태업[1]

[문제] 초고가의 정밀 측정기를 k대 보유한 연구기관이 있다. 이 연구소는 업체의 의뢰를 받아서 특정 물질을 검사/측정해주는데, 이 작업에는 하루가 꼬박 걸린다. 이 업체는 미리 요청을 받은 작업을 바탕으로 1일부터 T 일까지의 일정을 짜려고 한다. 신청된 각 작업(task) t_i은 지불할 비용 c_i와 그 작업의 허용 마감 날짜(due date) d_i로 구성되어 있다. 요청한 작업은 지정한 마감일을 넘어서 작업할 수는 없다. 따라서 마감일 이전에 검사를 완료해주어야 한다. 만일 요청받은 검사를 처리할 수 없을 경우에는 이 사실을 알려주고 검사 요청은 되돌려 보낸다.

그런데 최근 여기서 대우 문제로 노동쟁의가 발생하였다. 이에 동조하는 연구원들은 작업에 최선을 다하지 않기로 합의를 하였다. 그러나 모든 작업을 거부할 수는 없으므로 얻을 수 있는 이익 중에서 **최대가 아닌 차순위(second maximum) 단계의 이득을 받을 정도로만 일을 하기로 하였다.** 즉 일종의 태업(slowdown strike)을 하는 것이다.

여러분은 주어진 요청받은 N개의 작업을 k개의 측정 장비로 처리해줄 때 얻을 수 있는 **최대 이득(maximal profit)과 차순위(second maximum)의 이득, 즉 2번째로 높은 이득**을 계산해야 한다. 차순위 이득은 모든 실현 가능한 이득 중에서 최대 이득보다 작은(less than) 이득 중에서 최대인 것이다. 아래의 예를 들어 설명해보자.

만일 $T=5$(일), $k=2$(대)인 상황에서 아래와 같은 작업의뢰가 접수되었다고 하자. 수업 시간에 설명한 욕심쟁이 방법(Greedy Method)을 적용하면 최대 이득은 쉽게 계산할 수 있다. 그러나 최대 이득이 아닌 그 차순위 최대 이득을 구하는 방법에 대해서는 조금 더 생각을 해야할 것이다.[2]

t_i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
c_i	32	25	55	60	27	44	15	40	33	28	51	24	33	28
d_i	4	3	2	3	2	5	5	4	1	2	4	4	3	1

c_i는 이득, d_i는 마감일. 이 날을 넘기면 안된다.

1) slowdown strike라고 부른다. 이것은 일상적인 업무는 하되 의도적으로 느린 작업으로 능률을 떨어뜨리는 행위다. 우리나라에서 보통 "파업"이라고 부르는 사보타주(sabotage)는 이보다 훨씬 과격한 행동을 동반하는 것으로 공장이나 작업장의 기물을 훼손하는 폭력까지를 동반하는 단체 행동을 의미한다.

2) 만일 가능하다면 차차순의, 세번째로 높은 이득도 생각해볼 수 있다. 이 경우 시간 복잡도는 꽤 올라간다.

DAY	1	2	3	4	5
M1	**t9**	**t3**	**t4**	**t11**	**t6**
M2	**t10**	**t1**	**t13**	**t8**	**t7**

A schedule for maximal profit (최대 이득 일정)

[입출력] 입력과 출력은 표준 입출력 stdio와 stdout을 사용한다. 입력 파일의 첫 줄에는 전체 작업의 개수 $|\{t_i\}| = N$, 작업 일수 T, 기계의 대수 k를 나타내는 3개의 정수가 'N T k'로 주어진다. 이어지는 N개의 줄에는 각 작업의 이익과 마감 일자(due date)를 나타내는 값이 2개의 'c_i d_i'가 정수로 주어진다. 여러분은 이 N개의 작업을 정리해서 얻을 수 있는 최대 이득 A와 차순위 (second maximal) 최대 이득 B를 두 개의 정수로 출력해야 한다. 즉 A > B 이며 이 사이의 값을 가지는 어떤 작업 일정도 있어서는 안된다. 입력 데이터의 범위는 다음과 같다.

$$3 \le N \le 100,\ 1 \le T \le 10,\ 1 \le k \le 5,\quad 1 \le c_i \le 100,\quad 1 \le d_i \le T$$

stdio	**stdout**
14 5 2 32 4 25 3 55 2 ... 33 3 28 1	391 390

[제한조건] 프로그램 이름은 **slowdown.{c, cpp, py}**이다. 최대 수행시간은 1초이다. 허용 token의 수는 500이다.

[도전문제] 3번째, 4번째로 가능한 높은 이득을 구해보고, 이를 바탕으로 k번째 maximum 이득을 구하는 일반적인 문제의 알고리즘도 생각해보자. (ChatGPT를 활용해도 좋을 것이다.[3])

3) 알고리즘 문제에 대한 GPT의 답을 매우 조심해야 한다. 틀린 답을 늠름하게 제시하고도 모르는 척하는 경우가 많다.