> 你将会获得一系列视频片段，这些片段来自于一项持续时长为 T 秒的体育赛事。这些片段可能有所重叠，也可能长度不一。
> 视频片段 clips[i] 都用区间进行表示：开始于 clips[i][0] 并于 clips[i][1] 结束。我们甚至可以对这些片段自由地再剪辑，例如片段 [0, 7] 可以剪切成 [0, 1] + [1, 3] + [3, 7] 三部分。
> 我们需要将这些片段进行再剪辑，并将剪辑后的内容拼接成覆盖整个运动过程的片段（[0, T]）。返回所需片段的最小数目，如果无法完成该任务，则返回 -1 。

分析：

1. 需要拼接长度为time的视频，返回最小的数目；
2. 因为要求最小数目，所以每次贪心的选择最长的字段；
3. 贪心问题，首先要进行排序，按起点升序排序，起点相同按照终点降序排序；
4. 如何进行贪心选择？
    - 从<0,0>开始，每次选择当前的起点小于当前的终点，则表示选中当前的视频段；
    - 如何选择下一个视频，对已更新的当前终点，开始循环，如果后续的视频段的起点都小于当前选中视频段的终点，则更新nextend(下一个选中视频段终点，取最大值)；
    - 将nextend赋值给currend, 并判断当前值current是否满足条件。

```
// @lc code=start
class Solution {
public:
  int videoStitching(std::vector<std::vector<int>>& clips, int time) {
    if (time == 0) {
      return 0;
    }

    // 起点升序，起点相同，终点降序
    std::sort(clips.begin(),
              clips.end(),
              [](std::vector<int> a, std::vector<int> b) {
                if (a[0] == b[0]) {
                  return a[1] > b[1];
                }

                return a[0] < b[0];
              });
    int res     = 0;  // 选择的视频数
    int currend = 0, nextend = 0;
    int i = 0, len = clips.size();

    while (i < len && clips[i][0] <= currend) {
      // 在第res个视频区间内贪心选择下一个视频
      // 起点一定小于等于当前的终点
      // 一直到找到一个区间的起点大于大于当前的终点结束
      while (i < len && clips[i][0] <= currend) {
        nextend = std::max(nextend, clips[i][1]);
        i++;
      }
```

```
        res++;  // 选中一个视频
        currend = nextend;
        if (currend >= time) {
          return res;
        }
      }

      return -1;
    }
};
```