

알튜브

그리디 알고리즘

오늘은 '욕심쟁이 기법'이라고도 불리는 그리디 알고리즘에 대해 배웁니다.
지금 이 순간 최적의 답이 곧 전체 문제의 답인 알고리즘이에요.

그리디 알고리즘

- 현재 상황의 가장 최선의 선택으로 결국 전체 문제의 최선의 답을 만드는 기법
- 시간적으로 매우 효율적임
- 모든 순간 답이 되는 방법은 아님

어떨 때 그리드를 적용하지?

그리드 알고리즘

- 주로 $O(N)$ 시간복잡도를 갖기에 입력 범위가 큰 경우가 많다 (1,000,000 이상인 경우)
- 순간의 최적해가 전체 문제의 최적해가 되어야 함
- 그렇기에 정렬 후 접근하는 문제가 굉장히 많음

순간의 최적해 = 전체 문제의 최적해

- 이를 판단하기 위해선, 수학적 증명이 많이 요구됨 (즉, 판단 어려움)
- 따라서, 코딩 테스트의 경우 비슷한 문제나 직관에 의해 판단할 수 있는 문제가 주로 출제
- 많은 문제 연습이 필요! 감을 익히자

/<> 13975번 : 파일 합치기 3 - Gold 5

문제

- 두 개의 파일을 합쳐서 하나의 임시 파일을 만들고, 이 임시파일이나 원래의 파일을 계속 두 개씩 합쳐서 파일을 합쳐나가고, 최종적으로 하나의 파일로 합치는 문제
- 두 개의 파일을 합칠 때 필요한 비용을 두 파일 크기의 합이라 할 때, 최종 한 개의 파일을 완성하는 데 필요한 최소 비용 구하는 문제

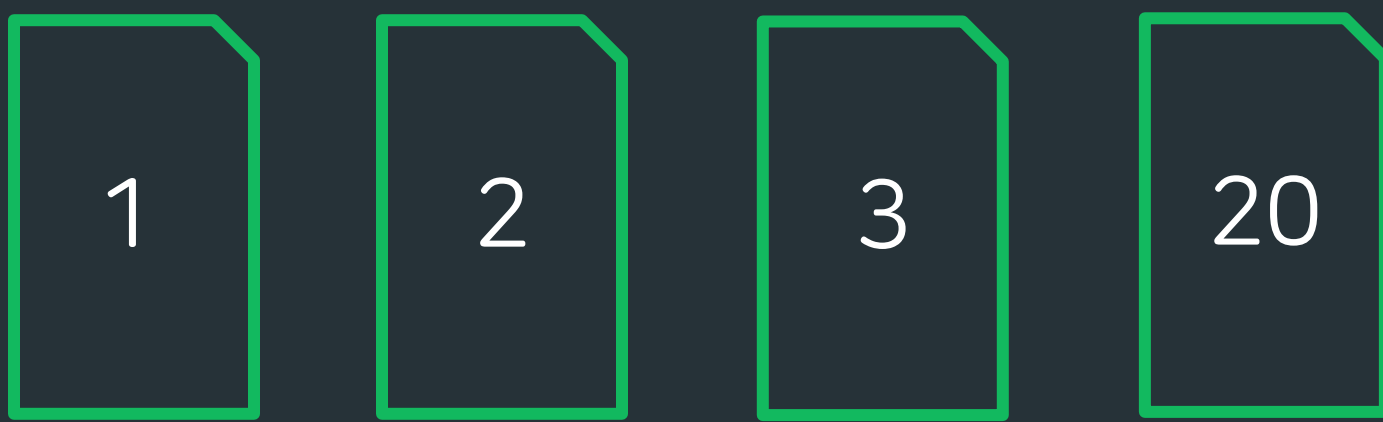
제한 사항

- 파일 개수의 범위 $3 \leq K \leq 500$
- 파일 크기의 범위 $\leq 10,000$

한 번 해보자!



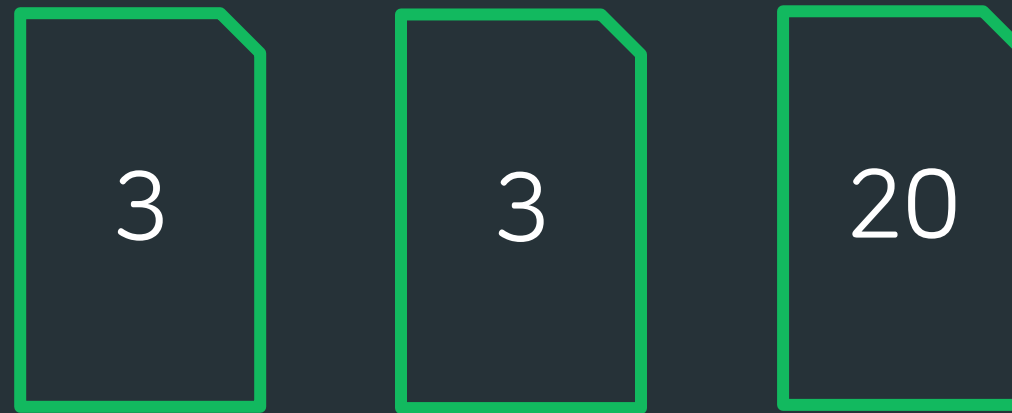
● 1 20 2 3
→ 1 2 3 20



한 번 해보자!

● 3 3 20

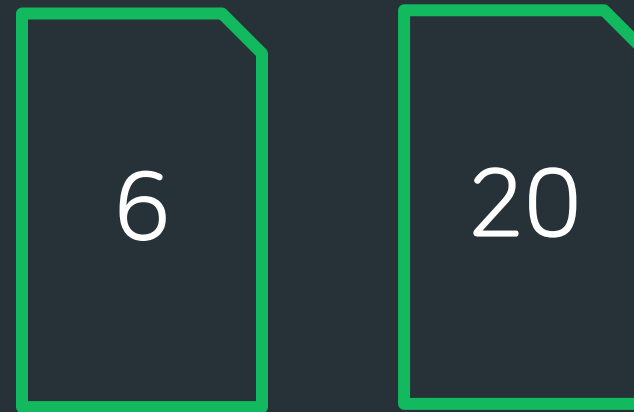
● 파일 비용
= 3



한 번 해보자!

● 6 20

● 파일 비용
= 3 + 6



한 번 해보자!

- 26

- 파일 비용
 $= 3 + 6 + 26 = 35$



/<> 13975번 : 파일 합치기 3 - Gold 5

문제

- 두 개의 파일을 합쳐서 하나의 임시 파일을 만들고, 이 임시파일이나 원래의 파일을 계속 두 개씩 합쳐서 파일을 합쳐나가고, 최종적으로 하나의 파일로 합치는 문제
- 두 개의 파일을 합칠 때 필요한 비용을 두 파일 크기의 합이라 할 때, 최종 한 개의 파일을 완성하는 데 필요한 최소 비용 구하는 문제

→ 파일 순서에 대한 제약 조건이 없으므로 항상 크기가 작은 파일 2개를 합치면 된다!

→ 즉, 그 순간에서 크기가 최소가 되도록 합치는게 최종 답을 만드는 그리디 풀이!

/<> 11066번 : 파일 합치기 - Gold 3

문제

- 두 개의 파일을 합쳐서 하나의 임시 파일을 만들고, 이 임시파일이나 원래의 파일을 계속 두 개씩 합쳐서 파일을 합쳐나가고, 최종적으로 하나의 파일로 합치는 문제
- 두 개의 파일을 합칠 때 필요한 비용을 두 파일 크기의 합이라 할 때, 최종 한 개의 파일을 완성하는 데 필요한 최소 비용 구하는 문제
- 이때, 파일이 연속이 되도록 합쳐 나간다 -> 즉, 인접한 2개 파일만 합칠 수 있음

제한 사항

- 파일 개수의 범위 $3 \leq K \leq 500$
- 파일 크기의 범위 $\leq 10,000$

한 번 해보자!

● 1 20 2 3

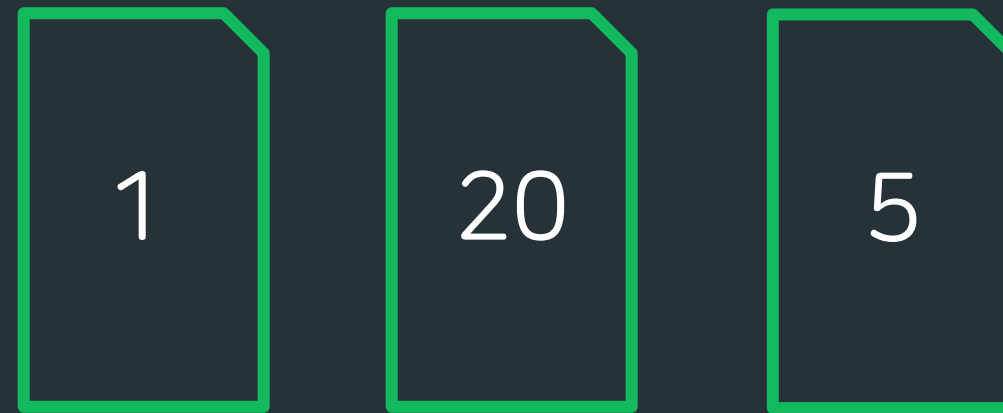
* 연속 파일을 합쳐야 하므로 순서 마음대로 못 바꿈!



한 번 해보자!

- 1 20 5
- 파일 비용
= 5

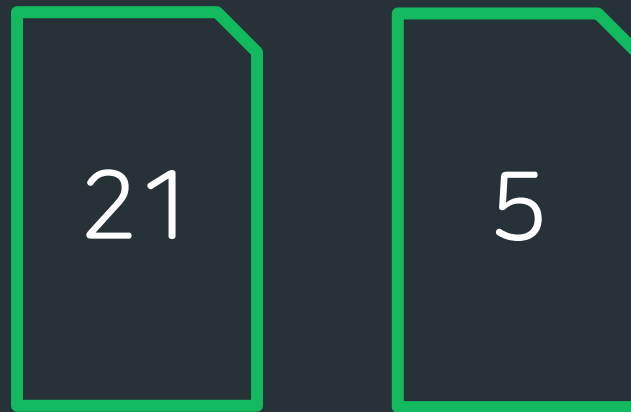
* 연속 파일을 합쳐야 하므로 순서 마음대로 못 바꿈!



한 번 해보자!

- 21 5
- 파일 비용
= 5 + 21

* 연속 파일을 합쳐야 하므로 순서 마음대로 못 바꿈!



한 번 해보자!

- 26
 - 파일 비용
- $$= 5 + 21 + 26 = 52$$

* 연속 파일을 합쳐야 하므로 순서 마음대로 못 바꿈!



아까 문제랑..?

- 1 20 2 3

그리디 풀이

- 파일 비용
 $= 3 + 6 + 26 = 35$

≠

현재 풀이

- 파일 비용
 $= 5 + 21 + 26 = 52$

/<> 11066번 : 파일 합치기 - Gold 3

문제

- 두 개의 파일을 합쳐서 하나의 임시 파일을 만들고, 이 임시파일이나 원래의 파일을 계속 두 개씩 합쳐서 파일을 합쳐나가고, 최종적으로 하나의 파일로 합치는 문제
- 두 개의 파일을 합칠 때 필요한 비용을 두 파일 크기의 합이라 할 때, 최종 한 개의 파일을 완성하는 데 필요한 최소 비용 구하는 문제
- 이때, 파일이 연속이 되도록 합쳐 나간다 -> 즉, 인접한 2개 파일만 합칠 수 있음

→ 파일 순서에 대해 제약조건이 있고, 이 조건이 그리디하게 푸는 풀이를 만족시키지 않기 때문에 그리디가 아니다!

→ 사실 이 문제는 DP 문제예요

/<> 11047번 : 동전 0 - Silver 2

문제

- N 종류의 동전을 무한히 가지고 있을 때, 적절히 사용해서 가치의 합을 K로 만든다
- 필요한 동전 개수의 최솟값을 구하는 문제
- N 종류 동전의 가치(A)는 오름차순으로 주어진다

제한 사항

- N의 범위 $1 \leq N \leq 10$
- K의 범위 $1 \leq K \leq 100,000,000$
- A의 범위 $1 \leq A \leq 1,000,000$ ($A[i]$ 는 $A[i-1]$ 의 배수)

예제 입력

```
10 4200
1
5
10
50
100
500
1000
5000
10000
50000
```

예제 출력

```
6
```

- 3000원을 만들어 보자!

100

200

400

800

- 3000원을 만들어 보자!

100

200

400

800

- 가능한 경우의 수

100 x 30

- 3000원을 만들어 보자!

100

200

400

800

- 가능한 경우의 수

100

x 30

200

x 15

- 3000원을 만들어 보자!

100

200

400

800

- 가능한 경우의 수

100 x 30

200 x 15

400 x 7

200 x 1

- 3000원을 만들어 보자!

100

200

400

800

- 가능한 경우의 수

100 x 30

200 x 15

400 x 7

200 x 1

800 x 3

400 x 1

200 x 1

- 가능한 경우의 수

$$\textcircled{100} \times 30 \quad \textcircled{200} \times 15 \quad \textcircled{400} \times 7 \quad \textcircled{200} \times 1 \quad \textcircled{800} \times 3 \quad \textcircled{400} \times 1 \quad \textcircled{200} \times 1$$

→ 뒤로 갈수록 동전 개수 줄어듬.

→ 즉, 가치가 적은 동전의 여러 개는 반드시 더 큰 동전으로 대체 가능 (동전 가치가 배수 관계 이므로)

→ 따라서, 가장 큰 동전부터 사용해나가면 된다!

→ 이는, 현재의 최적이 곧 답에서도 최적으로 적용되는 그리디 접근

/<> 2294번 : 동전 2 - Silver 1

문제

- N 종류의 동전을 무한히 가지고 있을 때, 적절히 사용해서 가치의 합을 K로 만든다
- 필요한 동전 개수의 최솟값을 구하는 문제

제한 사항

- N의 범위 $1 \leq N \leq 100$
- K의 범위 $1 \leq K \leq 10,000$
- 동전 가치 $\leq 100,000$, 가치가 같은 동전 여러 번 주어질 수도 있음

/<> 2294번 : 동전 2 - Silver 1

문제

- N 종류의 동전을 무한히 가지고 있을 때, 적절히 사용해서 가치의 합을 K로 만든다
- 필요한 동전 개수의 최솟값을 구하는 문제

제한 사항

- N의 범위 $1 \leq N \leq 100$
- K의 범위 $1 \leq K \leq 10,000$
- 동전 가치 $\leq 100,000$, 가치가 같은 동전 여러 번 주어질 수도 있음
→ 동전 가치에 대해 배수 조건이 없어서 불가능!

반례를 보여드릴게요

- 50원을 만들어 보자!

1

25

30

반례를 보여드릴게요

- 50원을 만들어 보자!

1

25

30

- 그리디로 접근했다면..

30

x 1

1

x 20

반례를 보여드릴게요

- 50원을 만들어 보자!

1

25

30

- 그리디로 접근했다면..

30

x 1

1

x 20

≠

- 실제 답은..

25

x 2

그리디 알고리즘

- 모든 순간에 적용할 수 있는 건 아님
- 순간의 최적해가 전체 문제의 최적해가 되어야 사용 가능!!
- 결국 많은 문제 연습이 필요! 감을 익히자
- 문제에서 요구하는 답에 "최소" "최대" 란 말이 주로 들어감

/<> 1931번 : 회의실 배정 - Silver 2

문제

- 한 개의 회의실이 있음. N개의 회의가 존재
- 각 회의 I에 대해 시작 시간과 끝나는 시간이 주어짐 (시작 시간과 끝나는 시간은 같을 수 있음)
- 각 회의가 겹치지 않게 하면서 회의실 사용할 수 있는 회의의 최대 개수를 구하는 문제
- 끝나는 동시에 다음 회의 시작 가능하며, 회의는 중단될 수 없음

제한 사항

- N의 범위 $1 \leq N \leq 100,000$
- 시작 시간과 끝나는 시간 $\leq 2^{31}-1$ (즉, int범위)

그리디 접근

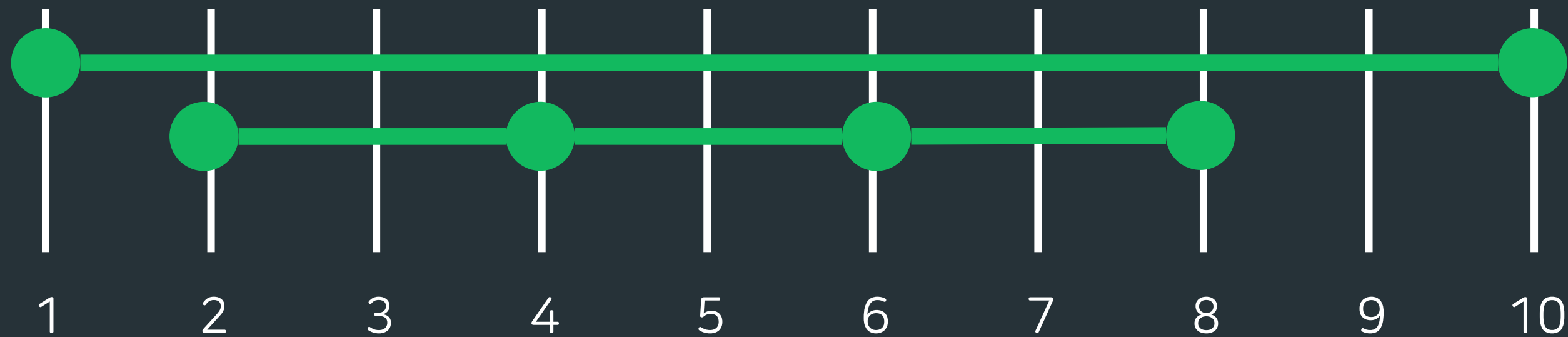
- 회의를 일찍하는 거부터 하면, 그 다음 회의를 많이 배치할 수 있지 않을까?

그리디 접근

- 회의를 일찍하는 거부터 하면, 그 다음 회의를 많이 배치할 수 있지 않을까?

반례

- (1, 10) (2, 4) (4, 6) (6, 8) 로 주어졌을 때



그리디 접근

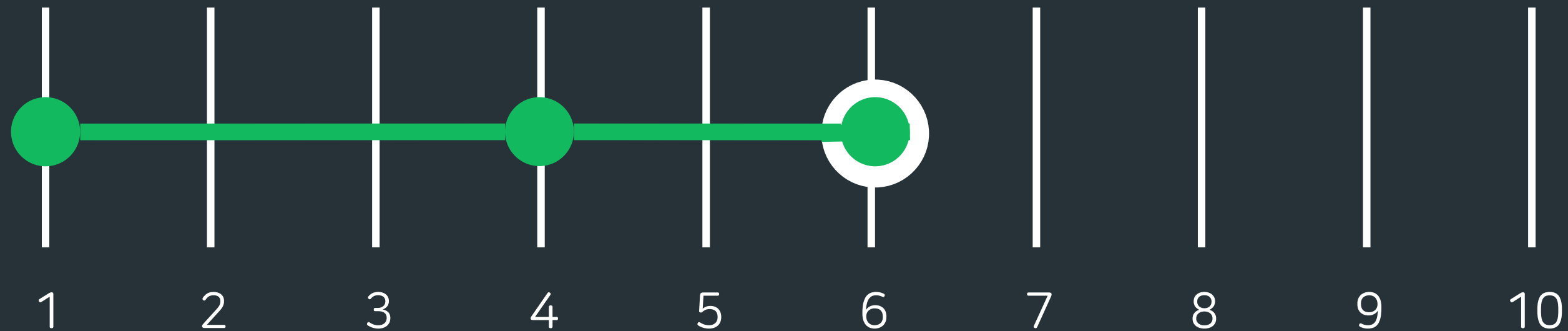
- ~~회의를 일찍하는 거부터 하면, 그 다음 회의를 많이 배치할 수 있지 않을까?~~
- 회의가 끝나는 시간이 빠를수록, 그 다음 회의를 많이 배치할 수 있다!
- 만약 끝나는 시간이 같다면? 시작 시간은 상관 없는건가..?

그리디 접근

- ~~회의를 일찍하는 거부터 하면, 그 다음 회의를 많이 배치할 수 있지 않을까?~~
- 회의가 끝나는 시간이 빠를수록, 그 다음 회의를 많이 배치할 수 있다!
- 만약 끝나는 시간이 같다면? 아래 반례로 인해 꼭 시작 시간이 빠른 순으로!

반례

- (1, 4) (4, 6) (6, 6) 로 주어졌을 때



- 다음과 같이 6개의 회의가 주어졌을 때 (시작 시간, 끝나는 시간)

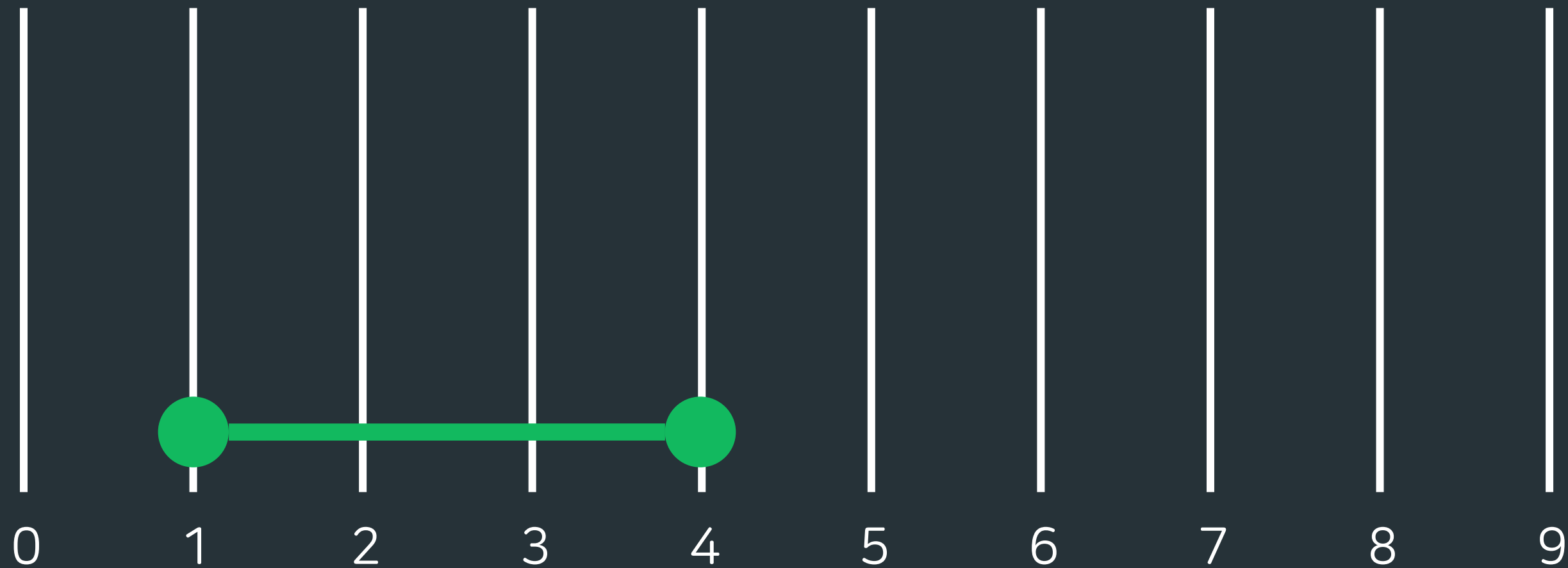
(1 4) (0 6) (5 7) (3 8) (5 9) (3 5)

- 다음과 같이 6개의 회의가 주어졌을 때 (시작 시간, 끝나는 시간)

(1 4) (3 5) (0 6) (5 7) (3 8) (5 9)

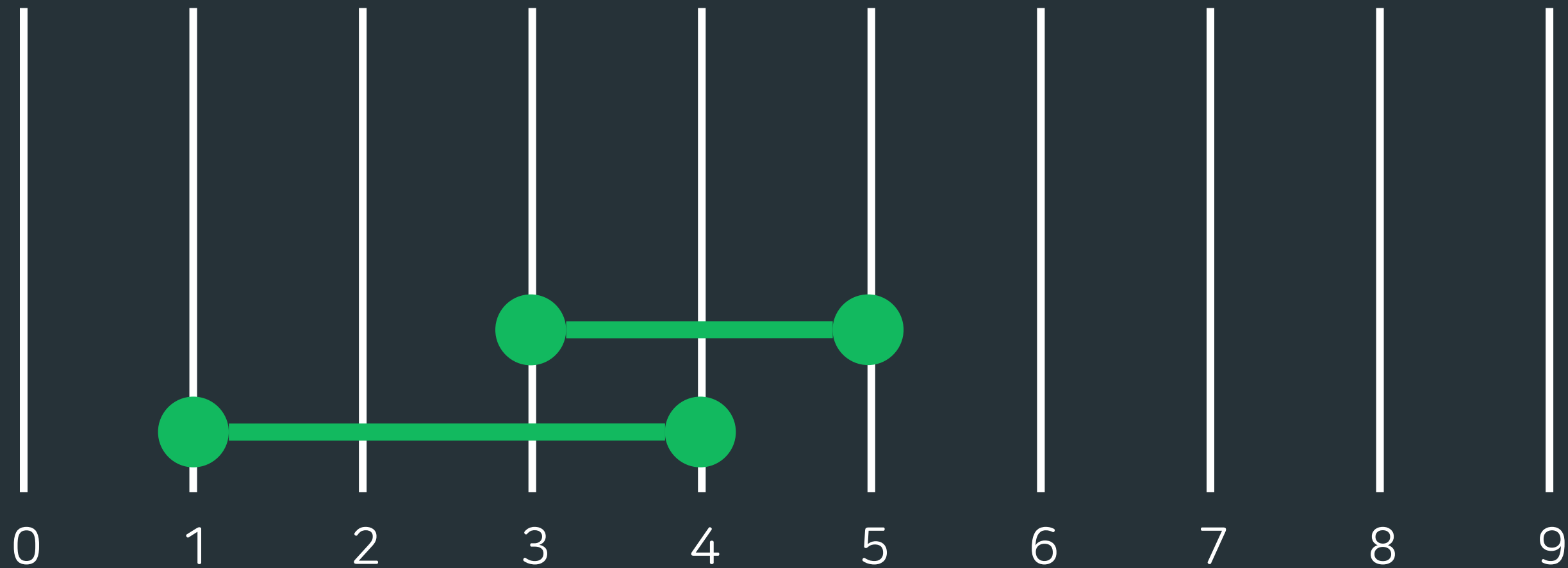
- 다음과 같이 6개의 회의가 주어졌을 때 (시작 시간, 끝나는 시간)

(1 4) (3 5) (0 6) (5 7) (3 8) (5 9)



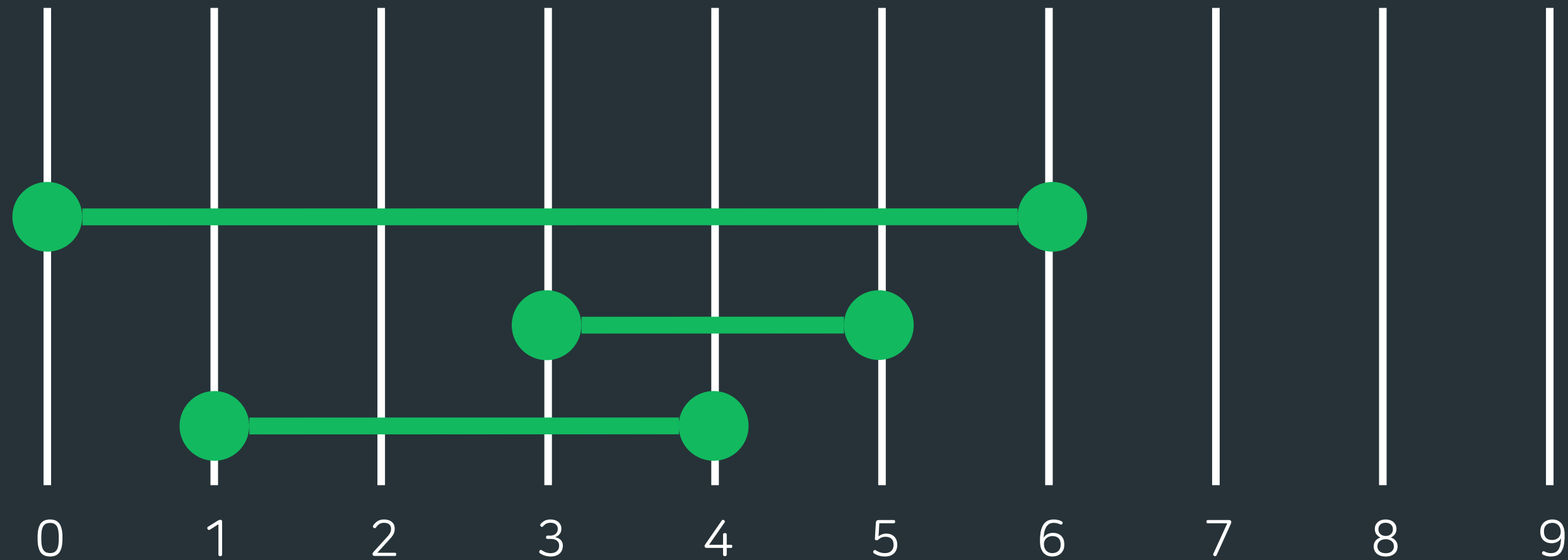
- 다음과 같이 6개의 회의가 주어졌을 때 (시작 시간, 끝나는 시간)

(1 4) (3 5) (0 6) (5 7) (3 8) (5 9)



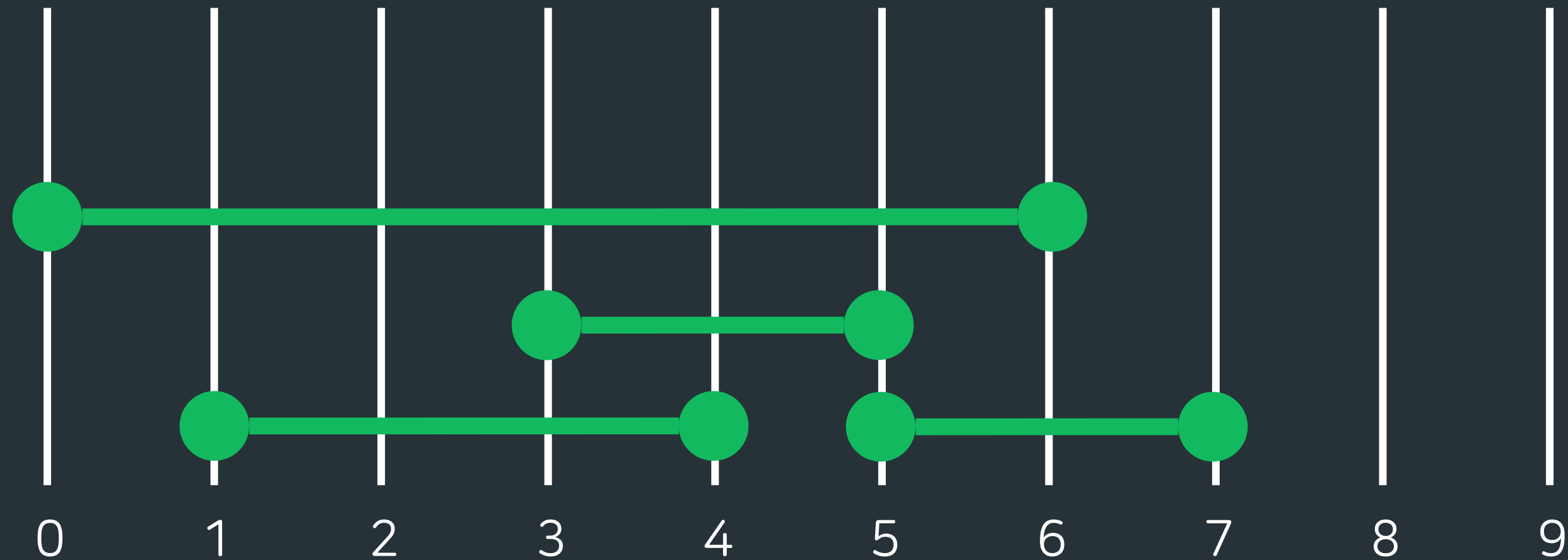
- 다음과 같이 6개의 회의가 주어졌을 때 (시작 시간, 끝나는 시간)

(1 4) (3 5) (0 6) (5 7) (3 8) (5 9)



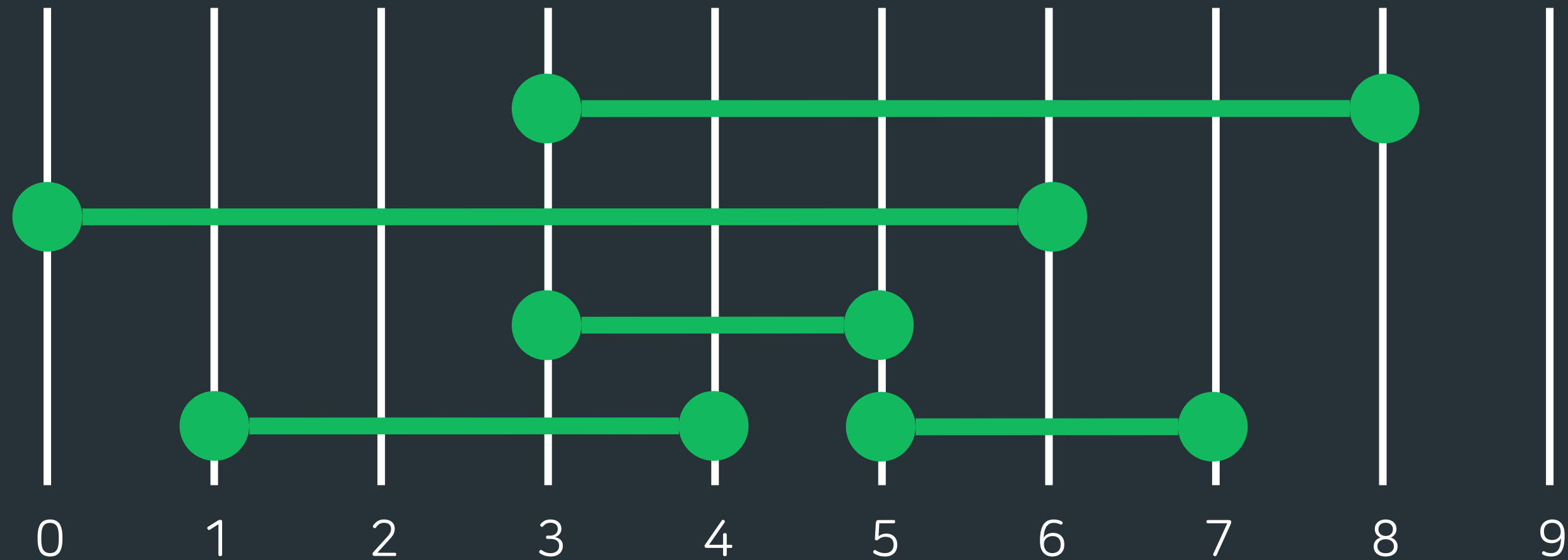
- 다음과 같이 6개의 회의가 주어졌을 때 (시작 시간, 끝나는 시간)

(1 4) (3 5) (0 6) (5 7) (3 8) (5 9)



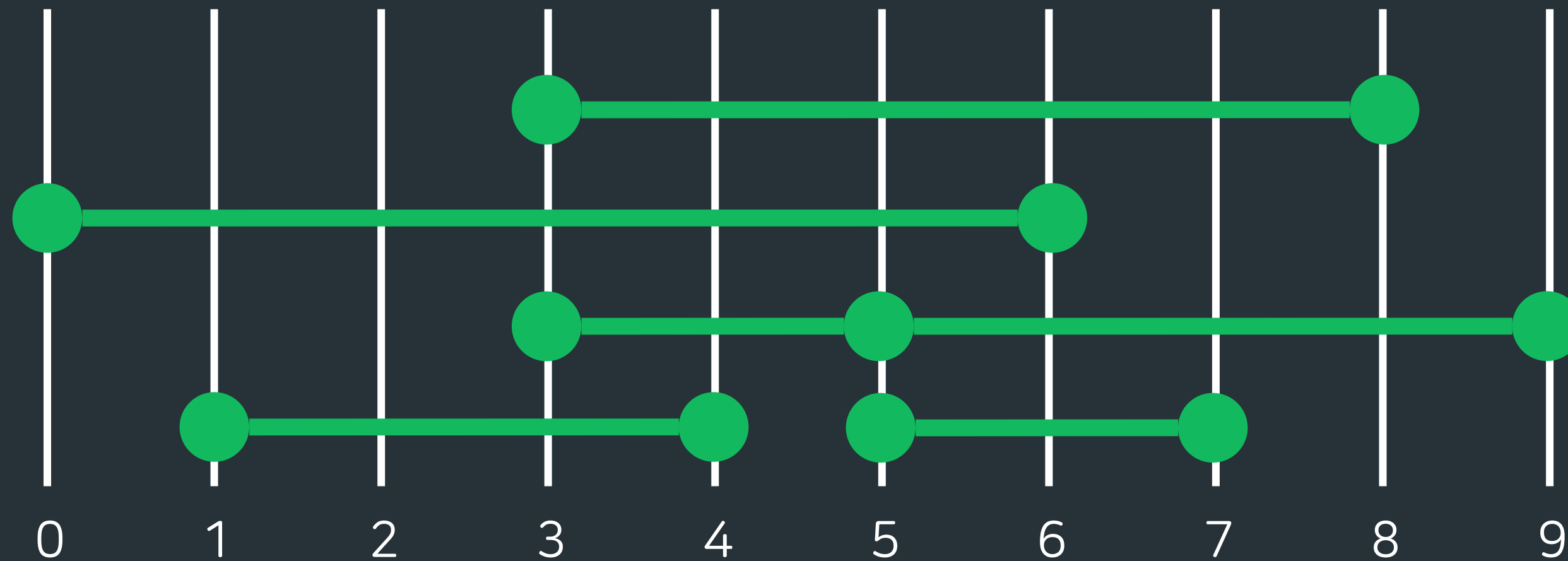
- 다음과 같이 6개의 회의가 주어졌을 때 (시작 시간, 끝나는 시간)

(1 4) (3 5) (0 6) (5 7) (3 8) (5 9)



- 다음과 같이 6개의 회의가 주어졌을 때 (시작 시간, 끝나는 시간)

(1 4) (3 5) (0 6) (5 7) (3 8) (5 9)



그리디 접근

- ~~회의를 일찍하는 거부터 하면, 그 다음 회의를 많이 배치할 수 있지 않을까?~~
- 회의가 끝나는 시간이 빠를수록, 그 다음 회의를 많이 배치할 수 있다!
- 만약 끝나는 시간이 같다면? 시작 시간이 빠른 회의를 기준으로!

→ 그 후, 다음 회의의 시작 시간이 현재 회의의 끝나는 시간보다 클 경우 배치시켜 주면 된다!

/<> 11000번 : 강의실 배정 - Gold 5

문제

- 시작 시간과 끝나는 시간이 주어진 수업 N개가 있다
- 최소한의 강의실을 사용하여 모든 수업을 가능하도록 하는 문제
- 즉, 필요한 최소 강의실을 구하는 문제

제한 사항

- N의 범위 $1 \leq N \leq 200,000$
- 시작 시간과 끝나는 시간 $\leq 10^9$ (즉, int범위)

탐욕법(Greedy) : 체육복 - Level 1

문제

- 일부 학생이 체육복을 도난당했다
- 다행히 **여벌 체육복**이 있는 학생이 다른 학생에게 체육복을 빌려주려 한다
- 학생들은 바로 **앞 번호의 학생**이나 바로 **뒤 번호**의 학생에게만 체육복을 빌려줄 수 있다
- 체육복을 적절히 빌려, 체육복이 있는 학생이 **가장 많도록** 하는 문제

제한 사항

- 전체 학생 수 $2 \leq N \leq 30$

탐욕법(Greedy) : 체육복 - Level 1

문제

- 일부 학생이 체육복을 도난당했다
- 다행히 **여벌 체육복**이 있는 학생이 다른 학생에게 체육복을 빌려주려 한다
- 학생들은 바로 **앞 번호의 학생**이나 바로 **뒤 번호**의 학생에게만 체육복을 빌려줄 수 있다
- 체육복을 적절히 빌려, 체육복이 있는 학생이 **가장 많도록** 하는 문제

접근

- 체육복을 빌려주는 학생 말고, **빌리는 학생** 기준으로 생각하자!
- 체육복은 **양 옆의 학생**에게 빌릴 수 있음
- **양 옆의 학생**에게만 빌리므로, 그리디하게 **제일 왼쪽 학생부터** 체육복 빌리는 것이 결국 **전체** 답을 만듦

정리

- 현재 최적의 해가 전체의 최적의 해라는 걸 알고 푸는 기법
- 입력 범위가 큰 경우가 많다
- DP 풀이가 생각났는데, 입력 범위가 너무 크다면 그리디로 접근해봐도 좋을 것
- 모든 경우에 성립하는 알고리즘은 아니다!
- 따라서 감을 익혀야 함. 많은 문제를 풀어보자!

이것도 알아보세요!

- 이전에 풀었던 과제들 중, 사실 그리디였던 문제가 굉장히 많아요. 찾아봅시다!

3문제 이상 선택

 탐욕법(Greedy) : 큰 수 만들기 - Level 2

/<> 13305번 : 주유소 - Silver 4

/<> 16206번 : 롤케이크 - Silver 1

/<> 1448번 : 삼각형 만들기 - Silver 3

/<> 1080번 : 행렬 - Silver 2

/<> 8980번 : 택배 - Gold 3