Interval Scheduling（区间调度问题）。给你很多形如[start,end]的闭区间，请你设计一个算法，算出这些区间中最多有几个互不相交的区间。

分析：

1. 按照结束时间排序，选择最小的边。

```
int intervalSchedule(std::vector<std::vector<int>> intvs) {
  if (0 == intvs.size()) {
    return 0;
  }

  sort(intvs.begin(), intvs.end(), [](std::vector<int> a, std::vector<int>
b) {
    return a[1] < b[1];
  });

  int count = 1;  // 必有一个区间不相交
  int x_end = intvs[0][1];
  for (auto item : intvs) {
    int startx = item[0];
    if (startx >= x_end) {
      count++;
      x_end = item[1];
    }
  }

  return count;
}
```

不相交区间

```
class Solution {
public:
  int eraseOverlapIntervals(std::vector<std::vector<int>>& intervals) {
    int len = intervals.size();
    if (1 >= len) {
      return 0;
    }

    std::sort(intervals.begin(),
              intervals.end(),
              [](std::vector<int> a, std::vector<int> b) {
                return a[1] < b[1];
              });
    int count = 1;
    int endy  = intervals[0][1];
```

```
    for (int i = 1; i < len; i++) {
      int startx = intervals[i][0];
      if (startx >= endy) {
        count++;
        endy = intervals[i][1];
      }
    }

    return len - count;
  }
};
```

最少的箭击破气球

```
class Solution {
public:
  int findMinArrowShots(std::vector<std::vector<int>>& points) {
    int len = points.size();
    if (len <= 0) {
      return 0;
    }

    std::sort(points.begin(),
              points.end(),
              [](const std::vector<int> a, std::vector<int> b) {
                return a[1] < b[1];
              });

    int endy  = points[0][1];
    int count = 1;
    for (int i = 1; i < len; i++) {
      if (points[i][0] > endy) {
        count++;
        endy = points[i][1];
      }
    }

    return count;
  }
};
```