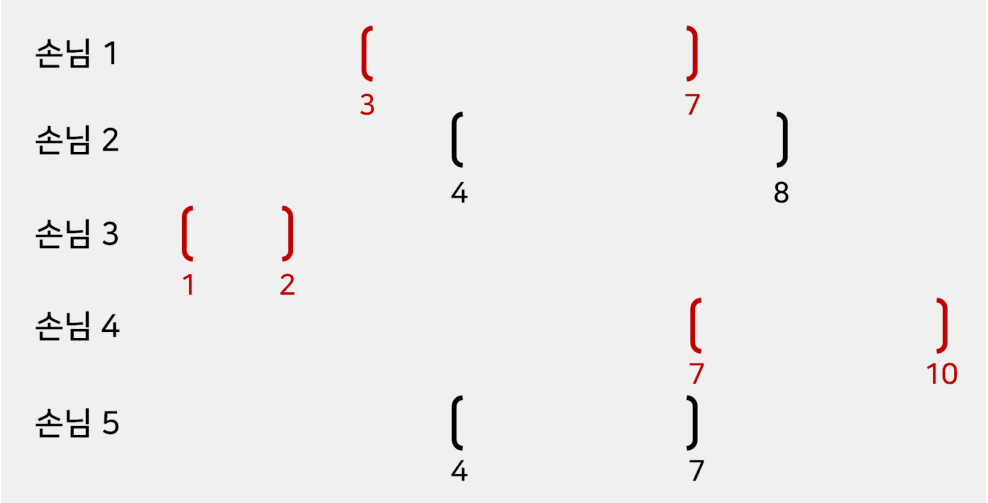
미용실 문제

현재 미용실에서 커트를 받고 싶은 사람의 수가 N명이다. N명의 사람들은 각각 초 단위로 예약을 진행했다고 가정하자. 각 사람이 예약한 커트 시간은 시작 시각($start$)과 종료 시각(end)으로 표시된다. 이를 배열로 나타낸다면 [$start$, end] 형태로 이해할 수 있을 것이다. 이때, 하루는 86,400초이므로, 시각 시각(start)은 0 이상 86,399 이하의 정수로 주어진다.

또한 미용사는 동시에 여러 명의 손님에 대하여 커트를 진행할 수 없다. 예를 들어 예약 시간이 [3000, 6000]인 손님과 [4000, 7000]인 손님 두 명이 있을 때, 두 손님을 모두 받는 것은 불가능하다. 또한, 손님들의 미용실 입장 시간 및 대기 시간은 0초라고 가정하자. 이때, 1명의 미용사가 받을 수 있는 **최대한의 손님의 수**를 계산하는 프로그램을 작성하여라.

예를 들어, N = 5명의 손님이 커트를 받고 싶은 상태라고 가정하자. 현재 예시에서 각 손님이 예약한 커트 시간은 [3, 7], [4, 8], [1, 2], [7, 10], [4, 7]이다. 이 경우 1명의 미용사가 받을 수 있는 예약자 수의 최댓값은 3이다.



입력 조건

가장 먼저 예약자 수 N이 자연수로 주어진다. (N은 $100,000$보다 작거나 같은 자연수다.)

이어서 각 예약자가 원하는 커트 시간 정보가 담긴 배열 $reserved$가 주어진다. 배열은 행의 크기가 N이고, 열의 크기가 2인 형태의 2차원 배열이다. 각 손님에 대한 예약 시간은 $[start, end]$ 형태로 주어진다. 각 손님에 대해 $start$는 0 이상 86,399 이하의 정수이며, end는 $start + 1$ 이상 86,400 이하의 정수이다.

출력 조건

한 명의 미용사가 받을 수 있는 최대한 많은 손님의 수를 반환하여라.

입출력 예시

N	$reserved$	정답
5	[[3, 7], [4, 8], [1, 2], [7, 10], [4, 7]]	3
8	[[2, 7], [7, 10], [10, 13], [10, 12], [12, 15], [15, 16], [15, 18], [16, 17]]	6

해설 1. 미용실

본 문제는 Activity Selection Problem의 한 예시로, 각 손님의 커트 시간이 겹치지 않게 하면서 미용사가 받을 수 있는 최대한 많은 손님의 수를 구하는 것이 목표다. 문제 해결 아이디어는 다음과 같다.

1. 가장 먼저 모든 예약자에 대하여 오름차순 정렬한다.
2. 정렬할 때는 **(1) 종료 시점**과 **(2) 시작 시점**을 우선순위로 한다.
3. 첫 번째 예약자부터 시작해 겹치지 않게 최대한 많은 예약자를 선택한다.

본 문제는 해결을 위하여 정렬 알고리즘을 요구한다는 점에서 시간 복잡도 $O(NlogN)$으로 해결할 수 있다. 결과적으로 본 문제는 **정렬 및 그리디 알고리즘 유형**에 속한다.

- **Python3 정답 코드 예시**

```
# 예약자의 수(N), 예약자가 원하는 커트 시간 정보가 담긴 배열(reserved)
def solution(N, reserved):
    # (1) 종료 시점 (2) 시작 시점을 우선순위로 오름차순 정렬
    reserved.sort(key=lambda x: x[1])

    answer = 1  # 받을 수 있는 최대 예약자 수
    cur = 0  # 첫 번째 예약자부터 확인
    for i in range(1, N):
        # 현재 커트가 끝난 이후에 다음 예약자가 시작될 수 있는 경우 카운트
        if reserved[cur][1] <= reserved[i][0]:
            cur = i
            answer += 1
    return answer
```

- **Java 정답 코드 예시**

```
import java.util.*;

class Node implements Comparable<Node> {
    public int x;
    public int y;

    public Node(int x, int y) {
        this.x = x;
        this.y = y;
    }

    @Override
    public int compareTo(Node other) {
        return this.y - other.y;
    }
}

class Solution {
    // 예약자의 수(N), 예약자가 원하는 커트 시간 정보가 담긴 배열(reserved)
    public static int solution(int N, int[][] reserved) {
        // (1) 종료 시점 (2) 시작 시점을 우선순위로 오름차순 정렬
        ArrayList<Node> arr = new ArrayList<Node>();
        for (int i = 0; i < N; i++) {
            arr.add(new Node(reserved[i][0], reserved[i][1]));
        }
```

```
        Collections.sort(arr);

        int answer = 1; // 받을 수 있는 최대 예약자 수
        int cur = 0; // 첫 번째 예약자부터 확인
        for (int i = 1; i < N; i++) {
            // 현재 커트가 끝난 이후에 다음 예약자가 시작될 수 있는 경우 카운트
            if (arr.get(cur).y <= arr.get(i).x) {
                cur = i;
                answer += 1;
            }
        }
        return answer;
    }
}
```