Deadline Scheduling

J(D, P) D는 데드라인, P는 이익 모든 일은 1시간 걸린다고 가정.

job은 P 기준 내림차순 정렬 ↘

N개의 job이 있을 때 $J_1 \cdots J_N$

모든 D는 N 이하 라면

<이익이 큰것부터 가능한 제일 뒤에 넣는다. (못넣으면 버림)>

J_i 후의 optimal solution S는 다음을 만족한다.

· $j \le i$ 인 J_j가 알고리즘의 답에 있다면 S에도 존재한다.

있다면 S에도 없다.

Base : $i=0$ 일때, 그냥 성립

Step : i 번째 까지 알고리즘 답 = S 라면 ···

$i+1$ 에서

Case 1. J_{i+1}이 선택 안됨.

선택안된 이유는 J_{i+1}의 D 이전이 꽉 차서.

근데 S에도 그 이전 까지 꽉 찼기 때문에 J_{i+1}은 필요.

Case 2. J_{i+1}이 선택 됨.

2-1 S에 J_{i+1}이 없는 경우

2-1-1 그 자리가 빈 경우

→ 그 자리에 J_{i+1}을 넣으면 더 이익이 커짐 (모순)

2-1-2 그 자리에 다른 job이 있는 경우

→ J_i 까지 알고리즘 답과 S가 같으므로,

그 다음 job은 J_{i+1} 보다 이익이 높다.

그럼 그냥 J_{i+1} 로 바꾸면 됨.

2-2 J_{i+1} 이 S에 있는 경우

2-2-1 같은 곳에 있는 경우 → 그냥 정답

2-2-2 더 뒤에 있는 경우 → 불가능, 있다면 알고리즘이 선택했을 것

2-2-3 더 앞에 있는 경우 → 뒤로 땡겨오면 됨. 이익은 같음.

귄형트리로 $O(n \log n)$ 으로 구현 가능.

또 다른 문제...

J(S,T) 시작시간 S와 끝나는시간 T 가 주어질때,
한번에 한 Job 만 할때 어떻게해야 가장 많은 Job 가능?

끝나는 시간 T 기준으로 오름차순 정렬.
가장 빨리 끝나는 것 선택 (겹치면 버림)

왜? <귀납법>
알고리즘의 답 A 가 최적의 해가 아니라고 하자. 최적의 해는 S라 하자.

S 에서 가장 빨리 끝나는 작업을 X, A에서 가장 빨리 끝나는 작업을 Y.

- X와 Y가 같은경우 : 끝나는 시간이 A와 같으므로, 그 이후 부터 동일한 문제가 반복됨.
- X와 Y 가 다른경우 : A는 가장 빨리 끝나는 작업을 추가했으므로,
Y가 X보다 빨리 끝난다.
S는 X를 선택하여 Y 끝나는 시간과 X 시작하는 시간 사이의 작업을 놓치게 된다 => 모순!

따라서 끝나는 시간이 빠른 작업들 순으로 추가하면 최적의 해를 구하지 못한다는 가정은 틀렸다!