2021 여름학기 동국대학교 SW역량강화캠프

7일차. 그리디 2

● 그리디 알고리즘

▶ 탐욕 알고리즘 / 욕심쟁이 알고리즘

▶ 현재 단계에서 가장 최선의 선택을 하는 알고리즘

▶ 모든 경우에서 통하지는 않으며, 사용할 때 증명이 필요

▶ 주로 예시를 통한 관찰과 규칙 찾기로 문제 해결

▶ 가장 큰 ~~, 가장 빠른 ~~, 가장 긴 ~~ 부터 ~~ 한다.

● 회의실 배정(2709)

문제

컴돌이는 한 개의 회의실을 관리하게되었다. 이 회의실에는 N개의 회의들이 예약되어 있는데, 컴돌이가 자세히 살펴본 결과 이 중 서로 시간이 겹쳐있는 것들이 있었다.

한 회의실에서 두 개 이상의 회의가 열릴 수 없기 때문에 컴돌이는 예약되어있는 몇 개의 회의를 취소하고 회의실 시간표를 짜려고 한다.

회의를 하나 취소할 때마다 입는 타격이 크기 때문에 회의는 최소한으로 취소하려고 할 때, 컴돌이가 열 수 있는 회의의 최대 개수를 구하는 프로그램을 작성하여라.

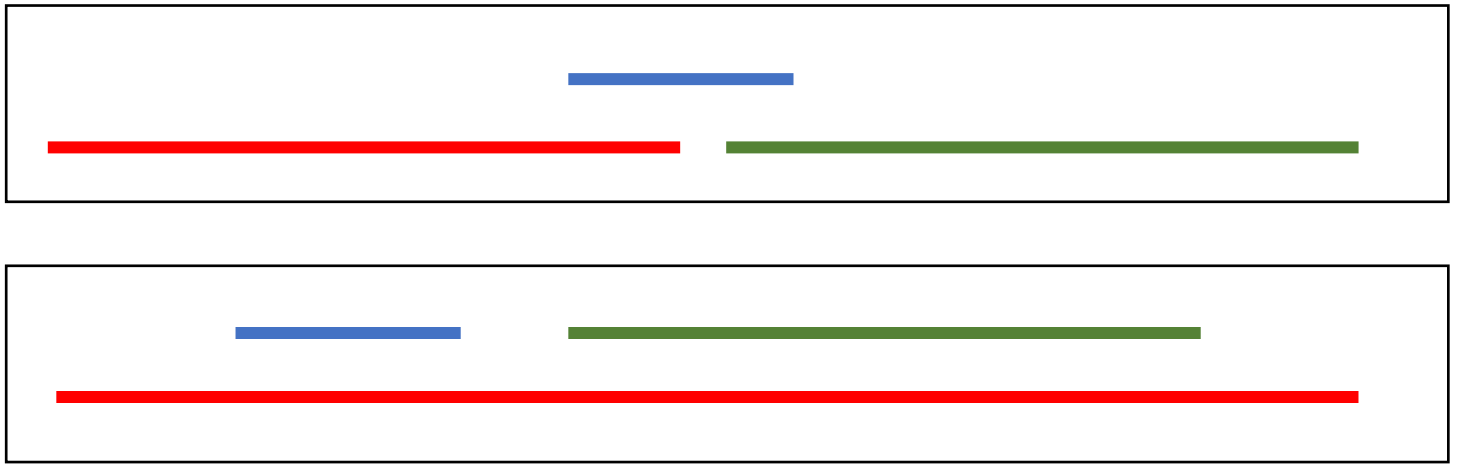
단, 회의는 한번 시작하면 중간에 중단될 수 없으며 한 회의가 끝나는 것과 동시에 다음 회의가 시작될 수 있다. 회의의 시작시간과 끝나는 시간이 같을 수도 있다. 이 경우에는 시작하자마자 끝나는 것으로 생각하면 된다.

▶ 최선의 선택은?

▶ 가장 회의시간이 짧은 회의부터 선택

▶ 가장 빨리 시작하는 회의부터 선택

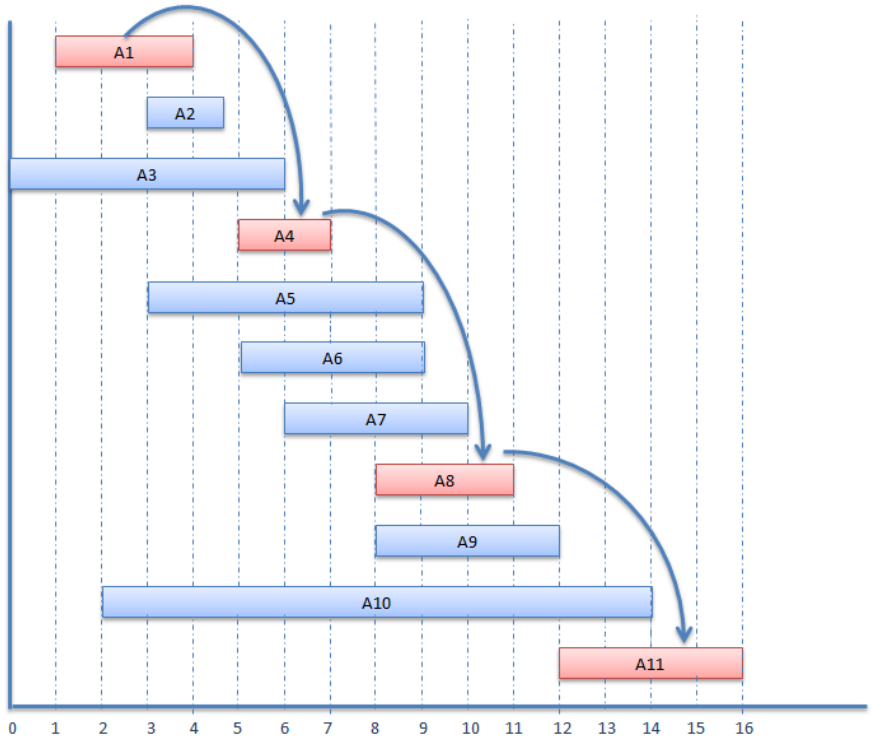
▶ 가장 빨리 끝나는 회의부터 선택



▶ 가장 빨리 끝나는 회의부터 선택

다음으로 진행되는 회의는 이 회의가 종료된 이후부터 진행 가능

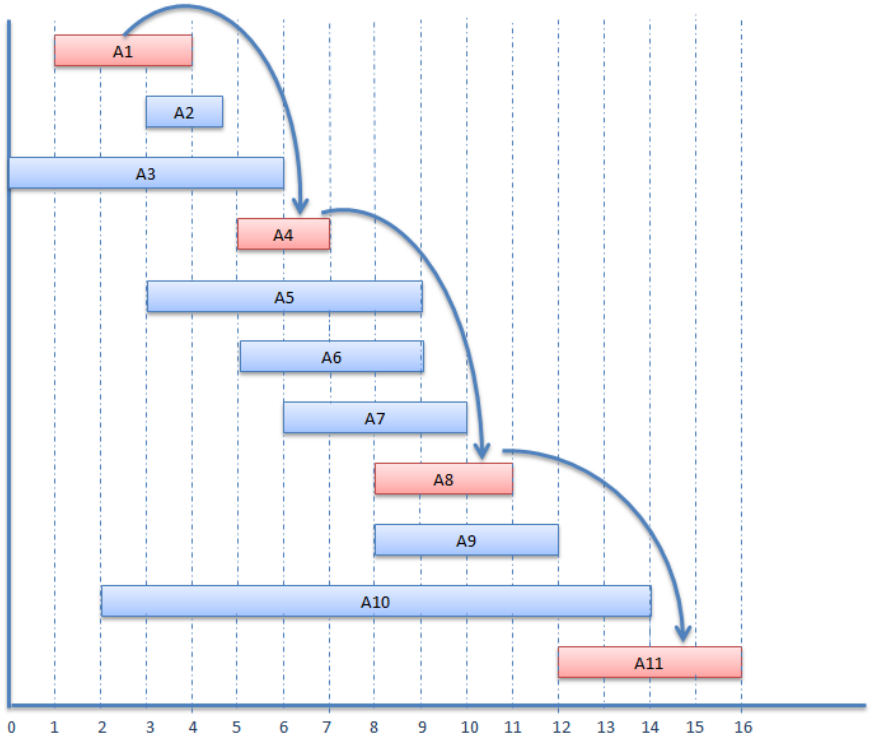
따라서, 가장 먼저 종료되는 회의를 선택하는 것이 이후 선택에 있어 비교우위를 가질 수 있다.



▶ 회의들을 종료시각 순으로 정렬

진행할 수 있는 회의들 중 가장 종료시각이 빠른 회의를 선택

→ 앞에서부터 보면서 가장 먼저 나오는 진행가능한 회의 선택



핵심 코드

```
ArrayList<Integer[]> arr = new ArrayList<Integer[]>();
for (int i = 0; i < N; i++) {
    arr.add(new Integer[] { start, end });
}

Collections.sort(arr, new Comparator<Integer[]>() { // end 기준으로 오름차순 정렬
    public int compare(Integer[] p, Integer[] q) {
        return p[1].compareTo(q[1]);
    }
});
```

핵심 코드

```
int ans = 0;
int last = 0;
// 종료시각이 빠른 회의부터 보면서, 진행할 수 있는 회의를 찾으면, 그 회의를 진행
// (진행할 수 있는 회의 중 종료시각이 가장 빠른 회의 선정)
for (int i = 0; i < N; i++) {
    Integer[] tmp = arr.get(i);
    if (last <= tmp[0]) {
        last = tmp[1];
        ans++;
    }
}
```

● 강의실 배정 (4167)

문제

오늘 학원에서는 N개의 강의가 열린다. 각 강의는 시작 시간과 끝나는 시간이 주어지고 한 강의실에서 동시에 두 개 이상의 강의가 진행될 수 없다. 그리고, 강의는 한번 시작되면 중간에 중단될 수 없으며 한 강의가 끝나는 것과 동시에 다음 강의가 시작될 수 있다. 강의의 시작 시간은 끝나는 시간보다 항상 작다. 이때 모든 강의를 진행하기 위해 강의실이 최소 몇개 필요한지 구해보자.

입력

첫째 줄에 강의의 개수 $N(1 \leq N \leq 10^5)$이 주어진다.
둘째 줄부터 $N+1$ 번째 줄까지 공백을 사이에 두고 강의의 시작시간 S_i와 끝나는 시간 E_i가 주어진다. $(0 \leq S_i < E_i \leq 2^{31}-1)$

출력

첫째 줄에 필요한 최소 강의실 개수를 출력한다.

문제 해설

▶ 강의 시작 시각 순으로 살펴보자

▶ 시간 x에 시작하는 강의가 있을 때, 현재 강의실들이 전부 사용 중이라면 새로운 강의실 개설

▶ 강의실 별로 강의 종료 시각을 저장해놓으면 시간 x에서의 강의실의 사용 여부를 알 수 있다.

1 4
2 3
3 6
5 6
5 8

문제 해설

▶ 모든 강의실이 전부 사용중이라는 정보를 빠르게 알 수는 없을까

▶ 종료시각이 가장 빠른 강의실과 비교

▶ 강의실 중 가장 빠른 종료시각 〉 강의 x 시작 시각이라면 새로운 강의실 개설

▶ 주어진 자료들 중 최소값을 빠르게 관리할 수 있는 자료구조

핵심 코드

```
Collections.sort(arr, new Comparator<Integer[]>() { // 시작시간 기준 정렬
    public int compare(Integer[] p, Integer[] q) {
        return p[0].compareTo(q[0]);
    }
});

PriorityQueue<Integer> pq = new PriorityQueue<>();
for (int i = 0; i < N; i++) { // 시작시간이 빠른 강의부터 보면서, 강의를 할 수 있는 강의실이 없다면 새로운 강의실 개설
    Integer[] tmp = arr.get(i);
    if (pq.isEmpty() || pq.peek() > tmp[0]) {
        pq.offer(tmp[1]);
    }
    else {
        pq.poll();
        pq.offer(tmp[1]);
    }
}
System.out.println(pq.size());
```

● 파일 합치기 (4152)

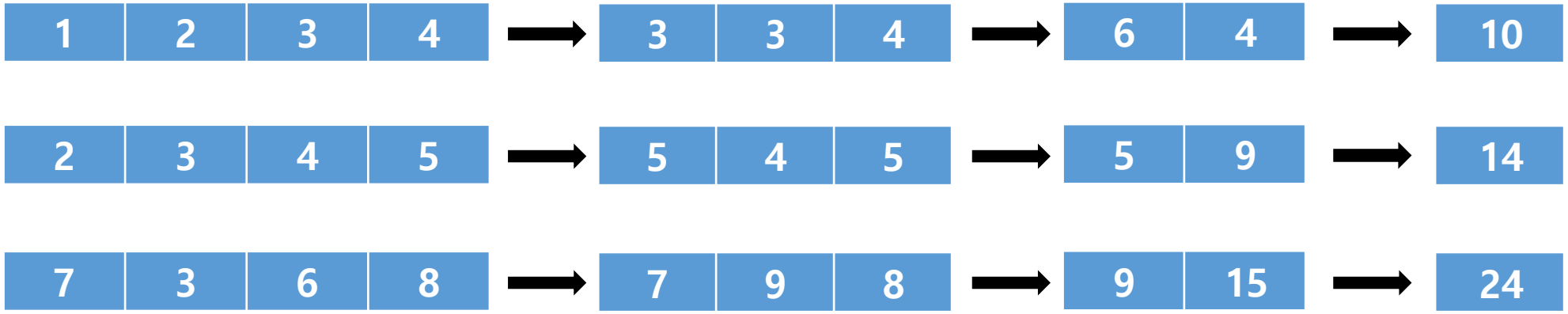
문제

윌리의 직박구리 폴더에는 N개의 파일이 있다. 윌리는 이 파일들을 전부 합쳐 하나의 파일로 만드려고 한다. 파일을 합칠 때에는 두 파일을 하나로 합치는 것을 반복해야 한다. 합칠 때에는 두 파일의 크기를 더한 것만큼의 시간이 걸린다.

놀랍게도, 파일을 고르는 순서에 따라서 비교 횟수가 달라진다. 예를 들어 크기가 10, 20, 40인 파일이 있다면 10과 20을 합친 뒤, 합친 30과 40을 합친다면 (10 + 20) + (30 + 40) = 100의 시간이 필요하다. 그러나 10과 40을 합친 뒤, 합친 50과 20을 합친다면 (10 + 40) + (50 + 20) = 120의 시간이 필요하므로 덜 효율적인 방법이다.

N개의 파일 크기가 주어질 때, 필요한 시간의 최솟값을 구해보자.

▶ 예시 2~3개를 통해 관찰



가장 작은 2개의 병합을 반복한다?

▶ 가장 작은 2개의 병합을 반복한다.

▶ 엄밀한 증명은 Skip! Proof by AC

▶ 가장 작은 2개를 합친다 → 우선순위 큐를 이용 (TreeSet도 가능합니다)

핵심 코드

```
int ans = 0;

while(pq.size() > 1) {
    int x = pq.poll();
    int y = pq.poll();
    ans += (x+y);
    pq.offer(x+y);
}
System.out.println(ans);
```

● 윌리의 과제 (4237)

문제

윌리는 과제가 많다. 하루에 한 과제를 끝낼 수 있는데, 과제마다 마감일이 있으므로 모든 과제를 끝내지 못할 수도 있다. 과제마다 끝냈을 때 얻을 수 있는 점수가 있는데, 마감일이 지난 과제는 점수를 받을 수 없다.

윌리는 가장 점수를 많이 받을 수 있도록 과제를 수행하고 싶다. 윌리를 도와 얻을 수 있는 점수의 최댓값을 구하시오.

입력

첫 줄에 정수 N $(1 \leq N \leq 1,000)$이 주어진다.

다음 줄부터 N 개의 줄에는 각각 두 정수 $d(1 \leq d \leq 1,000)$와 $w(1 \leq w \leq 100)$ 가 주어진다. d는 과제 마감일까지 남은 일수를 의미하며, w는 과제의 점수를 의미한다.

출력

얻을 수 있는 점수의 최댓값을 출력한다.

문제 해설

▶ d=1 이라면 1일에 과제 수행 가능, d=2 라면 1,2일에 과제 수행 가능

▶ 과제 기한에 대해 오름차순으로 과제들을 정렬하자.

▶ PriorityQueue 를 이용하여 현재 내가 고른 과제들을 관리
PQ = {2,3,5} 라면 1~3일에 점수가 2,3,5인 과제를 해결

▶ {D, W}인 과제가 들어왔다면

D가 pq.size()보다 크다면, pq에 W를 넣는다.

D가 pq.size()와 같다면 pq 내부에 있는 과제들 중 점수의 최소값을 W로 대체 가능한지 확인

기한	점수	PQ
1	20	
2	50	
3	30	
4	60	
4	30	
4	50	
6	5	

핵심 코드

```
PriorityQueue<Integer> pq = new PriorityQueue<>(); // pq는 지금 진행중인 과제들의 점수 목록을 담고 있다.

for (int i = 0; i < N; i++) { // 남은 기간이 짧은 것 먼저 살펴본다.
    Integer[] tmp = arr.get(i);
    int d = tmp[0], w = tmp[1]; // d일까지 진행할 점수 w의 과제
    if (pq.size() < d) // 만약 진행중인 과제가 d개 미만이라면 진행할 과제에 w 추가
        pq.offer(w);
    else { // 현재 진행중인 과제가 d개라면 진행중인 과제들 중 가장 점수가 작은 과제와 w 비교. w가 더 크다면 교체
        int minw = pq.peek();
        if (minw < w) {
            pq.poll();
            pq.offer(w);
        }
    }
}
```