ith order statics:一个集合里n个元素里第i小的element

median:一个集合里的中位数

当n时奇数，median是(n+1)/2

当n是偶数，lower median是n/2

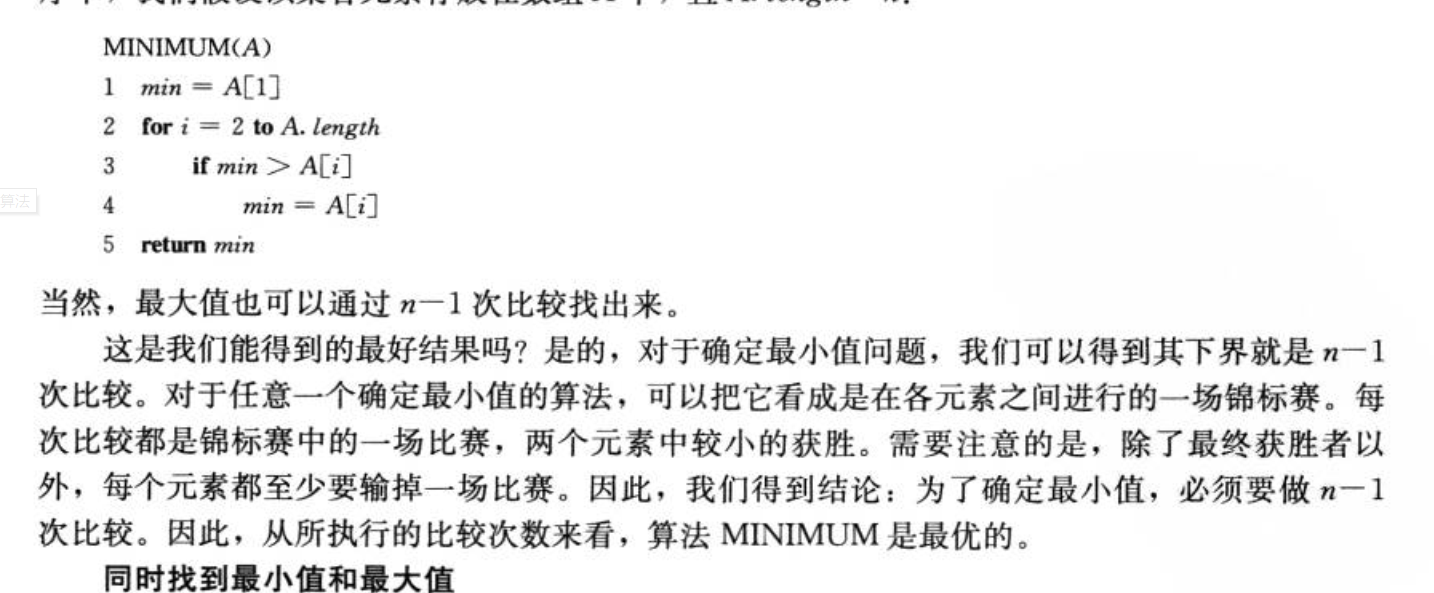
upper median是n/2+1

Selection Problem，给你一个有个n个不同元素的array，找到ith order statistics

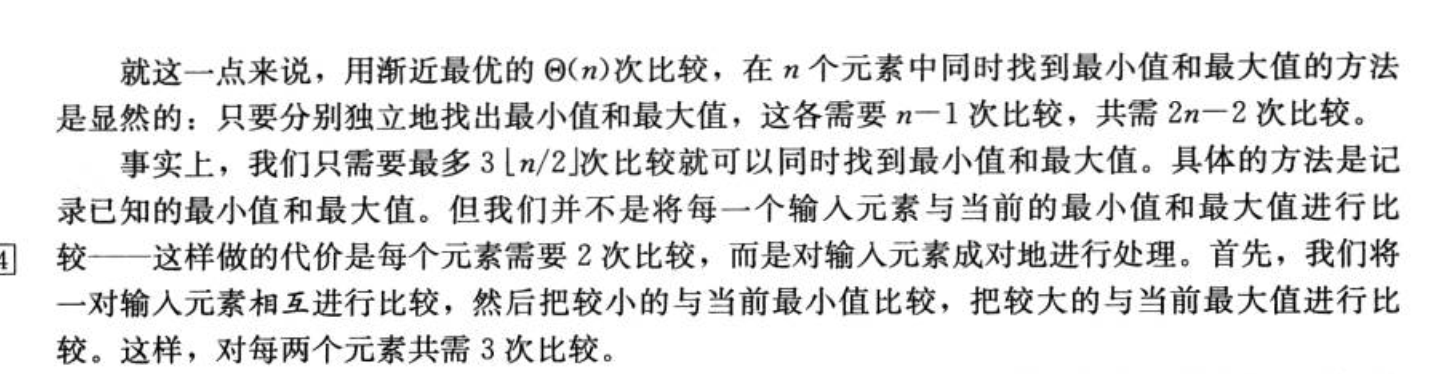
第一种思路，直接sorting整个array,那么ith order自然就是index i，这种方法需要Onlogn

但是关于这个问题，实际上我们能够达到On，通过divide and conquer

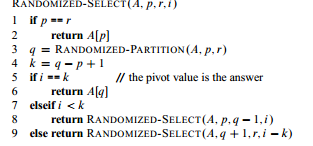
9.1 MAXIMUM AND MINIMUM

t

同时大小可以是2n-2，但可以更优化



Randomized-select



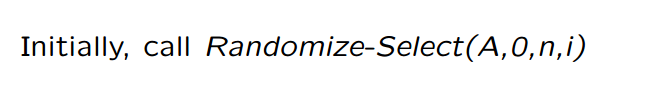
3：randomized-partition是前一章的那个，能够随机选一个pivot，小的在左边大的在右边，q是pivot最终index

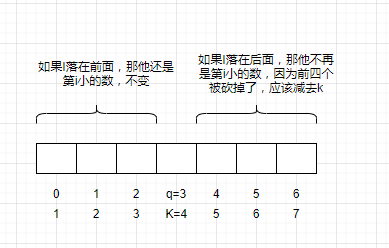
4:k=q-p+1,就是index这个Pivot实际上是第几小的数，例如array有六个数，他是第三小的数，那么他index就是2， 2-0+1=3,也就是他是第三小的数//而且这样可以是相对的位置，q是index，而p是subarray的开头

8：如果在左侧，舍去包括q在内(q到结尾)的所有array

9：如果在右侧：舍去包括q在内的所有array(从q+1开始，r不变)，而且，我们要找的不再是i，因为舍去了这么多数，他不再是第i小的数，i-k

一开始call这个





最差情况是O（n^2）,每次随机选取都是最小或最大，

然后T（n）=T(n-1)+O(n) //pivot所以会减一

那么我们怎么算出来期望时间呢

我们有很多种可能，有可能对半分，有可能19分....，为了表达不同的情况，我们使用Indicator random variable,

Definition

Indicator Random Vaiables: A random variable that has the value 1 or 0, according to whether a specified event occurs or not is called an indicator random variable for that event.

**T(n)：**我们不知道输入array,所以我们把n个元素的输入数组的运行时间T(n)看作**随机变量**



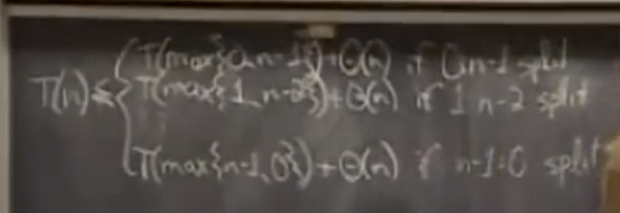
randomized-partition能够等概率的返回任何element作为pivot，因此子数组A[[p,q]有k个元素（全部小于或等于pivot）的概率是1/n，

Xk: k的范围是从0到n-1,如果partition切割后左边的数组长度等于k，为1，//换句话说，array被切割成，k，与n-k-1两段

否则为0

定义一个Indicator Random Variable Xk,紧当子数组正好包含k个元素的时候，为1，其他为0（用来标记不同的情况）

然后我们可以根据T（n）与Xk，分析不同情况下运行时间



那么不同情况下T（n）的值就是这样

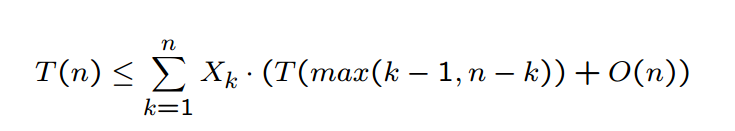
注意max是什么意思，我们要求最坏情况也就是upper bound，那么我们需要让ith order落在比较大的子array，

例如0，n-1切割的时候，(0,n-1)的max是n-1,我们希望它落在后半段，

那么对应的T（n）=T(n-1)+On

以此类推//max的作用主要是中间那一部分，

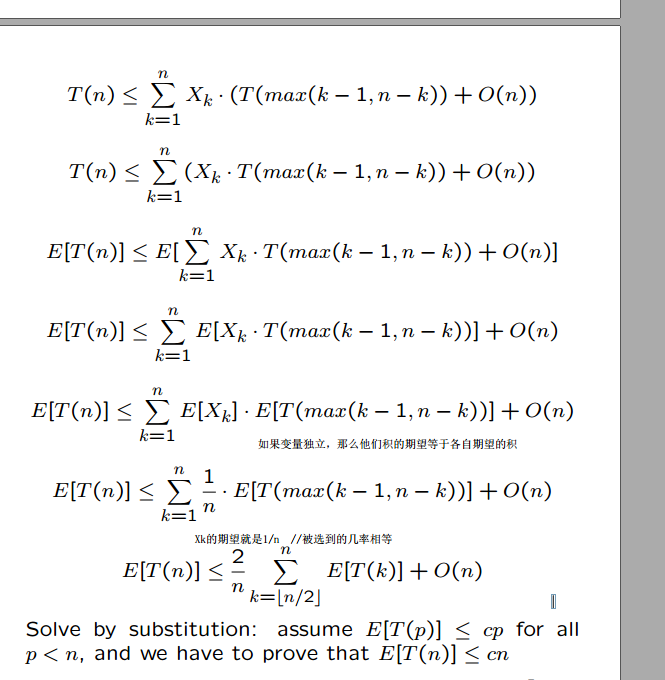
这样我们就得到了不同分割对应的情况



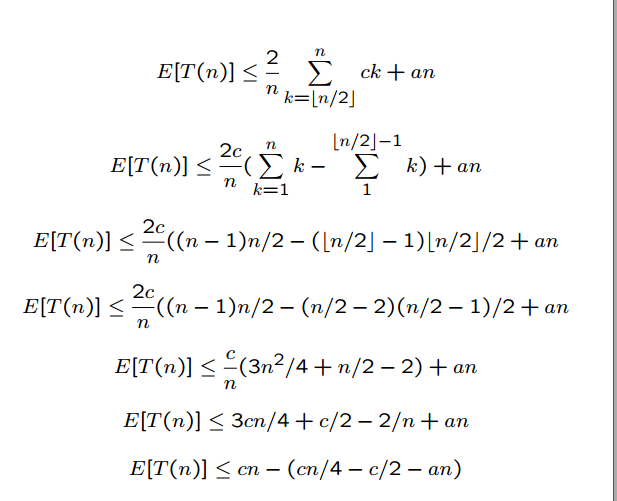
然后用Xk，指代各种不同情况//就是1，不会影响值

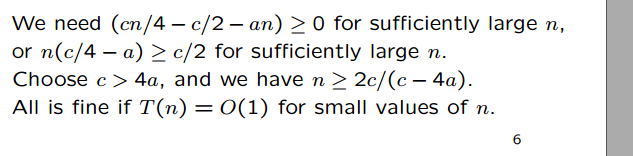
然后把各种情况用时相加 //每种情况几率是相等的

E[T(n)]就是T(n)的期望值



这里E（T（n））就是多个T（n）的期望，对他使用substitution法





当c>4a成立

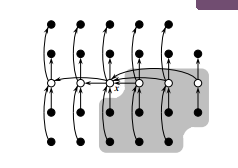
这时

9.3：线性时间的order stastics

第一步，把array所有元素排成5\*n/5网络



每一列称作一个group，最后一列可以不满五个，



第二步：找到每个group（列）的中位数，把这个goup重新排列，中间的就是median,需要的用时就是 O(n), 因为array5的比较时间是固定的，是O（1）,乘上n/5就好



第三步：使用SELECT,找这群median的median X，需要T（n/5）的时间

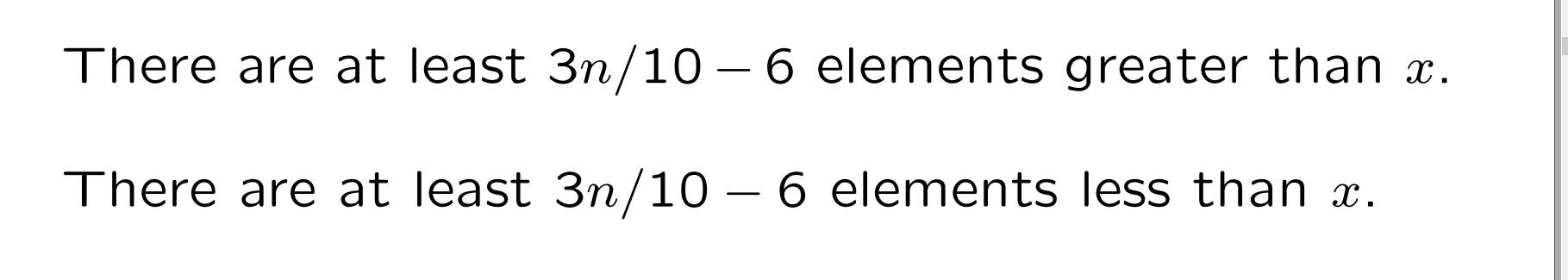


第三列

第四步，把这个最终中位数x作为pivot，开始切割 //以上四步就是每次选pivot的标准

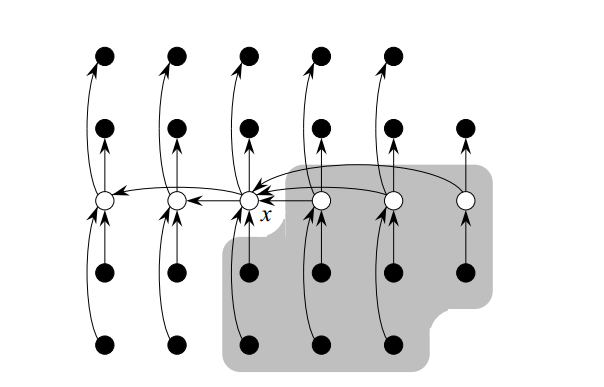
第五步：然后无限循环

这样的算法好处在于，



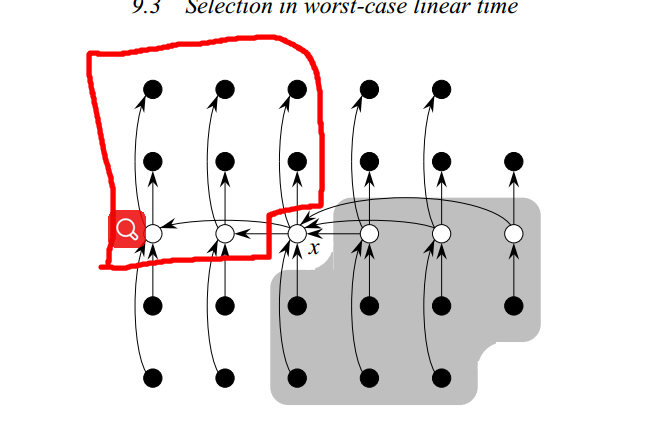
可以尽可能的取到中间的位置

为啥

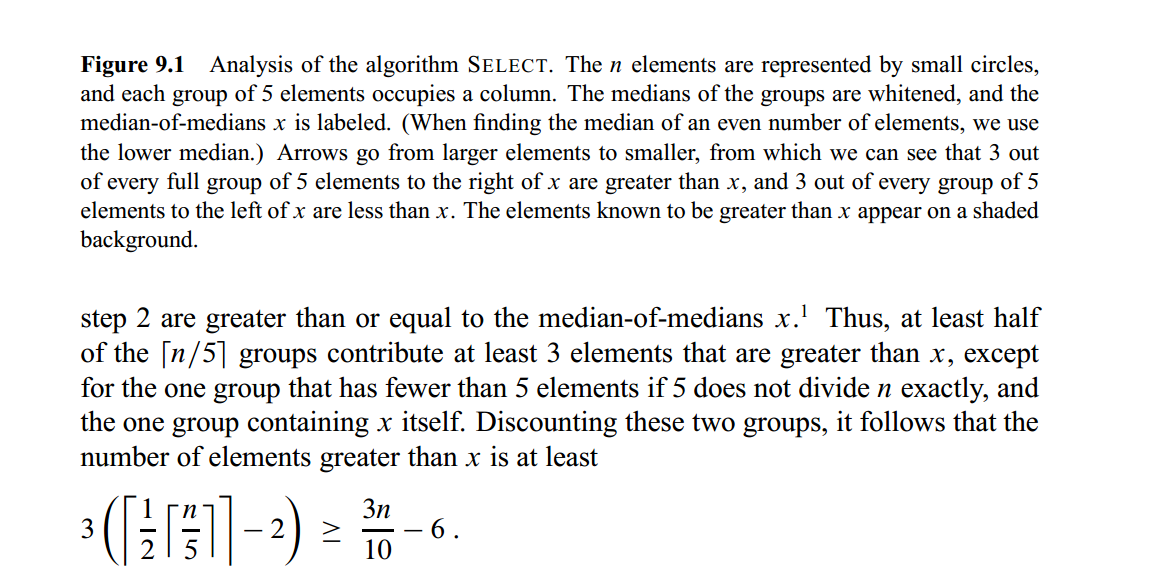


箭头起始点代表大的，终点代表小的

那么根据箭头的延续，红色区域内的element小于x, 黑色区域内的大于x， 而不在红色区域和黑色区域的白色区域不确定

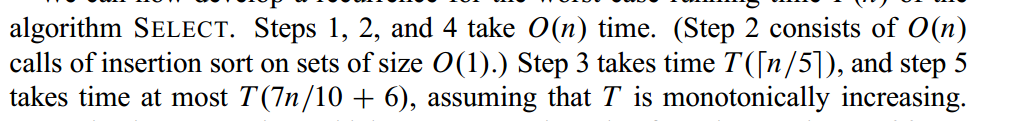


这是一列的3/5

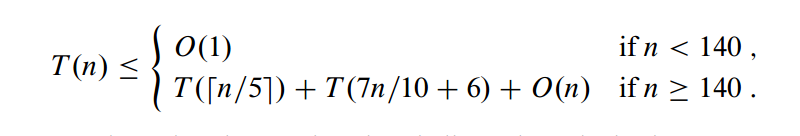


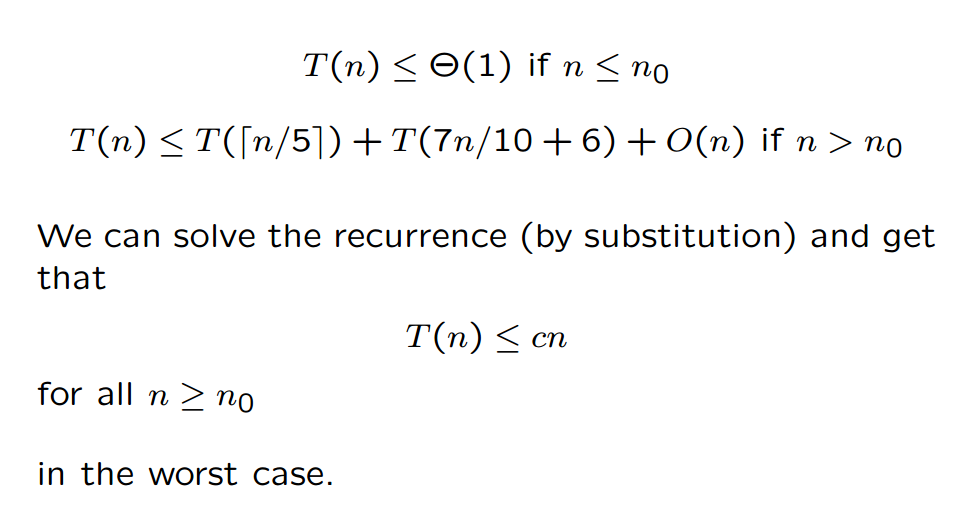
n/5代表有几个group， 一半的group大于x，减二是因为有一列包含了x,有一列不完整。

3代表着group里有三个元素能取到



4



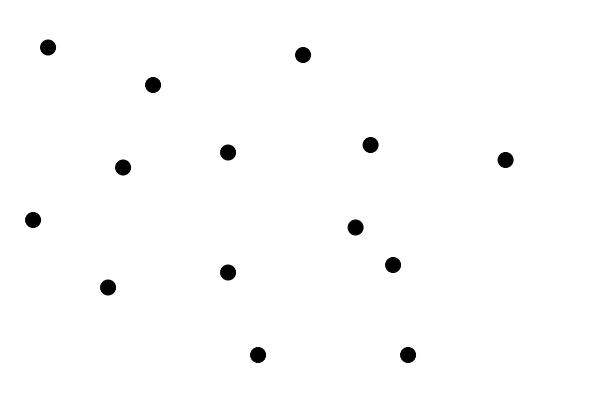


这个算法把问题分解成了一个选中位数问题加上更小的recursive问题

T n/5就是取中位数用时， T（7n/10+6）是剩下的部分，O（n）Step124

33.4 Find a closest pair of points

找距离最小的两个点



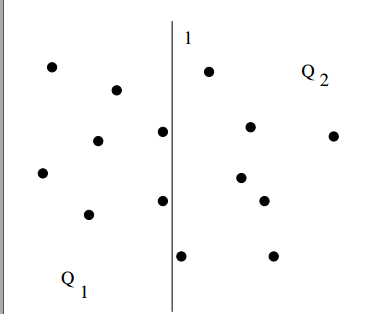
有n(n-1)/2配对法，暴力解可以在n^2内解决

但是我们用divide and conquer可以做到nlgn内

第一步：divide，

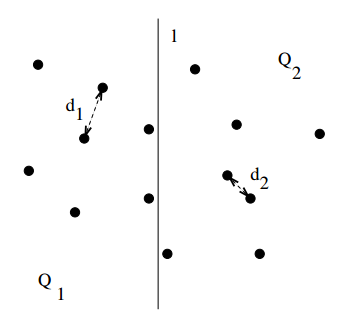
用一根vertical line，把所有点左右均分，左边可以多一个如果是奇数





第二步：找到左半区右半区最短的距离//这一步是recurrence，他把问题一切为二了

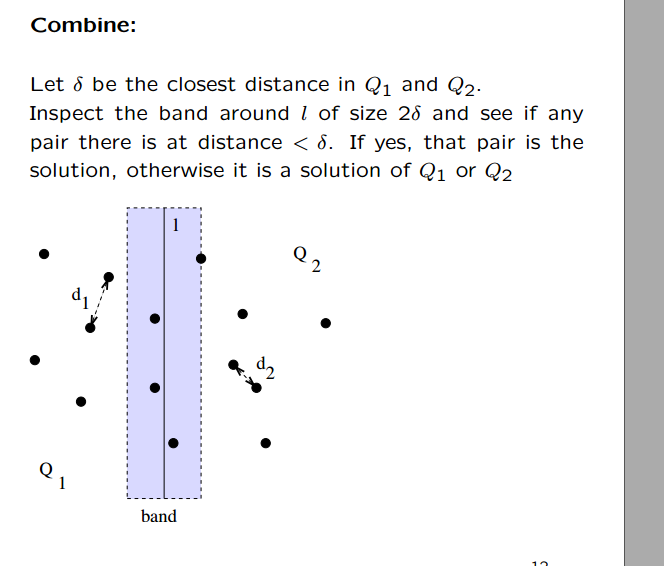
conquer



把最短的距离记作 , 

第三步

以vertical为line为中心轴，比较2内是否存在pair距离小于，如果是，取这个pair，如果不是，取



为了得到nlogn，我们需要得到



2T（n/2）是我们的第二步recurrence

我们需要证明第一步·和第三部都是O（n）

首先我们要创造一个X array与一个Y array

X array存放sorted X坐标

Y array存放sorted Y 坐标

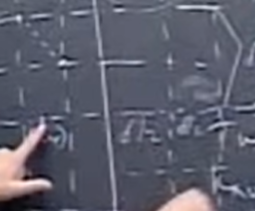
X\_array让第一步垂直切割非常简单，你直接取中间的index就知道这条线的位置了

Y\_array用来为第三步combine服务

创建sorted array本身总体上看需要nlogn，

为什么第三步是n

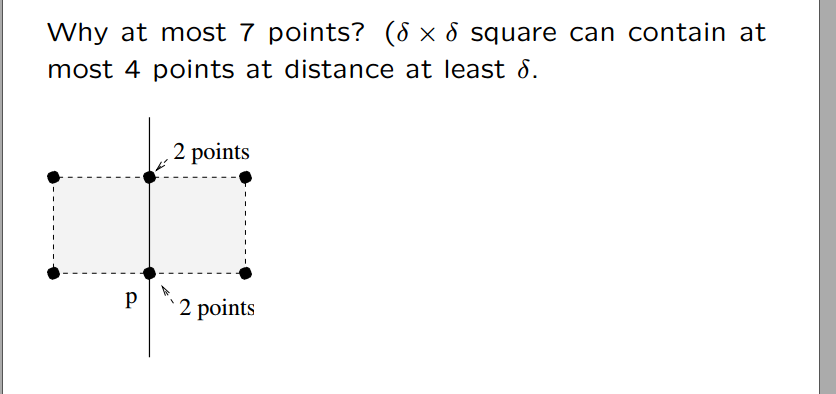
第三步考虑的是一个在L左边，一个在L右边的情况，



以左手指的这个点为例（一个虚线是1/2最短距离）

右边只能存在最多七个点，因此是有限的，因此是n

为什么，因为假如右边有n个点，那么最短距离就不是theta了



这个点越靠近L线，可能性越多，假设无限接近L线点的时候，这个点是7个

记录最小的点，最多O（n）时间

