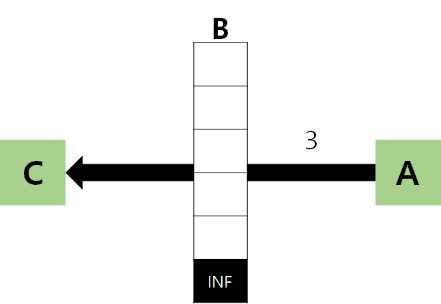
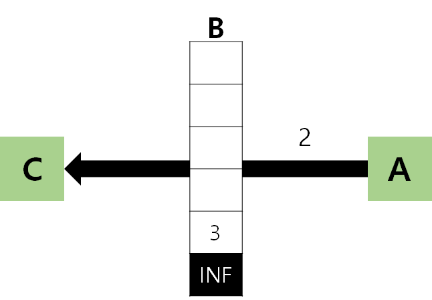
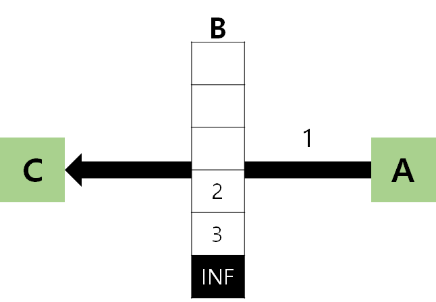
**고급소프트웨어실습I 보고서 4**

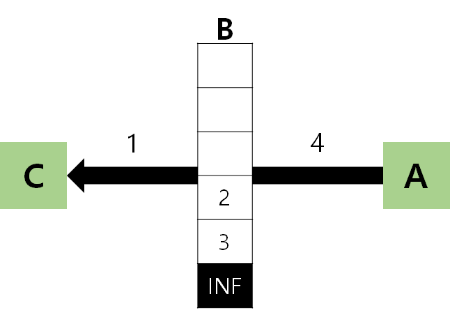
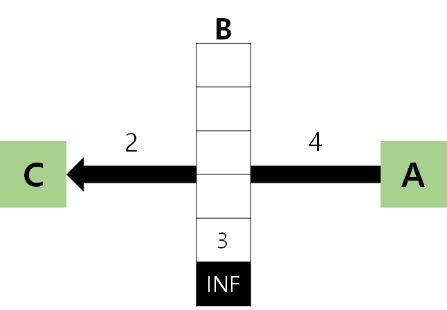
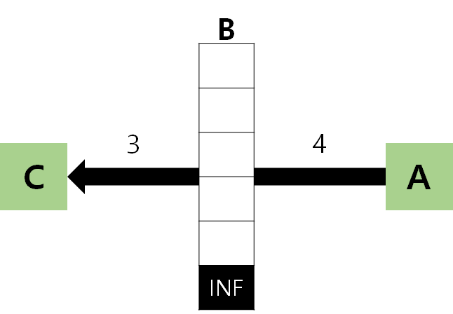
**20211584 장준영**

**A) Train Rearrangement**

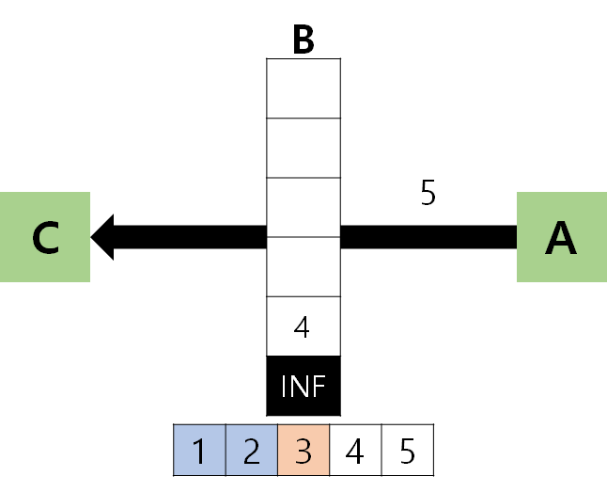
열차를 (b) 레일에 보관하여 순서를 정렬할 수 있는지 판단하는 문제이다. (b)는 last-in-first-out 구조이기 때문에 스택으로 표현할 수 있고, 따라서 순서를 정렬하는 방법이 하나로 정해져 있다. 예시로 3, 2, 1, 4 순서로 들어오는 경우를 확인해보겠다.

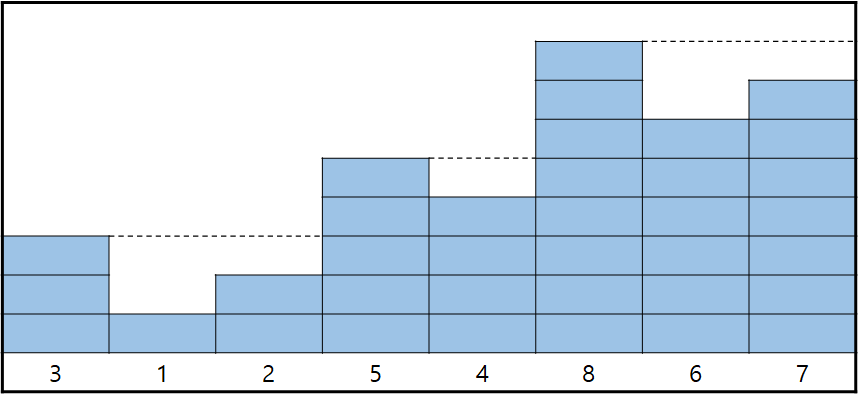
(b) 레일에 INF를 넣고 시작한다. 3이 들어오면 (b)의 top과 값을 비교해, 그보다 작다면 push한다. 따라서 3, 2, 1 모두 (b)에 들어온 상태이다.

다음으로 4 차례가 되면, 마찬가지로 (b)의 top과 값을 비교한다. top인 1이 4보다 작으므로, pop한다. 이 과정을 4보다 큰 top이 될 때까지 반복한다. 그러면 1, 2, 3이 모두 pop 되고 4가 push된다. 모든 열차를 확인했을 때 순차적으로 pop하면 1, 2, 3, 4 순으로 정렬할 수 있다. 이 방식을 적용했을 때, 정렬이 된다면 rearranging이 가능한 순열이고, 그렇지 않다면 불가능한 순열이다.



불가능한 순열로는 2, 1, 4, 5, 3이 있다. 위의 그림은 2, 1, 4까지 push를 마치고 5의 차례인 상황이다. 5를 push하기 위해선 (b)에 있는 5보다 작은 기차를 모두 pop해야 하는데, 현재 정렬된 순서에서 기다리고 있는 열차는 3번이기 때문에 4가 pop되면 정렬이 될 수 없다. 따라서, rearranging이 불가하다.



이것이 가능하려면, 순열을 탐색하면서 최댓값이 갱신될 때마다 이전 최댓값보다 작은 모든 숫자를 처리 완료한 상태여야 한다. 그림으로 표현하면 위와 같다.

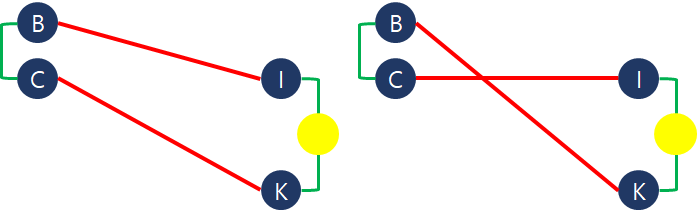
- 순열을 모두 탐색 :

▶ 한 열차에 대해 최대 두 번만 탐색하기 때문에 이다.

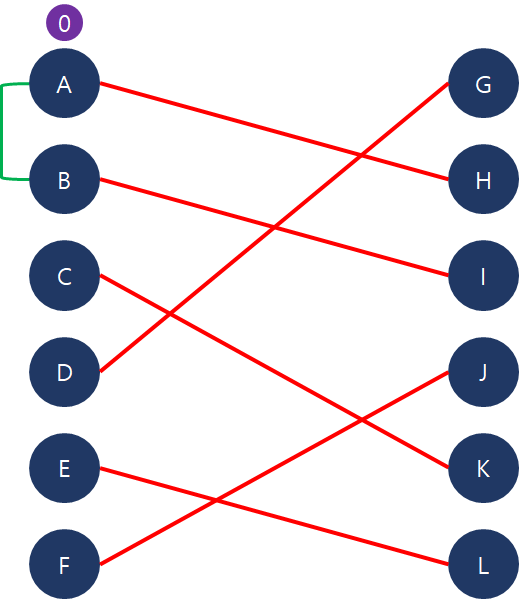
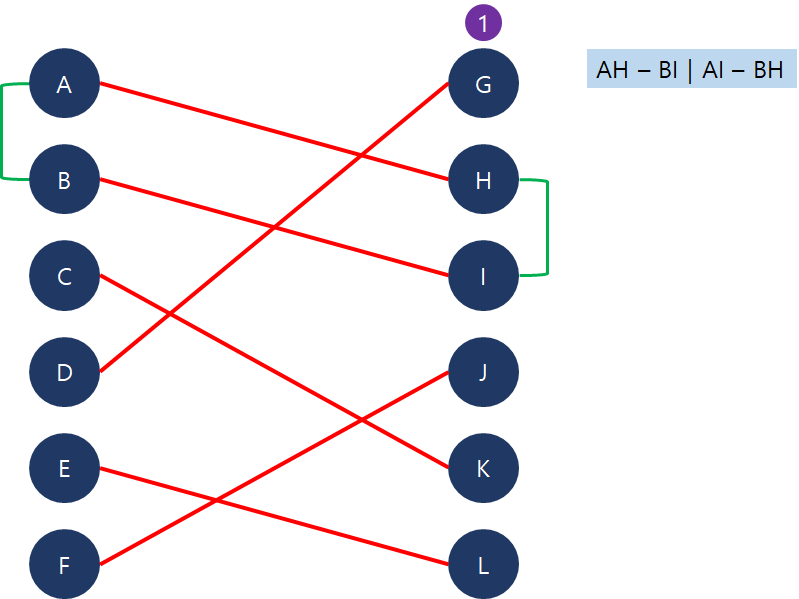
**>> 총 시간 복잡도 :**

**A) Agony of Engineer**

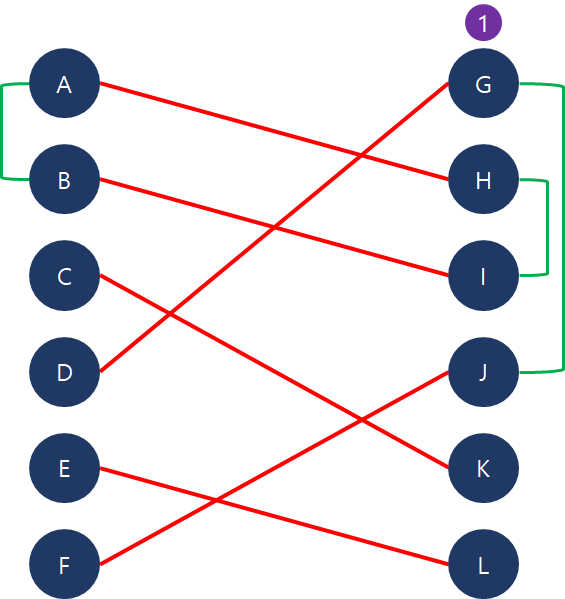
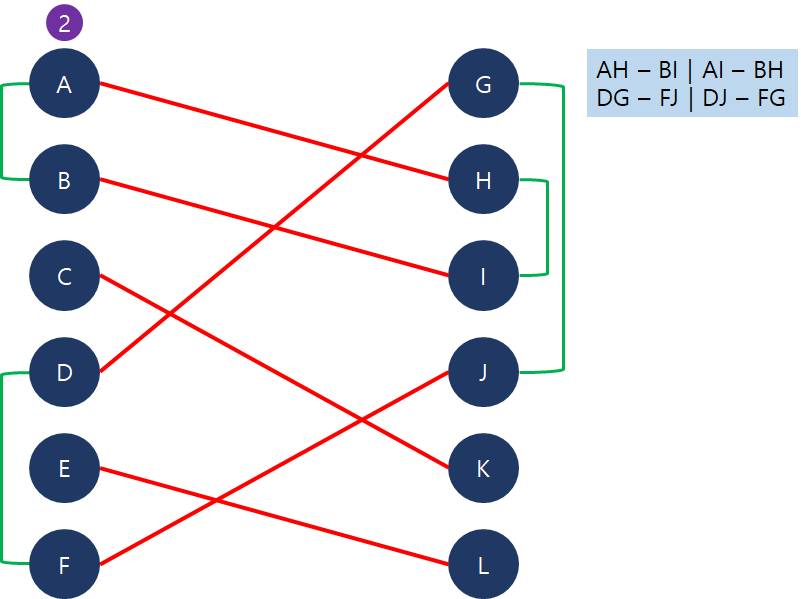
나의 알고리즘은 1) 후보 쌍 찾기, 2) 짝 검증으로 이루어져 있다.

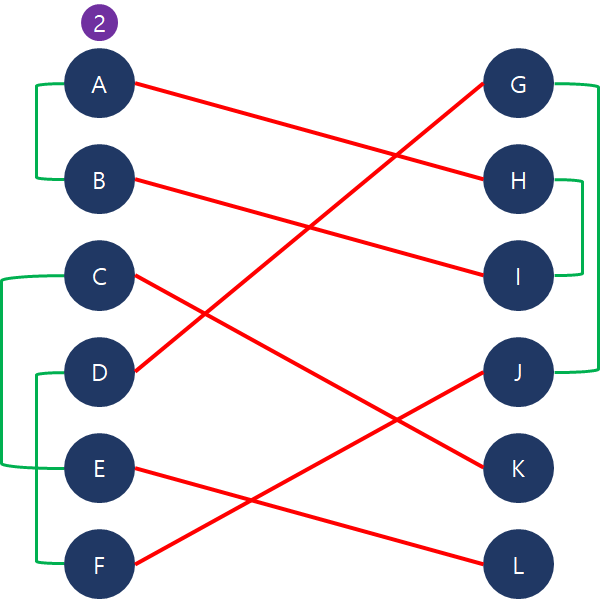
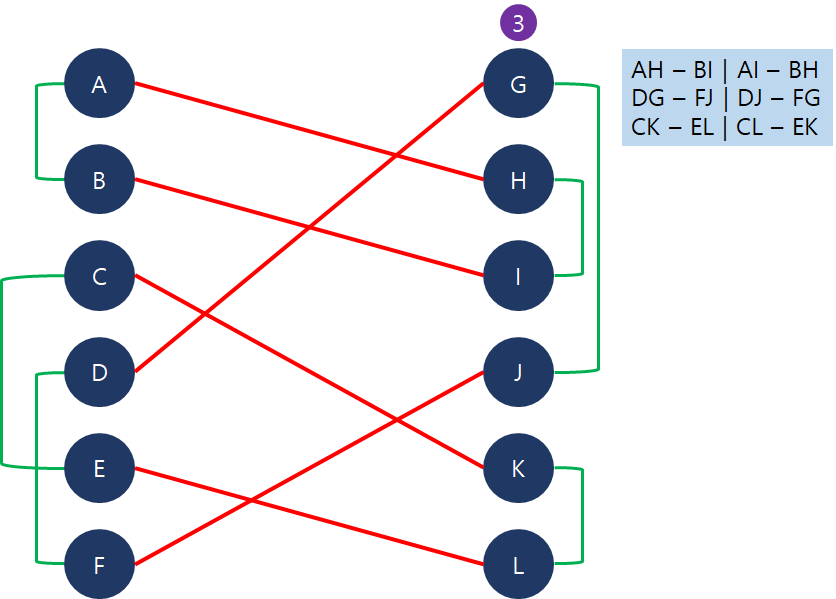


만약 B, C에 전선을, I, K에 다른 전선을 연결하고 배터리와 전구를 달았을 때 전구에 불이 들어온다면 위와 같은 연결쌍의 후보가 생긴다. BI-CK로 연결되었을 수도 있고, BK-CI로 연결되었을 수도 있다. 우선은 이런 후보 쌍을 모두 찾을 것이다. 이는 의 움직임으로 마칠 수 있다.

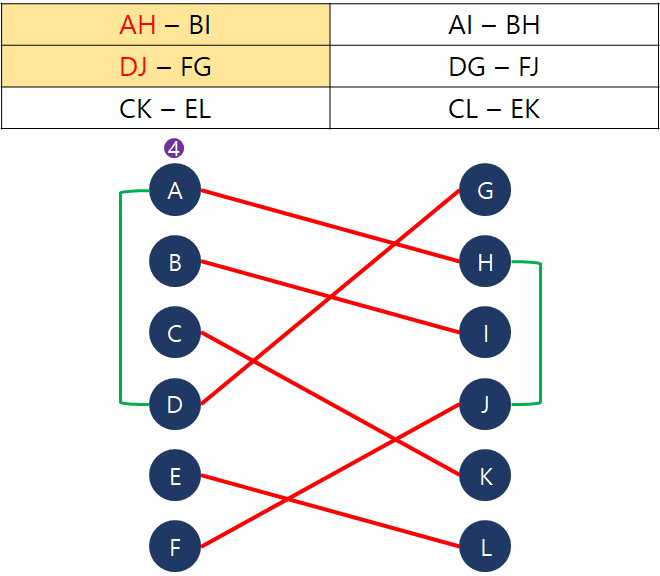
 

한 쪽 건물에서 시작해 빠른 번호부터 두 개를 전선으로 연결한다. 다른 쪽 건물로 이동해 전구에 불이 들어오게 할 수 있는 쌍을 찾는다. 이 경우, (A, B)와 (H, I)가 연결될 때 불이 켜지므로 가능한 후보 쌍은 AH-BI 혹은 AI-BH이다. 이를 기록한다.

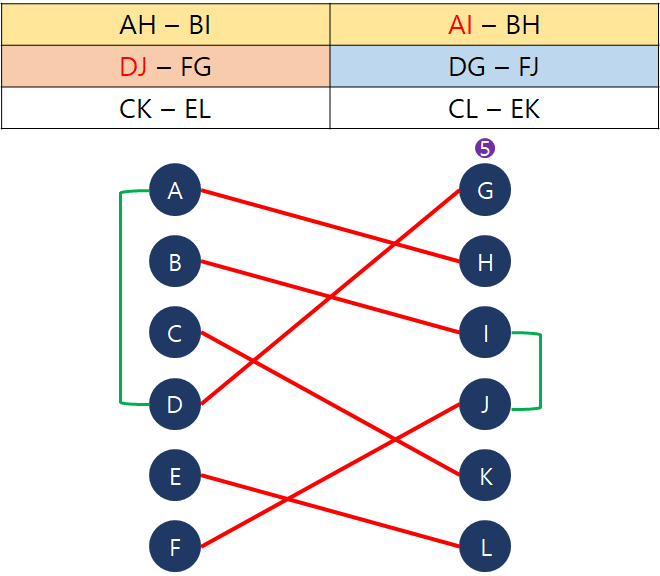
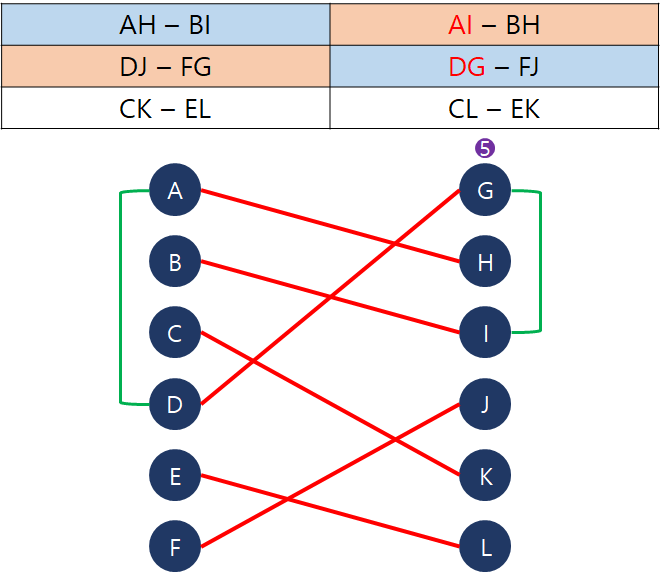
 

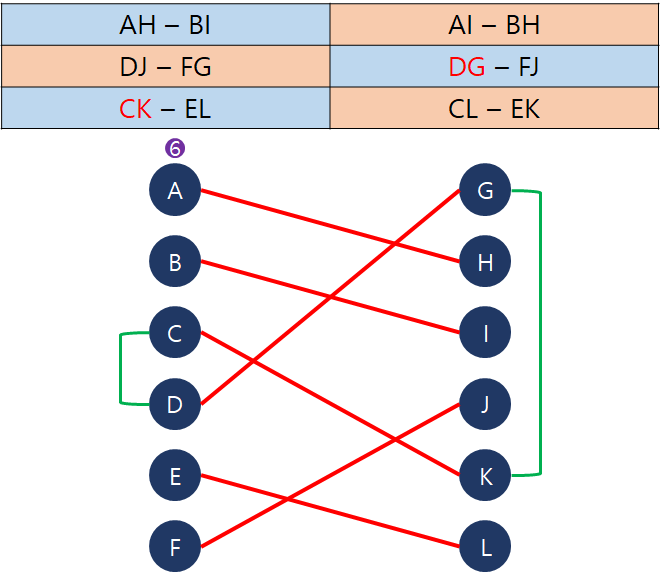
이렇게 존재하는 모든 후보 쌍을 의 움직임으로 찾았다면, 이제 그 두 가지 경우의 수 중 옳은 경우를 추려야 한다. 이는 두 후보 쌍을 비교하며 진행할 것이고, 이 역시 의 움직임으로 가능하다. 후보 쌍에는 두 개의 연결이 포함되어 있는데, 그 중 한 개의 연결만 참이라면 다른 연결 역시 참이다. 따라서, 후보 쌍의 연결 두 개 중 앞의 연결만을 사용하여 움직임을 최소화할 것이다.



우선 오른쪽 건물에서, 작성된 후보 쌍 중 두 개를 택해 앞의 연결만 확인한다. 이 경우 AH와 DJ이므로, 이 연결이 참인지 확인하기 위해 오른쪽 건물에 있는 터미널인 H, J를 전선으로 연결한 후 왼쪽 건물로 이동한다. 이후, 왼쪽 건물의 터미널인 A, D를 전선으로 연결해본다. 이 경우, 연결되지 않았으므로 불이 들어오지 않는다.

그렇다면 왼쪽 건물에서 확인할 수 있는 다음 후보인 AI-BH 후보 쌍으로 눈을 돌려, A, D를 전선으로 연결해본다. 이 경우에도 연결되지 않았으므로 불이 들어오지 않는다. 그 어떤 경우에도 불이 들어오지 않는다면, 두 번째 후보 쌍에 문제가 있음을 알 수 있다. 따라서, DJ-FG는 연결되지 않았고, 자연스럽게 DG-FJ가 연결되었다는 것을 알 수 있다. 다시 오른쪽 건물로 돌아와서, 연결되었다고 판단되는 DG-FJ를 이용해 I, G를 연결한다. 이때 불이 들어온다면 왼쪽에서 연결해놓고 온 후보 연결이 옳은 연결임을 알 수 있다. 이 경우, 불이 들어오지 않으므로 최종적으로 AH-BI, DG-FJ 두 연결이 참임을 알 수 있다. 두 번의 움직임으로 네 개의 연결을 찾아내었다. 이제부터는 옳은 연결로 찾아낸 바로 직전의 연결 쌍을 이용해 다음 연결 후보 쌍에서 맞는 경우를 추려낼 것이다.



이전에 DG-FJ가 옳은 연결임을 알았으므로, 다음 후보 쌍 중 한 가지 경우인 CK-EL을 검증하기 위해 G, K에 전선을 연결하고 이동한다. C, D를 연결했을 때 불이 들어오므로 이 연결은 옳은 연결이다. 이렇게 하면 의 움직임으로 모든 연결을 찾아낼 수 있다.

- 후보 연결 쌍 찾기 : (번)

- 옳은 연결 찾기 : (번)

**>> 총 이동 수 : (번)**

**▶ 홀수 개의 터미널이 있을 때 이동 수를 한 번 줄일 수 있는 이유는, N-1개의 쌍에 대해 두 개의 subtask의 해답을 찾아냈을 때 나머지 한 쌍은 자동으로 정해지기 때문이다.**