马上开始

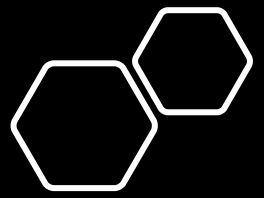
35tang-C++竞赛系列四阶课程

三五堂

《35tang-C++竞赛系列四阶课程》

扫描线和离散化

- 扫描线的一般做法，就是给定了很多个区间，这时候可以把区间的两个点离散化排序，然后按照顺序依次处理这些点。
- 特别多的情况就是一条直线或者一些坐标轴上面的点或者图形等等。
- 扫描线经常需要维护一个区间，队列或者set，或者最大或者最小，前缀和。因为每一次扫描不太可能把前面所有的点都重新遍历一次。



赤壁之战

https://www.luogu.com.cn/problem/P1496

- 先看题，理解题目
- 用测试数据验证你是否真的理解了
- 示例输入

3

-1 1

5 11

2 9

- 示例输出

11

flag计数？记录数轴上面每一个位置是否有区间覆盖。

但是，这里给出的区间端点几乎是最大和最小整数，flag数组做不了。就算用map，还是太大。而且时间不够。比如给你一个区间 0 100000000, 这里面就包含了100000000个位置，如果把这些位置都标记下来需要多大空间？

换个思路，区间的个数n并不是很大，可以扫描区间或者区间的开始结束点。我们不需要知道所有的位置的状态，只需要知道所有连续区间的开始和结束位置。这样数据量小了很多，速度页快。

第一个做法：扫描区间，维护当前活动区间

- 类似前面讲过的奶牛挤奶区间
- 把所有的区间按照开始时间排序
- 遍历区间，维护一个当前活动区间的开始和结束时间：
  - 每一个新的区间的结束时间和开始时间并别和当前活动区间的开始和结束时间去比较，看是否需要更新当前活动区间的开始和结束时间
  - 当遇到区间结束的时侯累加当前活动区间长度，并在下一个区间开始新的区间。

```
3
-1 1
5 11
2 9
```

做法二：

离散化

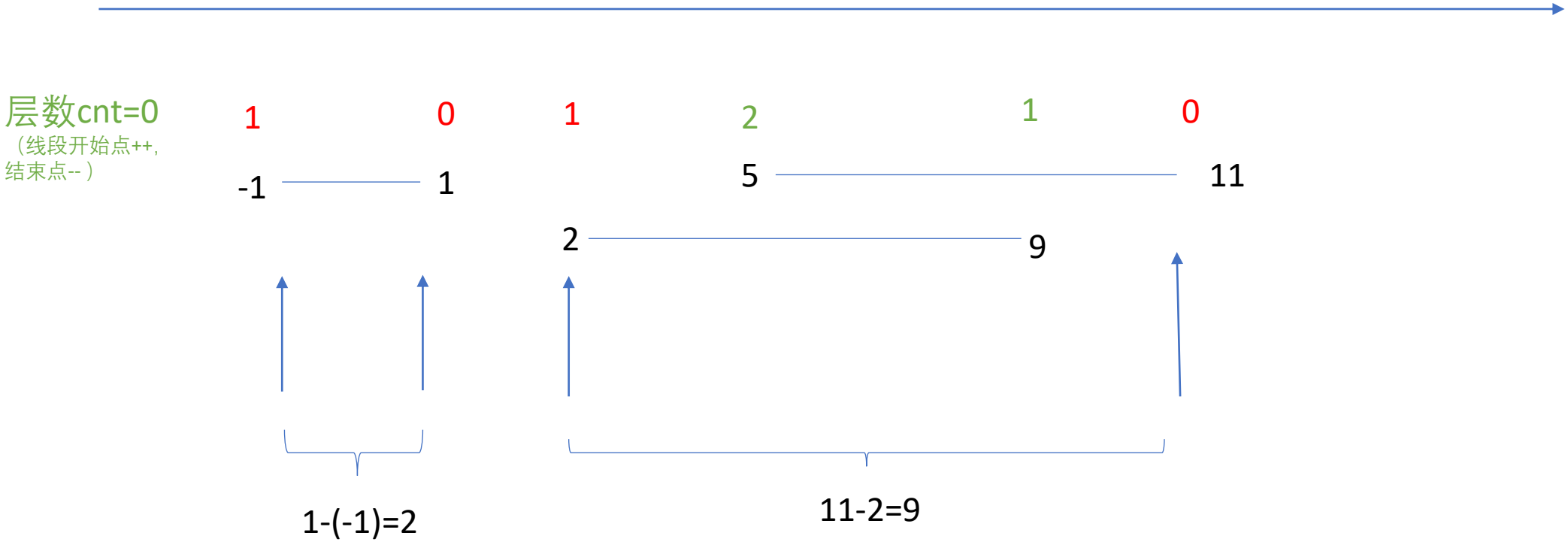
通过扫描变化的点来维护活动区间

- 可以先离散化了，变成2n个点，然后排序，然后挨个点去扫描，扫描过程中维护一个当前区间，以及一个层数cnt。通过层数的变化来判断当前区间是否结束和开始。简单的做就是线段开始点++，结束点--，什么时候从0变成1了开始记录一个start位置，从1变成0了就统计一个区间（当前位置减去start）加到总数去。

```
3
-1 1
5 11
2 9
```

答案是2+9==11；2和9怎么算出来?

层数从0变成1和从1变成0之间的间距全部加起来，从1变成0的时侯就用当前这个点的位置减去前面从0变成1的位置。



标准离散化

- pair<int int> a[40004]; //其实用vector会更方便

//第一个是位置，第二个是1或者-1表示开始或者结束

- 离散

```
for (int i=0;i<=n;i++)
{
        int x,y;
        cin>>x>>y;
        a[i*2]=make_pair(x,1);
        a[i*2+1]=make_pair(y,-1);
}
```

扫描第二步

- 排序
  - sort(a,a+2*n);

扫描第三步

- 扫描（遍历离散化的点）
  - 遍历过程有可能需要维护一个层数cnt，复杂的情况下可能需要维护一个队列，一个数组，或者一个set。

```
for(int i=0;i<2*n;i++)
{
    cnt+=  a[i].second;
    if (a[i].second==1 && cnt==1)//从0加到1的
            start =a[i].first;//新的区间开始
    else if (a[i].second==-1 && cnt==0)//从1减到0的
            r+=a[i].first-start; //区间结束，计算区间累加
}
```

题目说：且答案小于 2^{31} 这告诉我们什么？

思考，如果区间开始和结束重合，也就是出现两个位置一样的节点，一个-1，一个1，会不会因为先处理-1的节点还是先处理1的节点顺序不同导致答案不同？

例如：

3

2 3

3 10

3 9

离散化后6个点，分别是

(2,1), (3,-1), (3,1),(3,1), (10,-1), (9,-1)

也可能是

(2,1), (3,1), (3,1),(3,-1), (10,-1), (9,-1)

模拟一下，跟踪一下程序逻辑，看看结果一样么？

USACO 2017 January Contest, Silver Problem 1. Cow Dance Show
https://www.luogu.com.cn/problem/P3611

先看题，理解题目

用测试数据验证你是否真的理解了

手工验证示例数据

- N个奶牛都要跳舞，跳舞需要时间。如果舞台大小是K（同时K个奶牛在上面），K个奶牛中谁先跳完了，排队的奶牛中的第一个就会上来跳。那么K不同，显然最后所有奶牛跳完舞需要的时间不同，而且K越大，这个需要的时间越小。现在给定了所有奶牛跳完的时间限制，问K最小是多少。注意：不是让时间短的先跳，而是按照1到n的顺序依次安排。
- 示例输入

5 8

4

7

8

6

4

- 示例输出

4

题目说k最小是多少，现在如果最小是4，也就是说3不行，5，6…都可以，我们来验证为什么3不行，为什么4，5，6可以，注意过程中你是怎么验证的？

最值问题：二分+贪心

- 但是现在k不知道，所以就是去尝试k，最值问题想一想二分法。而且我们知道k越大越容易满足条件，所以单调的，用二分。每一次尝试一个k，如果这个k可以就去尝试更小的k，因为比他大的肯定可以；反之，如果k不行，就去尝试大的k，因为小于k的肯定也不行。
- 二分中判断每一个K是否可行的函数怎么写：贪心。就是计算台上同时K个奶牛，最后奶牛都跳完舞的总时间是否满足条件。最优的做法： k头在台上的奶牛就相当于k个水管，对这k个位置最优的安排奶牛来跳舞，k个位置中最晚结束的时间就是所有奶牛跳完舞需要的时间。对于后面的n-k个奶牛，每一次取台上结束最早的水管（奶牛）把后面排队的第一个的时间加上去。需要两个队列，一个是保存当前台上k个位置每一个位置上奶牛结束的时间，一个保存排队的奶牛（接水的同学）。优化数据结构：台上的队列每一次需要找最小，所以可以用优先队列（这样比以前用数组时间函数从O(N)下降到O(logN)）；台下队列按照顺序取出来，所以普通队列（数组也可以，队列更方便）就好了。
- 问：第一个台上队列，每一次变化之后要找最小，还可以用什么数据结构？

具体算法逻辑

- 读入每一个奶牛需要的时间到数组
- 二分每一个k，对于每一个k，用checkok函数检查这个k是否满足条件
- checkok函数算法
  - 初始化台上优先队列，把前k个奶牛需要的时间入队（自动排序，最小时间在最前面）
  - 初始化台下队列，把没有上台的n-k个奶牛入队
  - 循环处理一直到台下队列为空
    - 取出台上队列的队头（也就是当前最早结束的），取出台下的队头（排队的第一个），二者相加入（新的结束时间）入台上队列
  - 统计台上队列中（k个位置的结束时间）最大的，也就是结束时间最大的，和t比较， 判断当前k是否满足条件

用示例数据验证你的checkok函数

- 示例输入

5 8

4

7

8

6

4

模拟一下，如果k是3怎么计算的，k是3最优的安排下，全部奶牛结束跳舞的时间是多少，是否小于等于8?

注意

- 优先队列缺省从大到小排序
  - priority_queue <int> onq;
- 如果要从小到大呢?
  - priority_queue <int,vector<int>, greater<int> > onq;

主程序：二分模板

```
int main()
{
	int r=0;
	cin>>n>>t;
	for (int i=1;i<=n;i++) cin>>d[i];

	//二分
	int start=1,end=n,mid;
	while (start<=end)
	{
		mid=(start+end)/2;
		if (checkok(mid) )
		{
			r=mid;
			end=mid-1;
		}
		else start=mid+1;


	}
	cout<<r<<endl;
	return 0;
}
```

checkok：

模拟

```
bool checkok(int k)//check台上k个牛行不行
{
	priority_queue <int,vector<int>,greater<int> > onq;
	//初始化台上优先队列，从小到大的顺序
	queue <int > offq;//台下的队列先进先出
	for (int i=1;i<=k;i++) //前k的奶牛入队
		onq.push(d[i]);//优先队列自动排序
	for (int i=k+1;i<=n;i++)
		offq.push(d[i]);//剩下n-k个奶牛按照编号顺序依次入队，先进先出，所以后面取的时侯编号的小的先去出来

	//一个一个取出台上最小的和台下排队的第一个，加起来重新入台上的队列，也就是更新最早结束位置的结束时间
	//onq这个优先队列大小一直是k，里面保存的是每一个位置结束的时间
	while (!offq.empty())
	{
		int newt=offq.front();
		newt+=onq.top();
		offq.pop();
		onq.pop();
		onq.push(newt);
	}
	//查看台上的k个位置中的最大结束时间 是否超过。
	while (!onq.empty())
	{
		if (onq.top()>t) return false;
		onq.pop();
	}
	return true;
}
```

USACO12FEB　Overplanting S

https://www.luogu.com.cn/problem/P1884

- 先看题，理解题目
- 非常难！开拓一下思路，尝试理解
- 用测试数据验证你是否真的理解了

- 在一个笛卡尔平面坐标系里（则X轴向右是正方向，Y轴向上是正方向），有N(1<=*N*<=1000)个矩形，这些长方形的边分别平行于x轴和y轴第i个矩形的左上角坐标是(x1, y1),右下角坐标是（x2,y2）。问这N个矩形所覆盖的面积是多少？注意：被重复覆盖的区域的面积只算一次。

分析

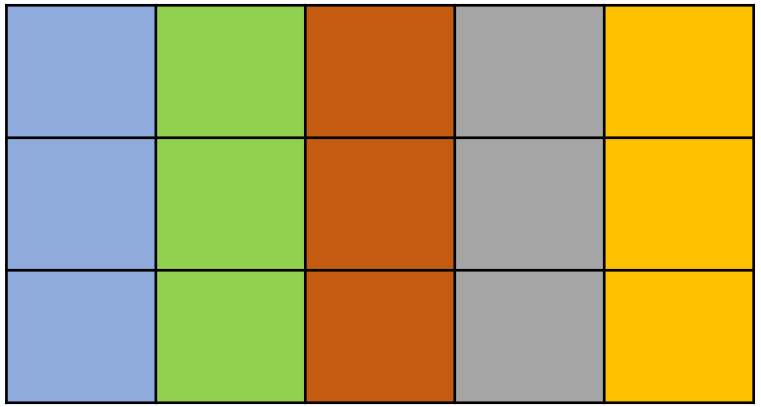
- N<=1000。所以O(N*N)的算法是没有问题的。
- 如果按照x横坐标排序，那么我们是可以找到每一个矩形可能相交的其他所有矩形的，无非是大小x的包含关系
- 但是这么多矩形互相包含的关系互相嵌套，有点乱了
- 由于是坐标，有顺序，是不是可以离散？类似前面的思路，每一个矩形都有左右两个x，拆成2个点？但是拆了之后剩下的是一个线段!

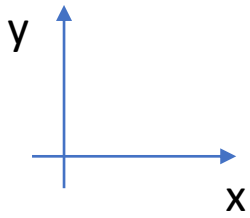
理论依据

一个长方形的面积可以是多个纵向线段的长度（纵坐标的差）*1组成的

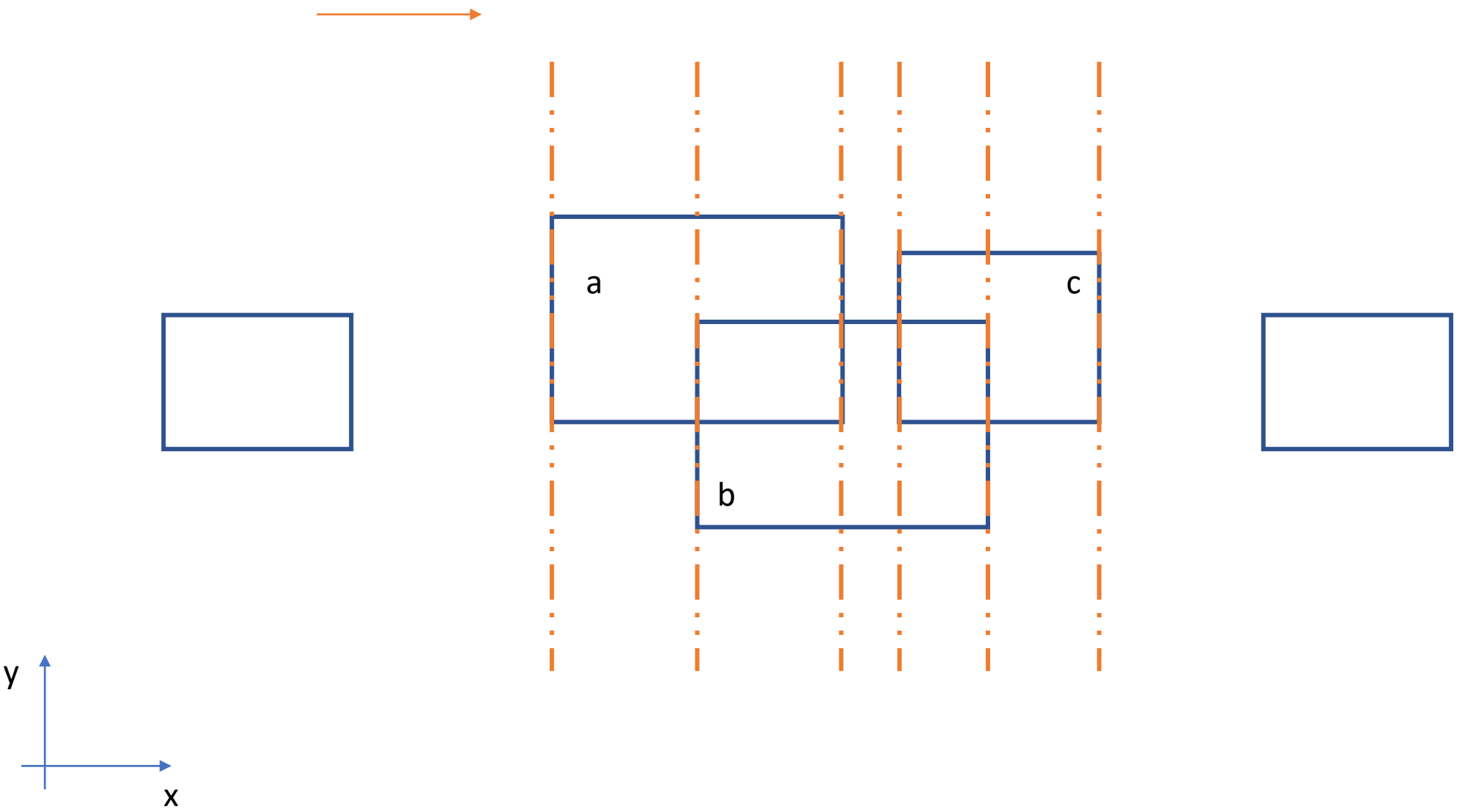
多少个线段呢？

横坐标的差

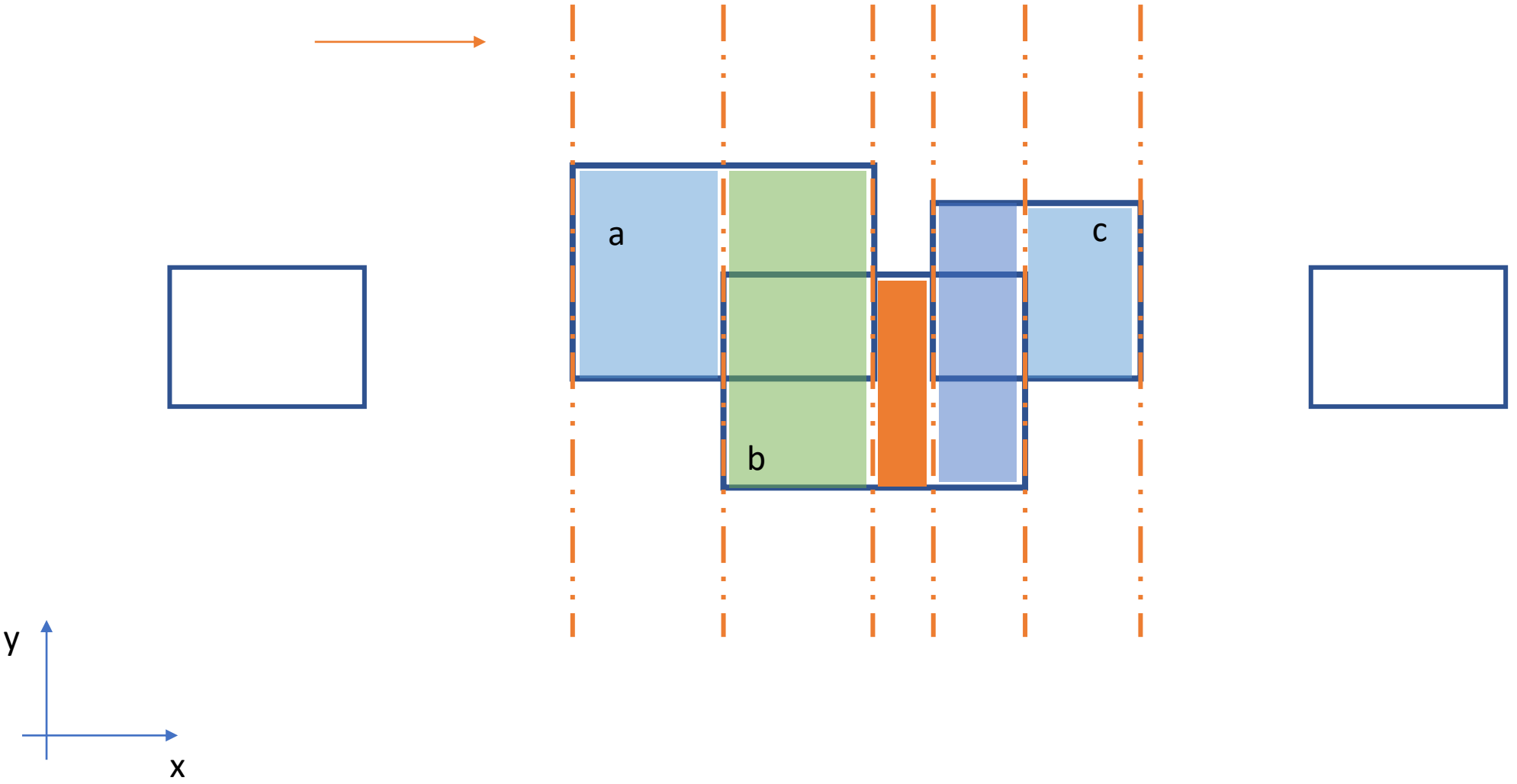




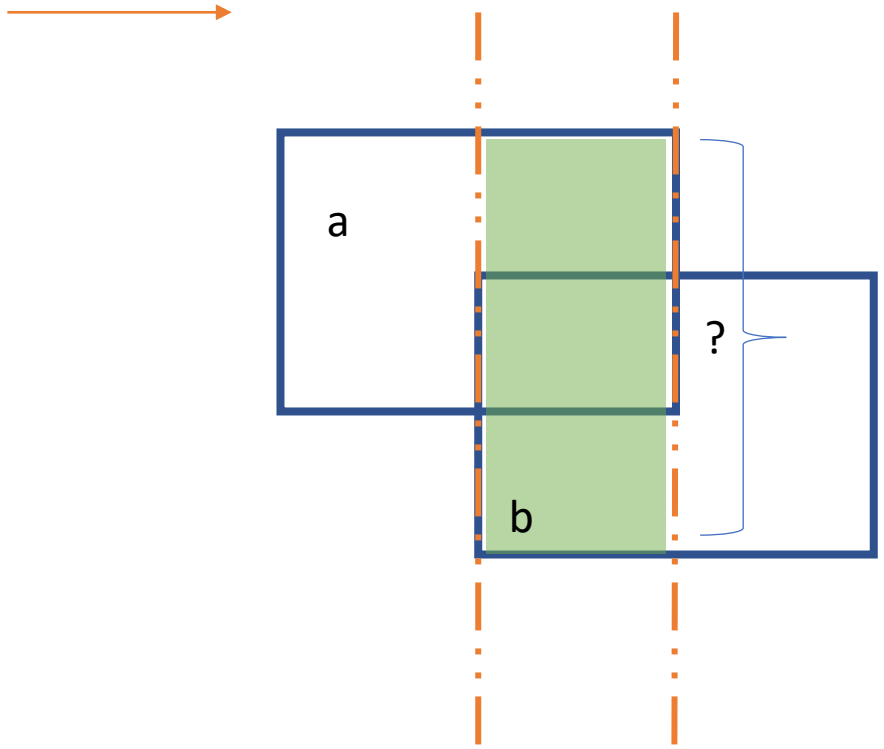
- 和坐标正负没有关系，因为计算面积算的是边长，是两个坐标的差
- 从前往后看，每一次面积发生变化只会在遇到新的x（纵边）的时候
- 如果x离散化后排序，按照从前往后的顺序去看，我们先看看abc三个长方形的情况

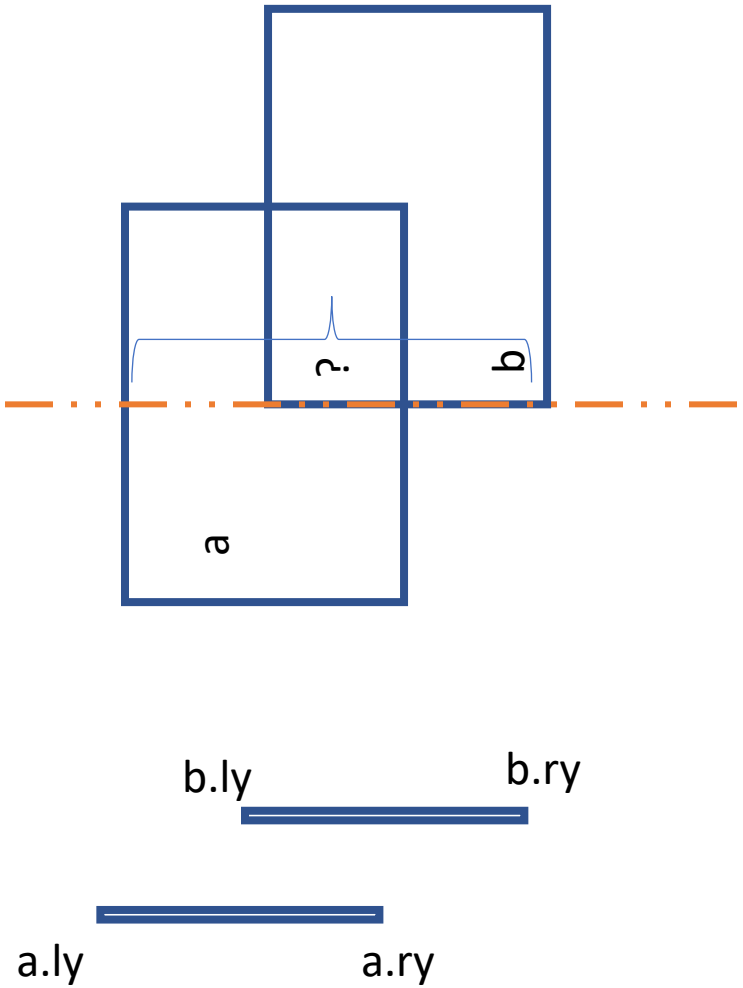


- 如果x离散化后排序，按照从前往后的顺序去看（扫描），前后2个x之间就是一个需要计算面积的长方形，横的边长就是2个x的差值，纵的边呢？就是前面没有关闭的y线段的和。
- 假设现在x扫描（遍历操作）到了第3条橙色线，现在需要计算绿色的区域面积，首先可以发现，因为当前这个线的x落在a和b两个矩形里面，所以只和矩形a，b发生关系。



- 假设现在x扫描（遍历操作）到了第3条线，现在需要计算绿色的快的面积，就是横边*纵边。首先可以发现，因为当前这个线的x落在a和b两个矩形里面，所以只和矩形a，b发生关系。这时候的横边长很容易就是两条橙色线所代表的x的差。
- 怎么计算纵的边长？倒过来看？





长方形a的纵边和b的纵边是两个线段，每一个线段的两个点就是长方形的两个y坐标。这就变成了前面赤壁问题了，是在找第二个x（第二条橙色线）上重叠的y线段的长度。

扫描线算法

- 先离散了x坐标，变成2n个点（或者说2n个线段，每一个线段有自己的最小y和最大y，是一个线段实际上）
- 排序
- 扫描每一个x（遍历x）
  - 遍历所有长方形，找到x相关的长方形（x落在长方形的lx和rx之间的长方形），把这些长方形的ly和ry作为点离散化了
  - 对该x相关的这些y点排序，用离散化的方法寻找重和线段的长度（类似赤壁问题，ly的点+1，ry的点-1，最后看1和0之间的长度）

- 这里相当于双离散化扫描，先离散化了x，然后扫描每一个x，相关的y再离散化扫描。
- 对于每一个x的所有y（就是和当前x坐标相关的长方形的所有y坐标），可以用数组，塞进去再排序，也可以直接使用multiset，插入的时候就排好序。大不了就是对于每一个x值（每一个长方形的两个x），遍历所有长方形，看看这些长方形的小x和大x是否包含当前的x，如果包含，这个长方形的两个y值就要放到这个x所属的set或者数组里面。这个找y的过程可以放到离散化处理每一个x的逻辑里面，也可以在处理x之前提前做预处理。

一起看看程序

- 课后研究给出的程序，尝试理解算法

小tips

- n个数，怎么找出最大（最小）的k个数?
- 如果数字小，类似上节课的例题，计数排序。
- sort排序后，取前k个，算法复杂度为n*log(n);
- 维护大小为k的优先队列，比如找最小的k个：先把头k个放到队列，这个队列队头是这k个元素最大的，对于剩下n-k个元素，每一个元素和这个优先队列队头比较，如果新元素大于等于队头，扔掉，否则pop队列头，把这个新元素入队列，算法复杂度为n*log(k)。

下节课预习
不需要做，争取理解题目，思考一下可能的算法

- USACO 2017 US Open Contest, Bronze, Bovine Genomics
  - 英文看不懂可以借助翻译软件，也可参考下面的大概描述：
  - 每个奶牛有一个长度M的串，该串只有四个字符组成（ACGT），给出有点和没有点的两种奶牛各N个牛的串，看看字符串中多少个位置可以唯一识别奶牛是有点或者没有点的。
  - 一个位置可以唯一识别的意思是这个字符只在有点或者没有点的奶牛中出现，在另外一种没有。
- USACO 2017 US Open Contest，Sliver　Bovine Genomics
  - https://www.luogu.com.cn/problem/P3670

复习作业1

选做：练习pair,vector和sort

给定n对数字，如下格式，请用pair数据结构存放每一对数字，并把这些对数字存入动态vector数组，然后按照两个数字的和进行排序，如果和一样，按照第一个数字排序。从小到大排序。要求用sort函数。

输入格式：第一行位整数n，表示n对整数，后面n行，每行两个数字；

输出格式：n行，排序后的n对整数。

输出样例

```
5
4 5
100 99
99 100
100 88
3 5
```

输出样例

```
3 5
4 5
100 88
99 100
100 99
```

挑战作业2

- USACO 2013 January Contest, Silver Problem 1. Painting the Fence
- https://www.luogu.com.cn/problem/P2205

  离散化和扫描线

作业提示

- 作业1：如果忘记了vector怎么用，怎么办？
  - 回到前面的课程，寻找例子。
  - 到 http://www.cplusplus.com/reference/ 去搜索vector，然后去查看vector的方法，比如push_back，可以看到例子程序
  - 上网搜索“C++ vector的用法”
  - 对于其他的也可以这么办。CSP考试的时候不能上网查，所以平时练习查看后需要彻底掌握，记住了。
- 作业2：和课堂赤壁问题非常类似。首先分成n段，有开始和结束。然后离散成2*n个点，排序，一个一个点扫描，遇到线段开始点就层数+1，反之-1. 什么时候层数加成k了就记下一个开始点的坐标，什么时候层数小于k的时候就用这个时候的坐标减去前面开始点的坐标。和赤壁问题的区别在于赤壁问题统计的是所有层数为1的线段和，这里统计的是层数为k的线段和。

由易到难，思维体系训练
实战结合，创新协作培养
兴趣导向，未来职业引领

https://www.35tang.com

https://www.三五堂.com

扫码关注公众号

添加辅导老师