# 2580. Count Ways to Group Overlapping Ranges

## Description

You are given a 2D integer array `ranges` where `ranges[i] = [start_i, end_i]` denotes that all integers between `start_i` and `end_i` (both **inclusive**) are contained in the `i^th` range.

You are to split `ranges` into **two** (possibly empty) groups such that:

- Each range belongs to exactly one group.
- Any two **overlapping** ranges must belong to the **same** group.

Two ranges are said to be **overlapping** if there exists at least **one** integer that is present in both ranges.

- For example, `[1, 3]` and `[2, 5]` are overlapping because `2` and `3` occur in both ranges.

Return *the* ***total number*** *of ways to split* `ranges` *into two groups*. Since the answer may be very large, return it **modulo** `10^9 + 7`.

### Example 1:

```
Input: ranges = [[6,10],[5,15]]
Output: 2
Explanation:
The two ranges are overlapping, so they must be in the same group.
Thus, there are two possible ways:
- Put both the ranges together in group 1.
- Put both the ranges together in group 2.
```

### Example 2:

```
Input: ranges = [[1,3],[10,20],[2,5],[4,8]]
Output: 4
Explanation:
Ranges [1,3], and [2,5] are overlapping. So, they must be in the same group.
Again, ranges [2,5] and [4,8] are also overlapping. So, they must also be in the same group.
Thus, there are four possible ways to group them:
- All the ranges in group 1.
- All the ranges in group 2.
- Ranges [1,3], [2,5], and [4,8] in group 1 and [10,20] in group 2.
- Ranges [1,3], [2,5], and [4,8] in group 2 and [10,20] in group 1.
```

### Constraints:

- `1 <= ranges.length <= 10^5`
- `ranges[i].length == 2`
- `0 <= start_i <= end_i <= 10^9`