给你一个数组 rectangles ，其中 rectangles[i] = [xi, yi, ai, bi] 表示一个坐标轴平行的矩形。这个矩形的左下顶点是 (xi, yi) ，右上顶点是 (ai, bi) 。
如果所有矩形一起精确覆盖了某个矩形区域，则返回 true ；否则，返回 false 。

分析：

1. 首先从面积角度，如果构成完美矩形，则完美矩形的面积一定等于所有小矩形面积之和；
2. 为保证面积相同时，避免出现空洞的情况，还需要保证对顶点进行去重之后，所有的顶点在集合中均出现偶数次；
3. 最后为避免顶点重合的问题，需要考虑大矩形的四个顶点一定出现在顶点的集合中。

```
class Solution {
public:
  bool isRectangleCover(std::vector<std::vector<int>>& rectangles) {
    int leftdownx = INT_MAX, leftdowny = INT_MAX;  // 组成大矩形的左下角顶点
    int rightupx = INT_MIN, rightupy = INT_MIN;  // 组成大矩形的右上角顶点

    std::set<std::pair<int, int>> set;          // 顶点集合
    int                           sumArea = 0;  // 小矩形面积和

    for (auto item : rectangles) {
      leftdownx = std::min(leftdownx, item[0]);
      leftdowny = std::min(leftdowny, item[1]);
      rightupx  = std::max(rightupx, item[2]);
      rightupy  = std::max(rightupy, item[3]);

      sumArea += (item[2] - item[0]) * (item[3] - item[1]);

      std::vector<std::pair<int, int>> points = {
          std::make_pair(item[0], item[1]),
          std::make_pair(item[0], item[3]),
          std::make_pair(item[2], item[1]),
          std::make_pair(item[2], item[3])};

      for (auto it : points) {
        if (set.count(it)) {
          set.erase(it);
        } else {
          set.insert(it);
        }
      }
    }

    int area = (rightupx - leftdownx) * (rightupy - leftdowny);
    if (area != sumArea) {
      return false;
    }

    if (set.size() != 4) {
```

```
        return false;
      }

      if (!set.count(std::make_pair(leftdownx, leftdowny))) {
        return false;
      }

      if (!set.count(std::make_pair(rightupx, rightupy))) {
        return false;
      }

      if (!set.count(std::make_pair(leftdownx, rightupy))) {
        return false;
      }

      if (!set.count(std::make_pair(rightupx, leftdowny))) {
        return false;
      }

      return true;
    }
};
```