는 $O(N)$ 입니다

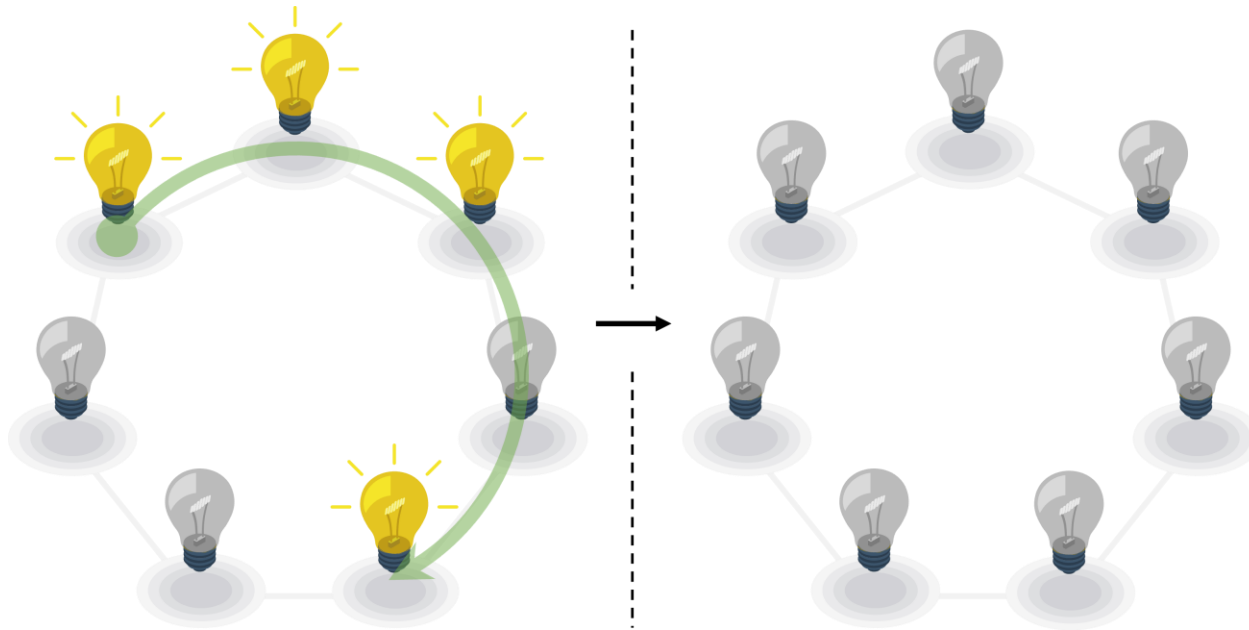


전구 끄기 - EASY

- 종류 - Greedy Algorithm
 - 출제진 의도 - Easy
-
- First Solve
 - Division 1: 유시온 (kir3i)
 - Division 2: 서명교 (devouring123)
 - 출제자: 오규민

■ 전구 끄기 - EASY

- 원순열의 일부를 단 한번 돌면서 모든 전구를 꺼야 합니다.
- 따라서 순회하지 않는 일부 구간에 모두 꺼진 전구만 있다면 한번 순회하는 것으로 모든 전구를 끌 수 있습니다.



■ 전구 끄기 - EASY

- 시작점과 끝점은 켜진 전구로 하고, 그 사이에 있는 전구들은 모두 꺼져 있는 순회 경로들을 모두 고려하면 문제를 풀 수 있습니다.



■ 전구 끄기 - EASY

- 켜진 전구가 k 개 있다면 고려해야 하는 경로의 수는 $k + 1$ 개이고, 이는 단순히 전구 배열을 for 루프로 순회하는 것으로 고려할 수 있습니다.
- 따라서, 최종 시간 복잡도는 $O(n)$ 입니다.



전구 끄기 - HARD

- 종류 - 최대 연속 부분합
 - 출제진 의도 - Hard
-
- First Solve
 - Division 1: 유시온 (kir3i)
 - Division 2: -
 - 출제자: 오규민

■ 전구 끄기 - HARD

$$\{1, 1, 0, 1, 0, 0, 1\} \longrightarrow \{1, 1, -1, 1, -1, -1, 1\}$$

- 켜진 전구를 끄면 꺼진 전구가 하나 증가합니다.(+1)
- 꺼진 전구를 키면 꺼진 전구가 하나 감소합니다.(-1)
- 꺼진 전구의 증감량을 나타낸 배열은 오른쪽과 같습니다.

■ 전구 끄기 - HARD

$\{1, 1, -1, 1, -1, -1, 1\}$

- 꺼진 전구의 최대 증가량을 구해봅시다.
- 이는 위 배열에서 **최대 연속 부분합**을 구하는 것과 같을 것입니다.
- 최대 연속 부분합은 동적계획법, 분할정복 등을 통해서 구할 수 있습니다.

■ 전구 끄기 - HARD

$\{1, 1, -1, 1, -1, -1, 1\}$

- 하지만 이 문제에서 전구는 원형으로 나열되어있기 때문에 위와 같이 배열의 양 끝을 선택한 것이 답이 될 수 있습니다.

■ 전구 끄기 - HARD

$\{1, 1, -1, 1, -1, -1, 1\}$

$\{1, 1, -1, 1, -1, -1, 1\}$

- 이 경우는 전체 배열에 합에서 **최소 연속 부분합**을 구해서 빼면 구할 수 있습니다.
- 최소 연속 부분합 또한 최대 연속 부분합과 같은 방식으로 구할 수 있습니다.

전구 끄기 - HARD

- 최적해가 배열의 중간인 경우: 최대 연속 부분합



- 최적해가 배열의 양 끝인 경우: 총 배열의 합 - 최소 연속 부분합



- 위 두 가지 값을 모두 구하고 max값을 구하면 최적해입니다.
- 총 시간 복잡도: $O(N)$** (동적계획법으로 최대 연속 부분합을 구한 경우)

메이데이

- 종류 - Math, Set
 - 출제진 의도 - **Medium**
-
- First Solve
 - Division 1: 전예준 (Newbieeee)
 - Division 2: 이명규 (kaki1013)
 - 출제자: 오규민

■ ■ 메이데이

- x축의 비행기와 y축의 비행기가 서로 충돌할 조건은 아래와 같습니다.

동일한 시간동안 이동했으므로 $s = v t$, $t = s/v$

$$d \frac{s_y}{v_x} = \frac{s_x}{v_y}, \quad s_y \times v_y = s_x \times v_x$$

해당 식 조건을 만족하는 경우 두 비행기는 충돌합니다.

■ ■ 메이데이

- naïve한 풀이:
x축 위의 어떤 비행기와 y축 위의 어떤 비행기가 충돌하는지 모두 확인하는 경우, 시간복잡도는 $O(N^2)$ 로 시간초과를 받습니다.
- 최적화된 풀이:
 $valuex = s_x \times v_x$ 값을 미리 저장해놓은 후, 해당 값이 같은 것의 쌍 ($valuex_i = valuey_j$) 의 개수를 세아리면 됩니다.

■ ■ 메이데이

- x 축 위의 어떤 비행기의 $value_x$ 값에 대해 동일한 값을 가진 $value_y$ 가 존재하는지 $O(\log N)$ 에 확인할 수 있으므로, 총 시간복잡도는 $O(N \log N)$ 이 됩니다.
- 구현은 set 또는 이분탐색을 통해 할 수 있습니다.

모빌

- 종류 - Tree DP, DFS, Number Theory
 - 출제진 의도 - **Medium**
-
- First Solve: -
 - 출제자: 오규민

모빌

- 모빌을 하나의 큰 Tree Graph로 볼 수 있습니다.

전체 문제를 부분 문제로 쪼갤 수 있습니다.

『정점 i 를 루트 노드로 하는 subtree의 최적 무게를 구하시오』

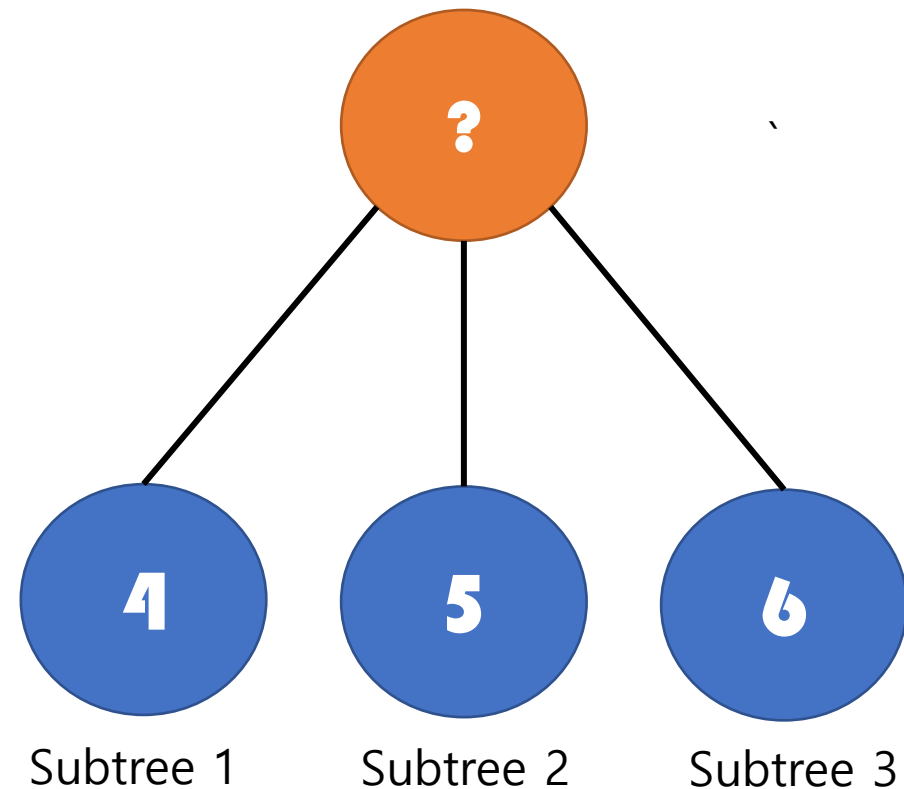
모든 subtree의 무게가 최적이라면,
최종적으로 만든 tree의 무게도 최적인 것이 보장됩니다!

모빌

- 부분 문제를 풀어봅시다.
- 주어진 tree의 최적 무게는:

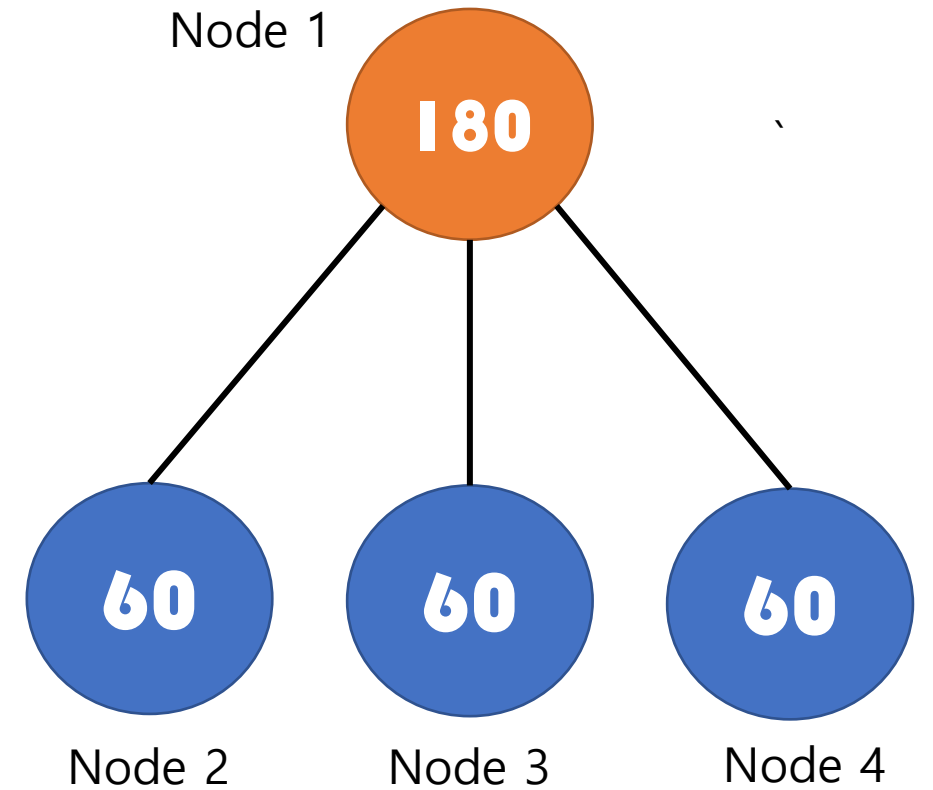
- 1) $4 * i = 5 * j = 6 * k$ 를 만족하는,
- 2) $4 * i + 5 * j + 6 * k$ 의 최솟값입니다 (단 i, j, k 는 자연수)

즉, 위 경우는 $i = 15, j = 12, k = 10$ 이며, 각 subtree의 무게는
모든 subtree의 최소공배수가 되어야 함을 알 수 있습니다.



모빌

- 부분 문제를 풀어봅시다
- 따라서, 어떤 서브트리의 무게는



[하위 서브트리의 개수 * 하위 서브트리 무게들의 최소공배수]
로 정리할 수 있습니다.

최소공약수는 유클리드 호제법을 이용해서 구하면 간단합니다

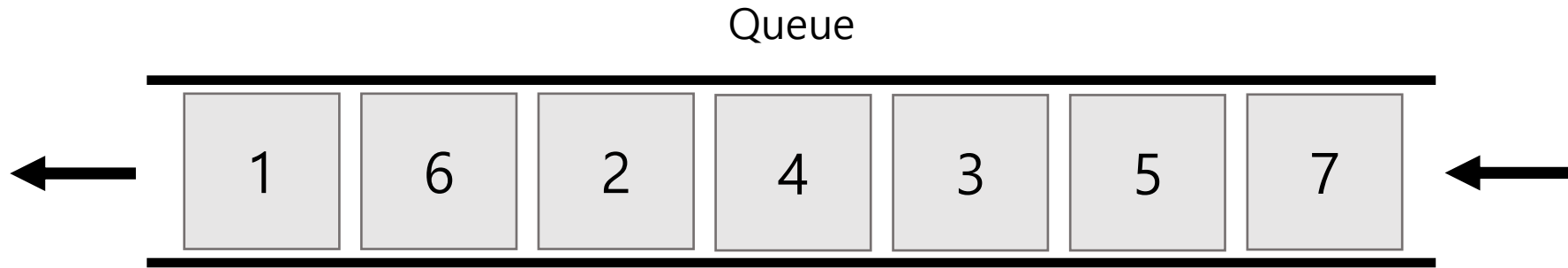
모빌

- 이제 DFS(깊이 우선 탐색)을 사용해서,
리프 노드부터 역순으로 올라오며 탐색을 하면 됩니다.
- 리프 노드에는 subtree가 없기 때문에 무게가 1입니다.
- 최종적으로 1번 노드를 루트 노드로 하는 트리의 무게가 문제의 최적해가 됩니다!
- 시간복잡도는 $O(n)$ 입니다.

스택과 큐

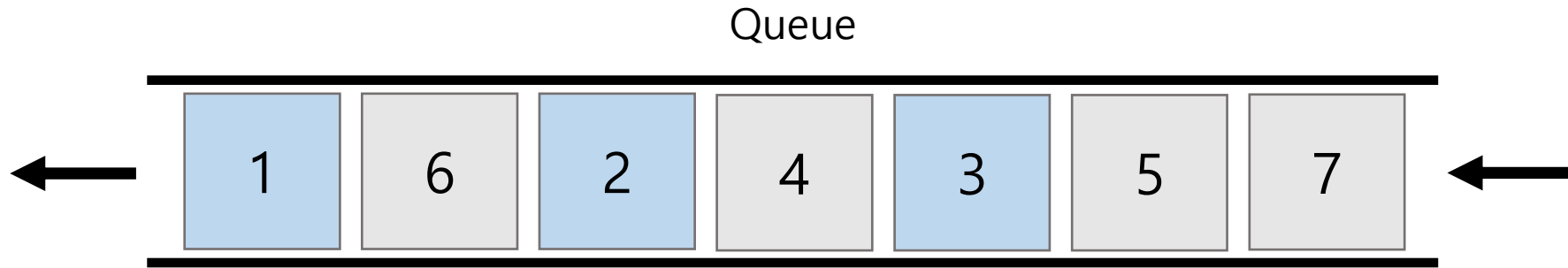
- 종류 - 그리디, 정렬
 - 출제진 의도 - Hard
-
- First Solve: 전예준 (Newbieeee)
 - 출제자: 오규민

■ ■ 스택과 큐



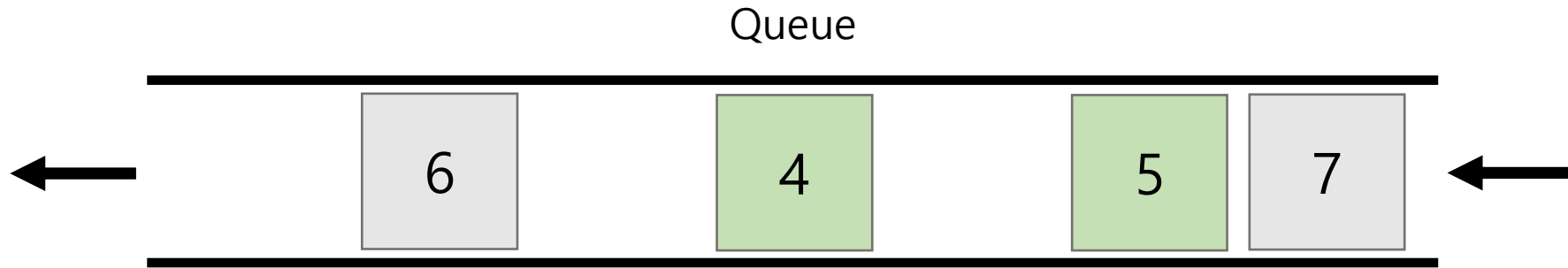
- 다음과 같은 순열을 예를 들어봅시다.
- 현재 큐의 모든 원소에 대해 한 번씩 다음 중 한 가지를 수행하는 것을 **SWEEP**이라고 합니다.
 - 원소를 버퍼로 pop하고 큐에 push한다(Queue-pop, Queue-push)
 - 원소를 버퍼로 pop하고 스택에 push한다(Queue-pop, Stack-push)

■ ■ 스택과 큐



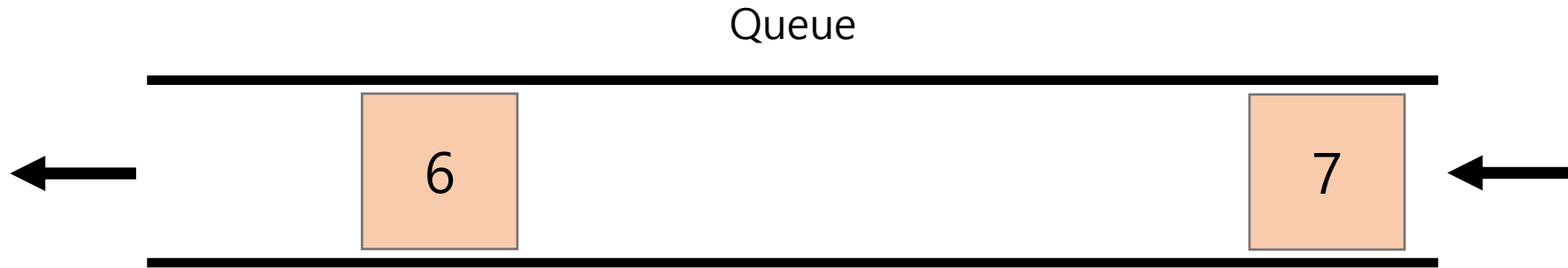
- 첫번째 SWEEP에서 스택에 push 할 수 있는 원소는 위와 같이 1, 2, 3 입니다.
- 첫번째 SWEEP에서 연산 횟수는 14번 입니다. (남은 원소의 수 \times 2)
 - 모든 원소에 대해서 [Queue-pop, Queue-push] 또는 [Queue-pop, Stack-push]를 한 번씩 수행했기 때문입니다.

■ ■ 스택과 큐



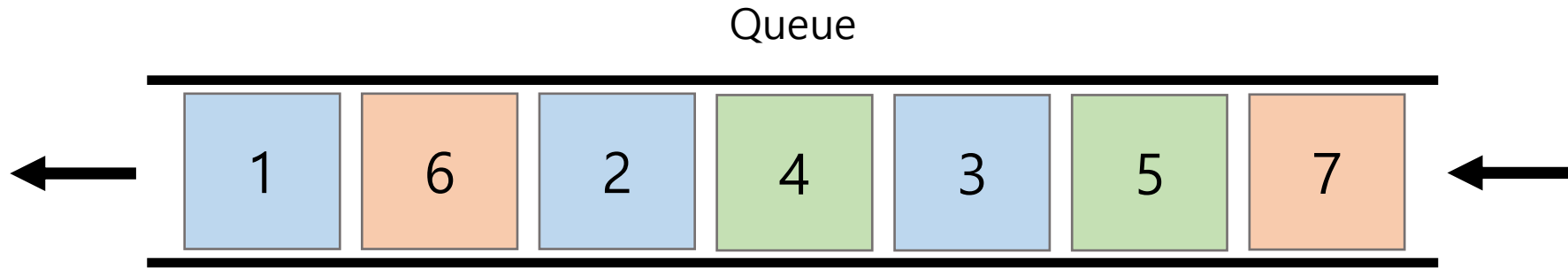
- 두번째 SWEEP에서 스택에 push 할 수 있는 원소는 위와 같이 4, 5 입니다.
- 두번째 SWEEP에서 연산 횟수는 8번 입니다.

■ ■ 스택과 큐



- 두번째 SWEEP에서 스택에 push 할 수 있는 원소는 위와 같이 6, 7 입니다.
- 두번째 SWEEP에서 연산 횟수는 4번 입니다.

■ ■ 스택과 큐



- 이 연산을 직접 시뮬레이션 하면 시간초과(TLE)를 받습니다. 순열이 내림차순일 때 연산 횟수가 $O(N^2)$ 에 비례합니다.
- 위 과정을 효율적으로 수행해봅시다.

■ ■ 스택과 큐



원소	1	2	3	4	5	6	7
인덱스	1	3	5	4	6	2	7
	1번째 SWEEP			2번째 SWEEP		3번째 SWEEP	

- 각 원소를 스택에 push하는 순서로 정렬 후 원래 순열에서의 인덱스를 살펴봅시다.
- 인덱스를 이전 원소와 비교했을 때
 - 인덱스가 증가한다면 이전 원소와 같은 SWEEP에서 스택에 push하고
 - 인덱스가 감소한다면 다음 SWEEP에서 스택에 push합니다.(다음 SWEEP으로 넘어가야합니다.)

■ ■ 스택과 큐

원소	1	2	3	4	5	6	7
인덱스	1	3	5	4	6	2	7
	1번째 SWEEP			2번째 SWEEP		3번째 SWEEP	

- 이제 각 SWEEP마다 몇 번의 연산을 수행하는지 합하면 최종 수행횟수를 구할 수 있습니다.
각 SWEEP에서 연산 횟수는 [남은 원소의 수 \times 2] 입니다.
 - 1번째 SWEEP : 남은 원소 {1, 2, 3, 4, 5, 6, 7}, 연산 횟수 14번
 - 2번째 SWEEP : 남은 원소 {4, 5, 6, 7}, 연산 횟수 8번
 - 3번째 SWEEP : 남은 원소 {6, 7}, 연산 횟수 4번
- 위 예시에서 총 연산 횟수는 26번 입니다.

■ ■ 스택과 큐

- 정렬 이후 한 번의 for문으로 위 과정을 수행할 수 있습니다.
- 답이 int 범위를 초과할 수 있습니다.
- **총 시간 복잡도: $O(N\log N)$**
- 여담: 순열에 같은 값의 원소가 있는 경우, 스택에서 pop이 가능한 경우 위 솔루션으로 문제를 풀 수 없습니다.

수상자 발표

■ Division 1 시상

- 대상 (1명): 유시온 님 (kir3i)
- 금상 (1명): 전예준 님 (Newbieee)
- 은상 (1명): 한상일 님 (131)
- 장려상 (3명): 이관희 님 (samdasoo), 김나연 님 (inourbubble2), 김지산 님 (wlts98)

■ Division 2 시상

- 대상 (1명): 이명규 님 (kaki1013)
- 금상 (1명): 박성우 님 (ece2017440048)
- 은상 (1명): 김민호 님 (vmfoslkim)
- 장려상 (3명): 김동현 님 (Corine013), 김형철 님 (tori1753), 장호빈 님 (hhhbabb)

AL林으로 오세요!



acm International Collegiate
Programming Contest

code jam

print "hello, world!"

2018

1st KAKAO

BLIND

RECRUITMENT

1차 코딩 테스트 문제 해설

감사합니다!



[대회 문제 및 테스트케이스 Repository](#)