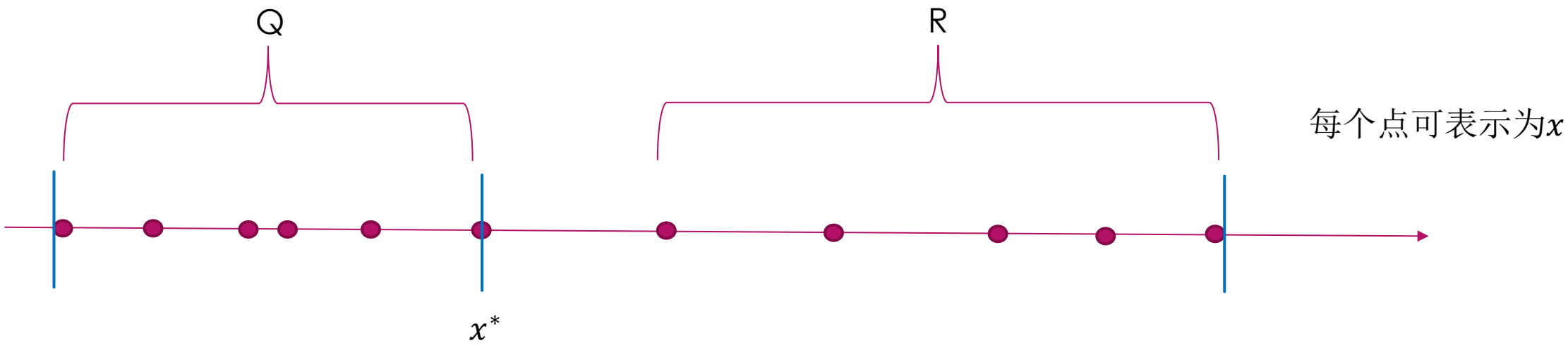
最邻近点对

赵耀

一维最邻近点对



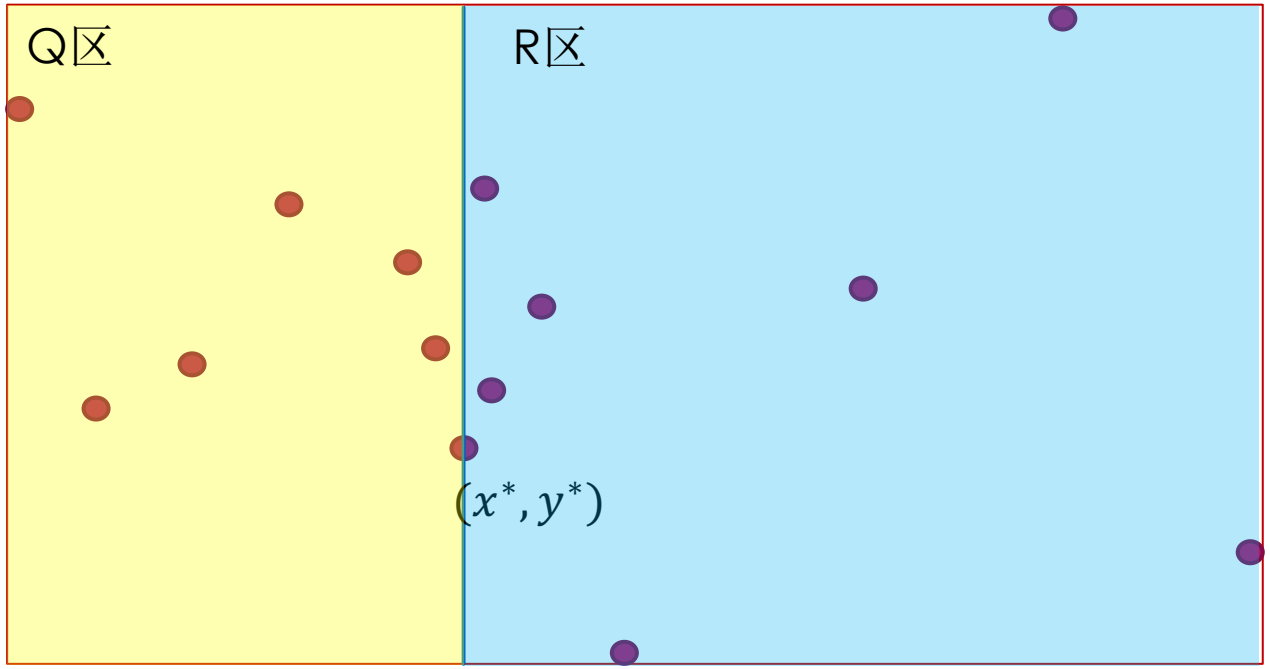
一维最邻近点对算法流程

输入：一维点序列P
输出：P中距离最小的一对点

```
Closest-Pair(P){
    Px =  对P升序排序
    pair(p0,p1) = Closest-Pair-Rec(Px)
    return (pair(p0,p1))
}
```

```
Closest-Pair-Rec(Px){
    If Px.size <=3
        如果是3个点，比较d(p0,p1) 和d(p1,p2)，返回较小的点对
        如果是2个点，直接返回该点对
    Endif

    将P划分为2个子序列Qx，Rx
    pair(q0,q1) = Closest-Pair-Rec(Qx)
    pair(r0,r1) = Closest-Pair-Rec(Rx)
    δ=min(d(q0,q1),d(r0,r1))
    x* = Qx中的点最大的x坐标
    s*为Rx中第一个元素
    If (d(x*,s*)< δ) return pair(x*,s*)
        Else if d(q0,q1) < d(r0,r1) return pair(q0,q1)
        Else return pair(r0,r1)
    Endif
}
```

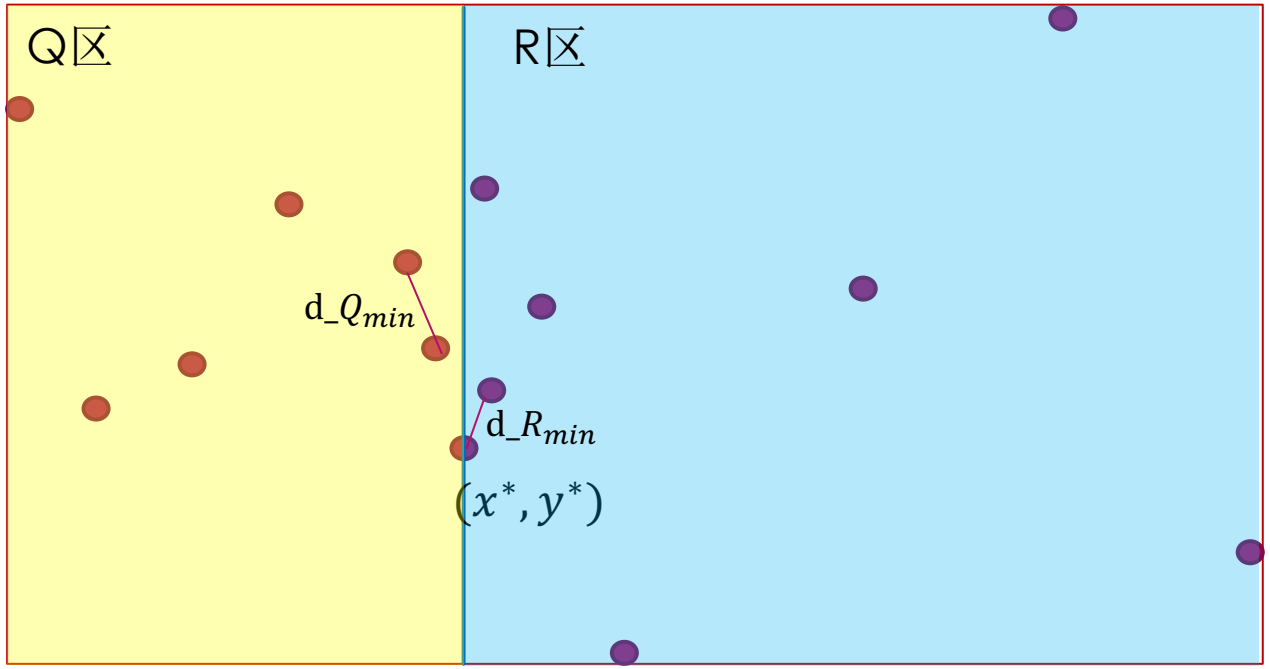
二维最邻近点对



每个点可表示为(x, y)

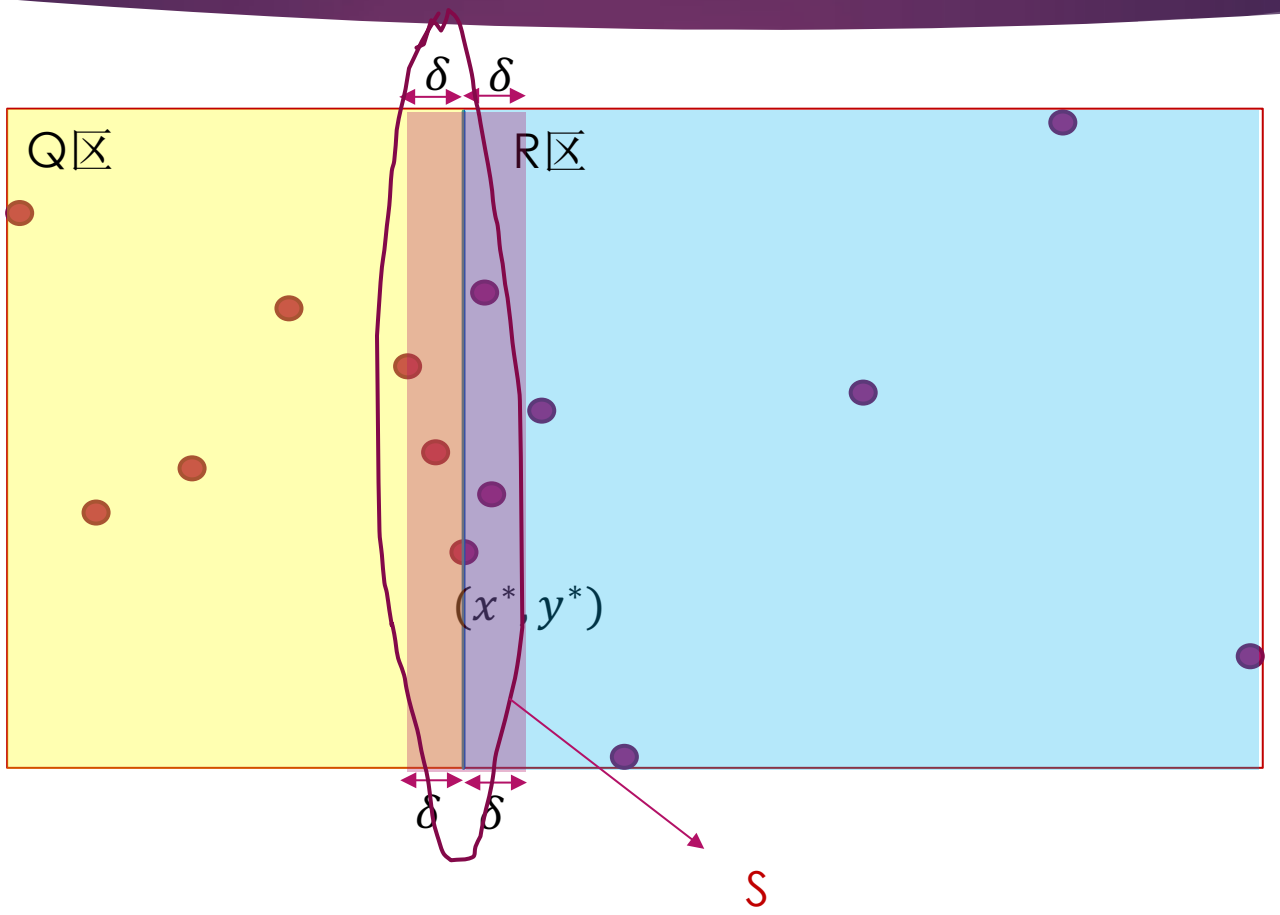
刚开始根据x的值取中间值用$x = x^*$这条线将点集分成2部分

二维最邻近点对



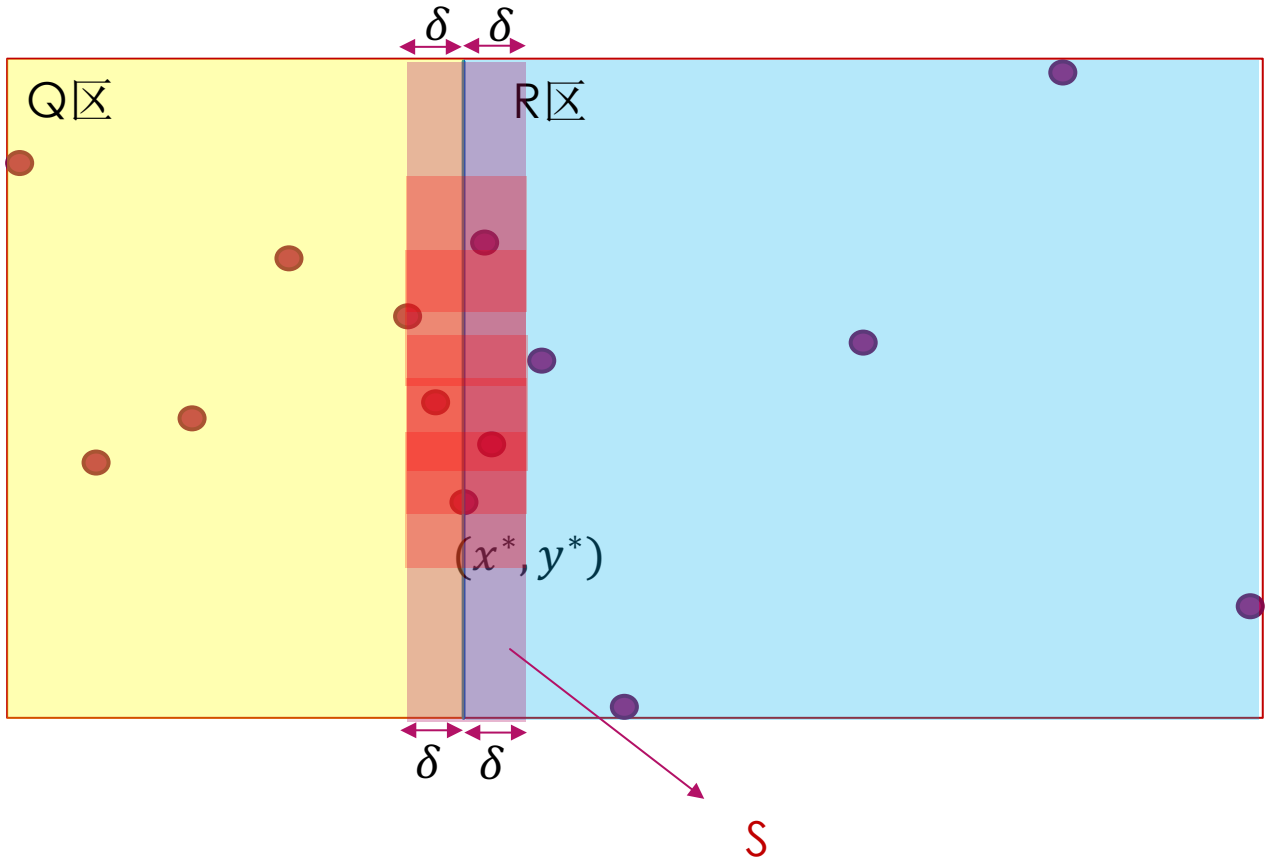
$$\delta = \min(d\_Q_{min}, d\_R_{min})$$

二维最邻近点对



$$\delta = \min(d\_Q_{min}, d\_R_{min})$$

二维最邻近点对

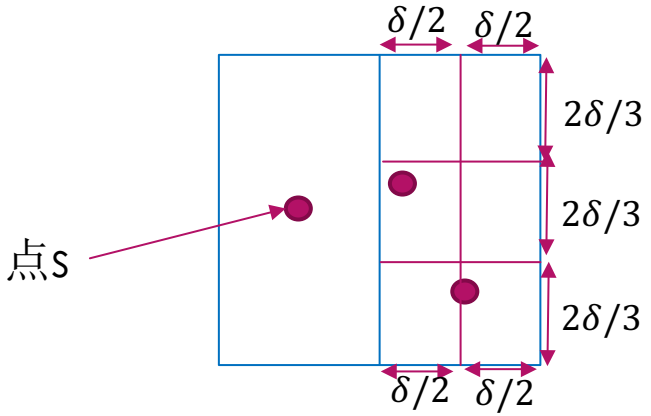


$$\delta = \min(d\_Q_{min}, d\_R_{min})$$

对于任意一个集合S中的点s，求出这个点到附近其他点的距离的最小值

怎么定义这个点附近的其他点？
其他点的y值与这个点的y值相差不超过δ

附近的点最多有多少？



假设有点s位于线L的左边，则与点s相邻的点必定要在线L的右边的R区找，而找寻的范围，在点s的坐标y的上下δ区间才可能存在比δ距离更临近的点。将右边的矩形分解成6个盒子，可以证明每个盒子中最多存在1个点。

如果一个盒子存在2个点，那么这两个点的距离<=$\sqrt{(\delta/2)^2+(2\delta/3)^2}=5\delta/6<\delta$,与$\delta$是Q区和R区最小的距离矛盾。

如果s在线L的右边，同理。

如果s在线L上，同在左边的情况。

二维最邻近点对算法流程

输入：二维点序列P
输出：P中距离最小的一对点

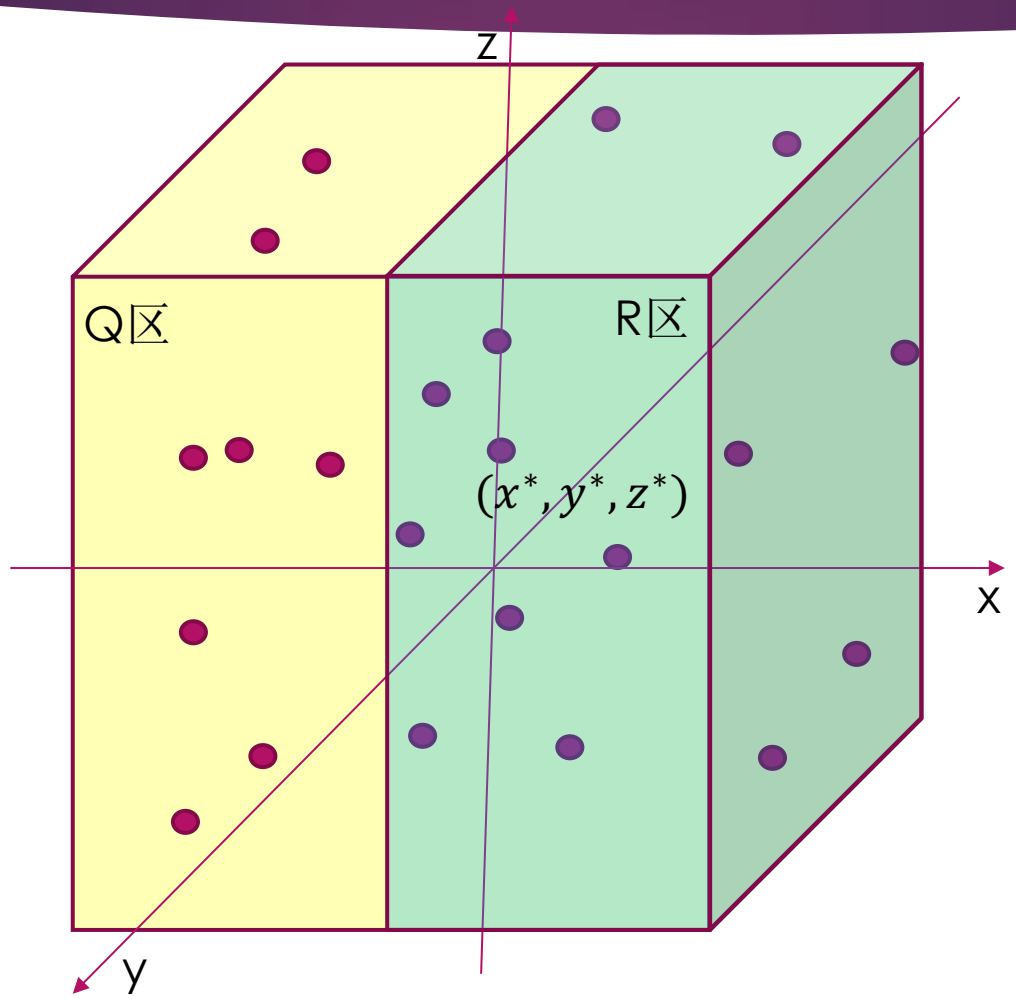
```
Closest-Pair(P){
    Px = 对P按照x值升序排序
    Py = 对P按照y值升序排序
    pair(p0,p1) = Closest-Pair-Rec(Px, Py)
    return (pair(p0,p1))
}
```

```
Closest-Pair-Rec(Px, Py){
    If Px.size <=3
        如果是3个点，两两比较，返回最小的点对
        如果是2个点，直接返回该点对
    Endif

    将Px划分为2个子序列Qx，Rx, 与Qx和Rx中点集合对应的，Py分为Qy,Ry，均保持升序排序
    pair(q0,q1) = Closest-Pair-Rec(Qx,Qy)
    pair(r0,r1) = Closest-Pair-Rec(Rx,Ry)
    δ=min(d(q0,q1),d(r0,r1))
    x* = Qx中的点最大的x坐标
    从Py构造列表Sy，包涵了x*左右δ范围内所有的点，并按照y值升序排序
    d_min= δ
    pair_min = null
    Foreach(s 属于 Sy且s_x<=x*)
        s_y在Sy的位置前后找s'，s'_x>x*且 | s'_y-s_y | <= δ
        If(d_min >d(s,s'))
            d_min = d(s,s')
            pair_min=(s,s')
        Endif
    Endforeach

    If (pair_min ! = null) return pair_min
    Else if d(q0,q1) < d(r0,r1) return pair(q0,q1)
    Else return pair(r0,r1)
    Endif
}
```

三维最邻近点对

请自己思考算法并提交。5月15号讲解题思路。
5月15号10点前可以全部AC的同学加bonus分20分。

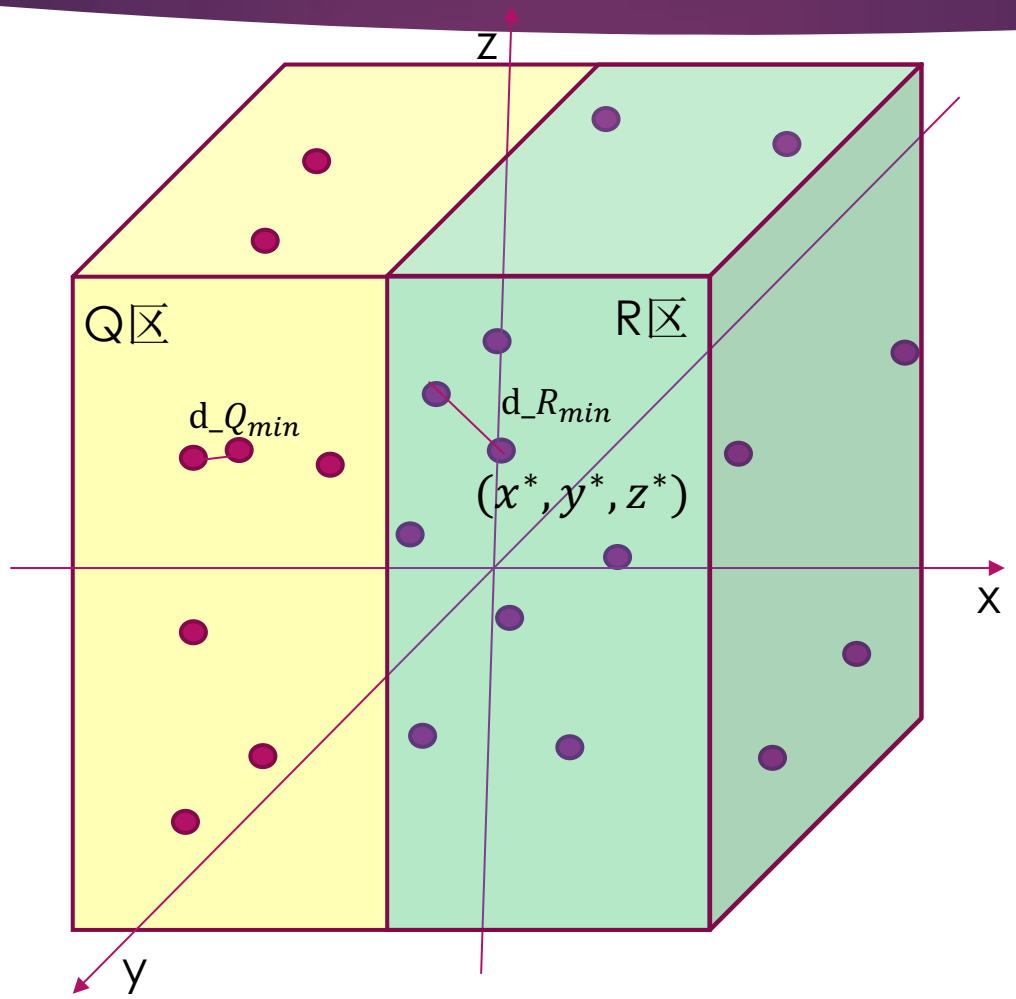
三维最邻近点对



每个点可表示为(x, y, z)

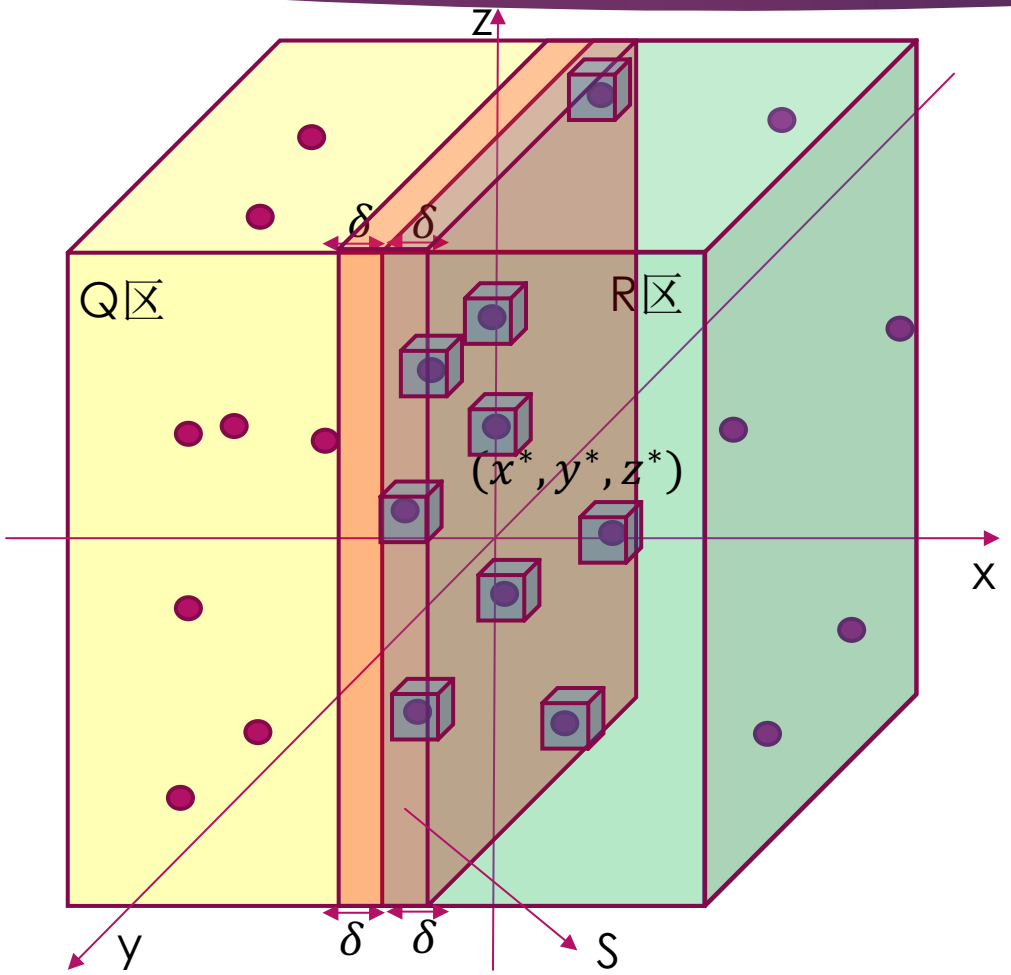
刚开始根据x的值取中间值用$x = x^*$这个面将点集分成2部分

三维最邻近点对



$$\delta = \min(d\_Q_{min}, d\_R_{min})$$

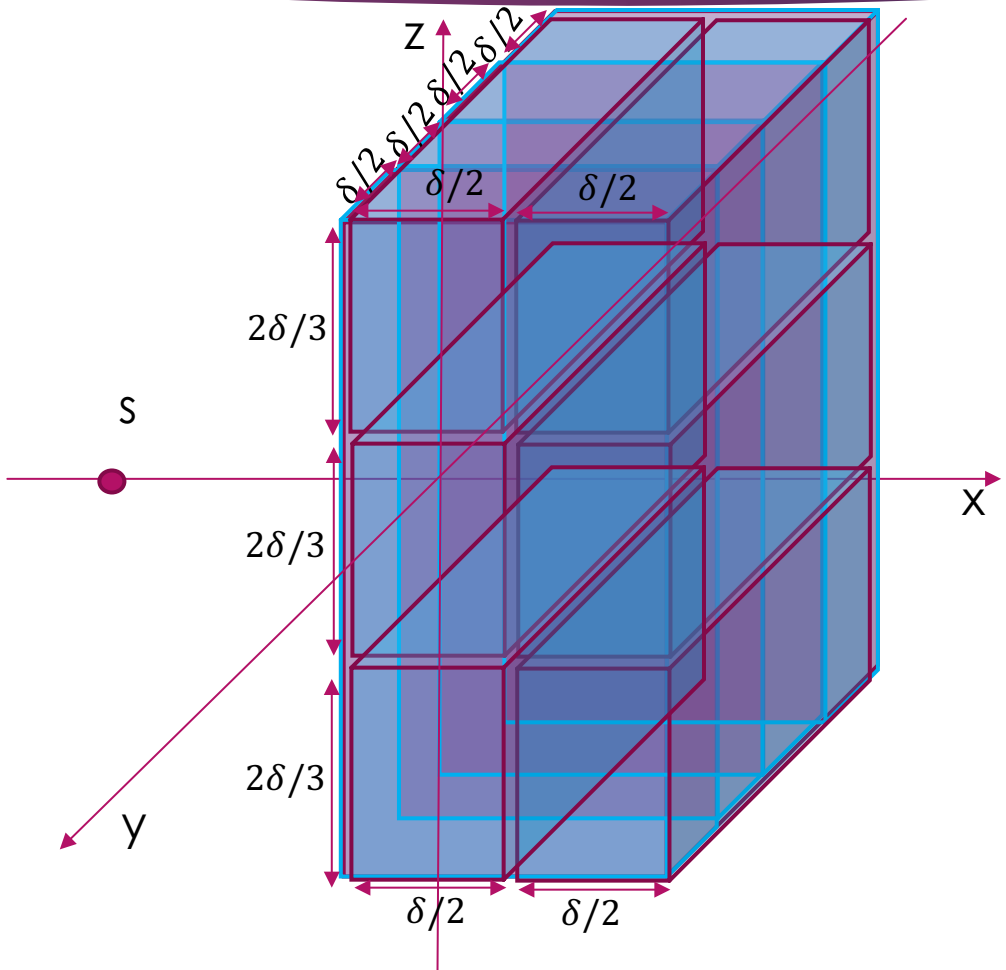
三维最邻近点对



对于任意一个集合S中的点s，求出这个点到附近其他点的距离的最小值

怎么定义点s附近的其他点？

三维最邻近点对

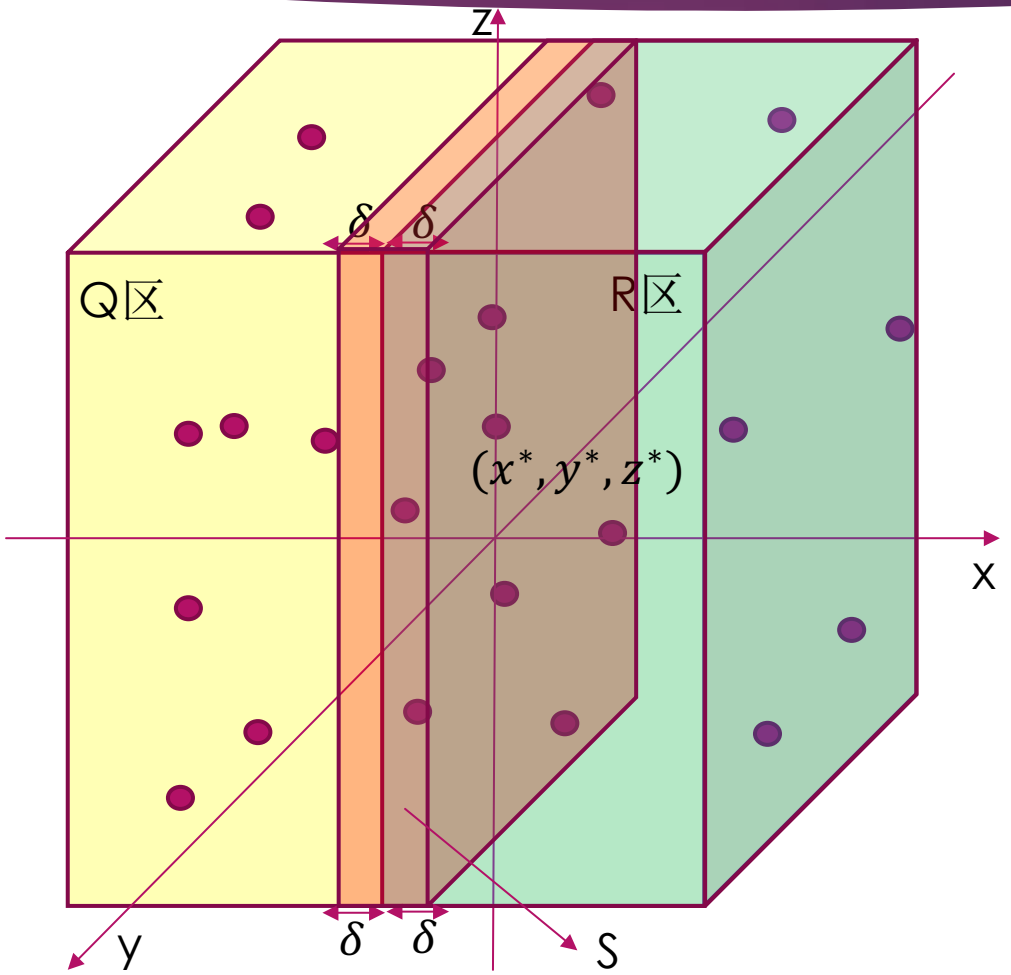


与点s相邻的点必定要在面L的右边区域找，而找寻的范围，在点s的坐标y的上下d区间同时坐标z的上下δ区间才可能存在比δ距离更临近的点。将右边的矩形分解成24个盒子，可以证明每个盒子中最多存在1个点。

如果s在面L的右边，同理。

如果s在面L上，同在左边的情况。

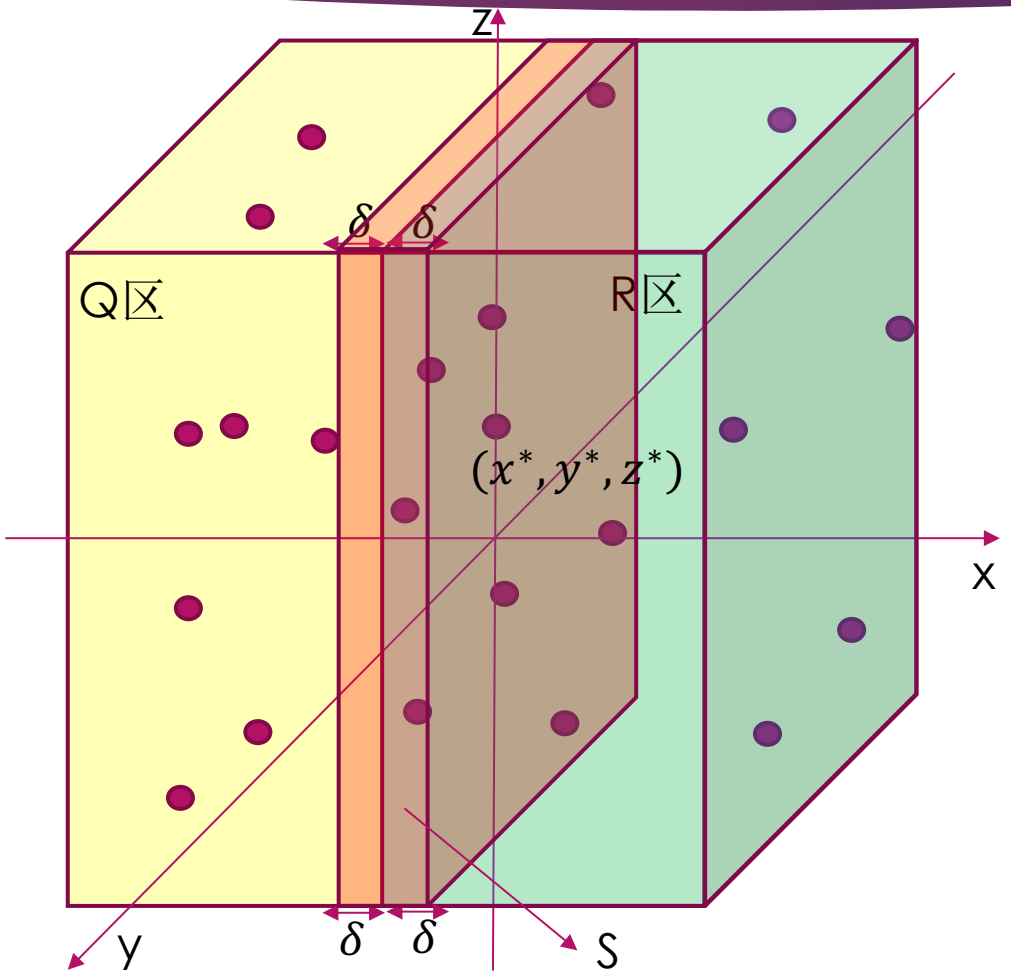
三维最邻近点对



虽然x的范围缩小了

y和z这种2个维度怎么处理呢？
按照y升序，y相等按照z升序？

三维最邻近点对

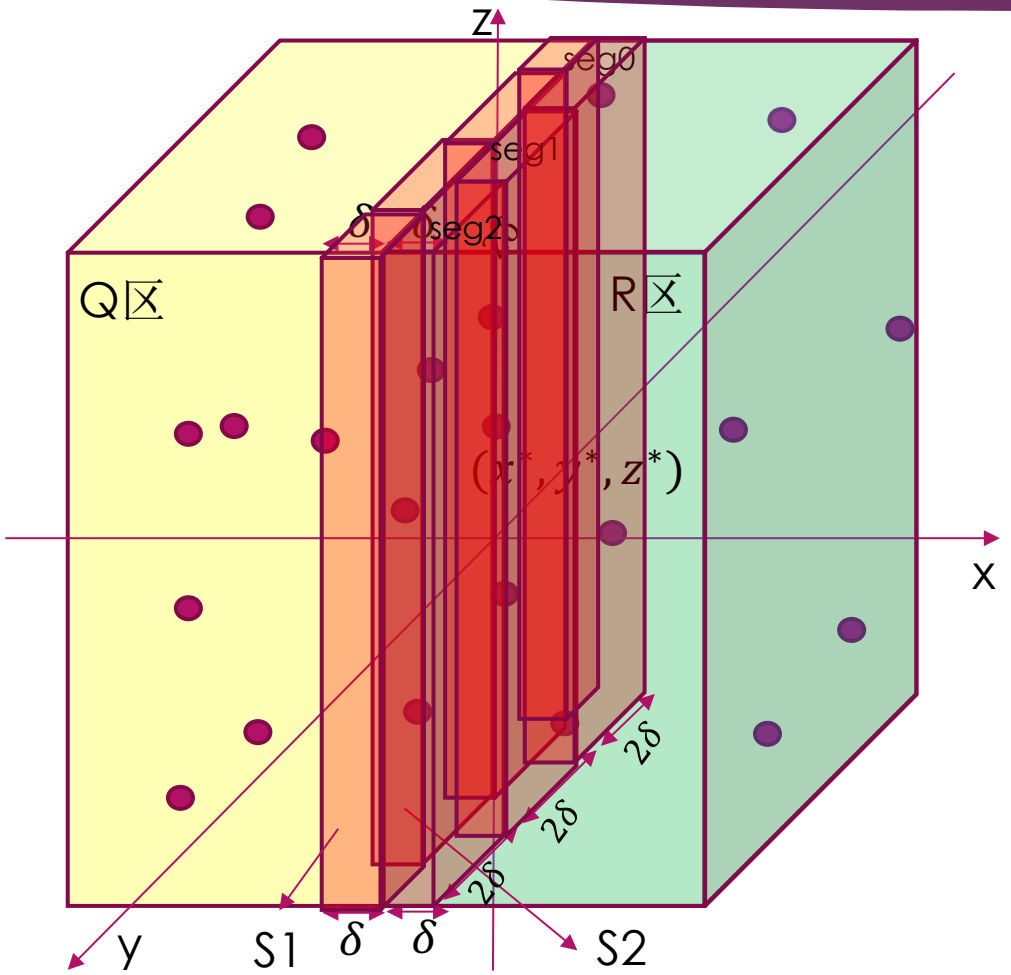


虽然x的范围缩小了

y和z这种2个维度怎么处理呢？
按照y升序，y相等按照z升序？

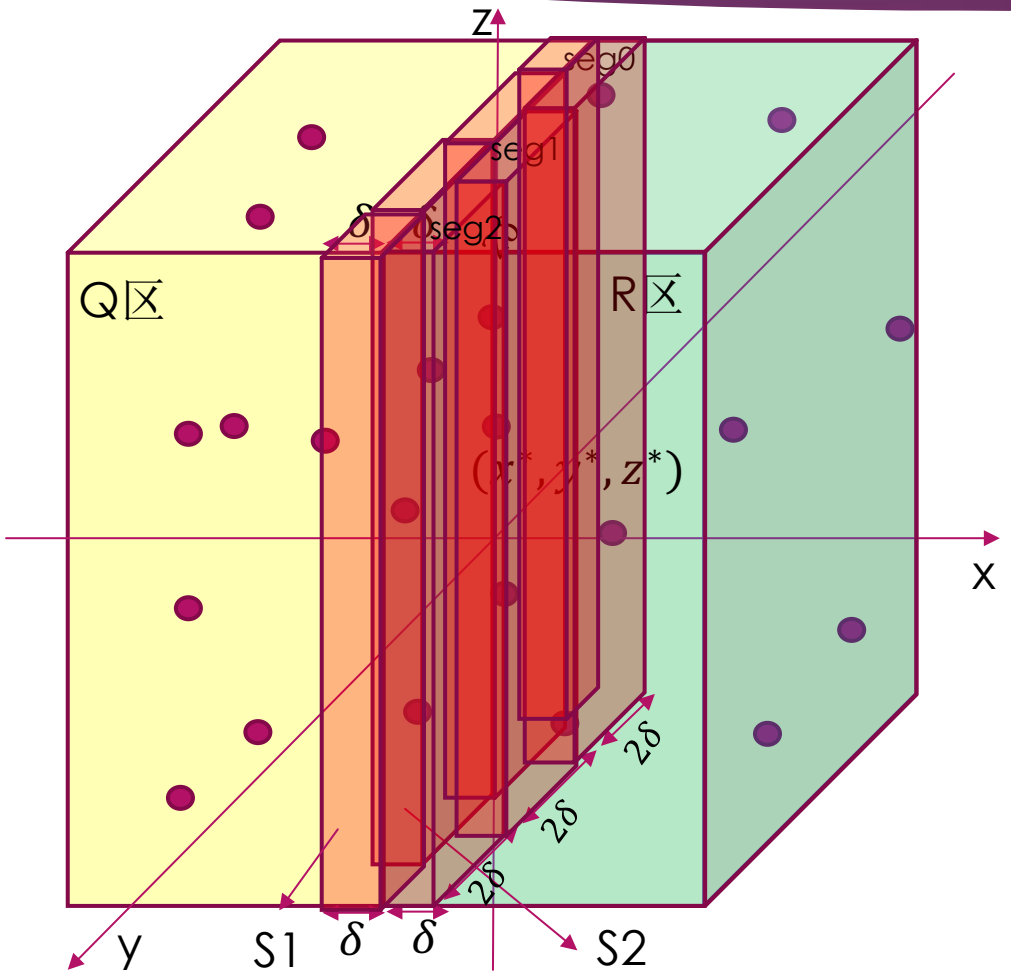
三维算法中最难的一点是，怎么根据点p快速找出其右边长方体中相邻的点。
在二维的时候，x虽然不确定，但是有个范围，所以在x范围确定的情况下，可以对y排序，当点确定时，其在y中的位置是确定的，其相邻的点即为其在二维空间中相邻的点。
在三维点时候，只确定了一个x的范围，而y和z都是不确定的，理论上有相当大的一个范围，此时只能再增加一个维度的切割，才能缩小查找范围。

三维最邻近点对



S集合按照面L可以分为S1区和S2区，分别可以按照y来切割。S2从点集中y的最小值开始按照2δ切割，对于（x-x*）<=δ中的点（属于R区），按照y切割成如下区段： (k)*2 δ<=|y-y_min|<(k+1)*2 δ(k = 0,1,2,3,....)
各区段中的点按照z值升序排列。
对于S1区的点，按照y值也可以划分区段，每个区段的点对应到S2区到相邻2个区段（首段和末位段除外）。例如S1的y坐标<y_min+d 对应到S2区seg0附近的点， S1区的y坐标>=y_min+d且< y_min+3d对应到S2区段0和段1中的点， S1区的y坐标>=y_min+3d且< y_min+5d对应到S2区段1和段2中的点。为了进一步缩小范围，使得在每个区段内查找到点限制在24内， S1区的点要分别纪录其z值的大小位于S2区对应区段的位置。
S1区的分段：
(2k-1)* δ<=|y-y_min|<(2k+1)*δ（k=1,2,3）
|y-y_min|< δ（k=0）
|y-y_min|<y_max − y_min (k为末尾段)
各区段的点按照z值升序排序

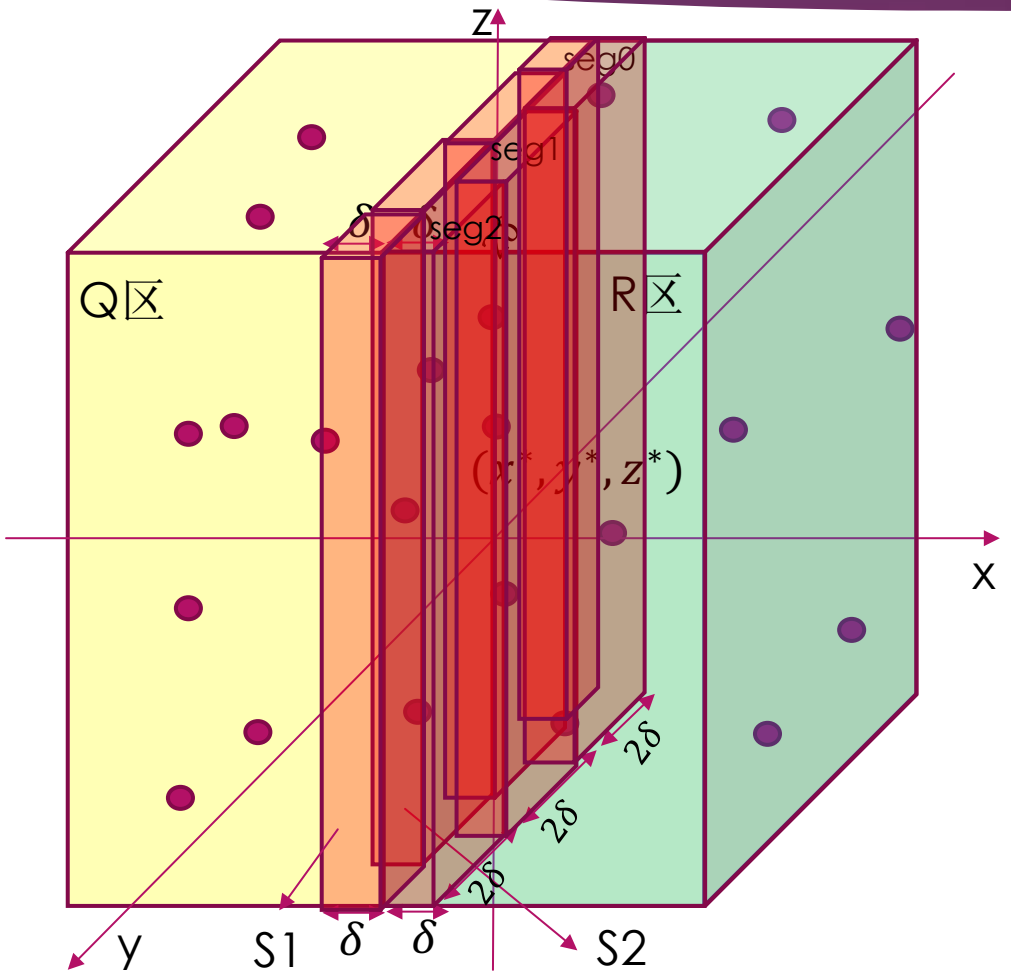
三维最邻近点对



S1区和S2区怎样在递归中以O(n)的时间构建？
初始时对P按照z值升序排序，形成数组Pz
合并时顺次扫描Pz
构造S1和S2的算法流程：

```
Foreach (p 属于 Pz){
    If (p的x坐标与x*相距在δ内且x<=x*){
      根据y值计算点p应该查找的S2的分段
      获取S2中对应分段的子列表的长度d1(和d2)
      将点p与长度d1和d2加入S1数组
    }

    if(p的x坐标与x*相距在δ内且x>x*){
      根据点p计算应该放入S2的区段
      S2该区段对应的子列表1(和2)添加该点P
    }
}
```

三维最邻近点对



举例说明：
以y_min为基准，2δ为步长，初始化S2列表的子列表个数，子列表初始化为空
S2的分段原则
[y_min ~y_min+2d) --- 段0
[y_min+2d ~ y_min+4d] ---段1
[y_min+4d ~ y_min+6d] ---段2
依次类推
如果S1的y坐标<y_min+d 找段0附近的点，需要在遍历的同时记录位置
如果S1的y坐标>=y_min+d且< y_min+3d,找段0和段1中的点，需要记录其z值分别在段0和段1中的位置
如果S1的y坐标>=y_min+3d且< y_min+5d,找段1和段2中的点，需要记录其z值分别在段1和段2中的位置
依次类推

三维最邻近点对算法流程

输入：三维点序列P
输出：P中距离最小的一对点

```
Closest-Pair(P){
    Px = 对P按照x值升序排序
    Pz = 对P按照z值升序排序
    pair(p0,p1) = Closest-Pair-Rec(Px, Pz)
    return (pair(p0,p1))
}
```

```
Closest-Pair-Rec(Px, Pz){
    If Px.size <=3
        如果是3个点，两两比较，返回最小的点对
        如果是2个点，直接返回该点对
    Endif

    将Px划分为2个子序列Qx，Rx, 与Qx和Rx中点集合对应的，Pz分为Qz,Rz，均保持升序排序
    pair(q0,q1) = Closest-Pair-Rec(Qx,Qz)
    pair(r0,r1) = Closest-Pair-Rec(Rx,Rz)
    δ=min(d(q0,q1),d(r0,r1))
    x* = Qx中的点最大的x坐标
    constructS1andS2(Px, Pz,  x*,δ,S1,S2) //见上上页PPT
    d_min = δ
    pair_min = null
    Foreach(s 属于S1)
        根据s纪录的S2的区段信息，到S2相应的位置前后(z差值不超过δ)找s'
        If(d_min >d(s,s'))
            d_min = d(s,s')
            pair_min=(s,s')
        Endif
    Endforeach
    If (pair_min != null) return pair_min
    Else if d(q0,q1) < d(r0,r1) return pair(q0,q1)
    Else return pair(r0,r1)

}
```

Lab8代码问题分析

```
int main()
{
    int length, temp;
    scanf("%d", &length);
    int origin[length];
    int clone[length];
    long long r = 0;
    for(int i = 0; i< length; i++){
        scanf("%d", &temp);
        origin[i] = temp;
        clone[i] = temp;
    }
    r = divide(origin, clone, 0, length-1);
    r = r % 1000000007;
    printf("%d",r);
}
```

运行最后一个样例会爆栈（ c++ ）

```
int divide(int origin[], int clone[], int low, int high)
{
    long long r = 0;
    long long rb = 0;
    long long ra = 0;
    if(high == low)
    {
        return 0;
    }
    int mid = low + (high - low) / 2;
    ra = divide(origin, clone, low, mid) % 1000000007;
    rb = divide(origin, clone, mid+1, high) % 1000000007;
    if (clone[mid] > clone[mid + 1])
    {
        r = mergeCount(clone, origin, low, mid, high) % 1000000007;
    }
    return (ra + rb + r) % 1000000007;
}
```



```
void divide(int low, int high)
{
    if(high == low)
    {
        return;
    }
    int mid = low + (high - low) / 2;
    divide(low, mid);
    divide(mid+1, high);
    if (clone1[mid] >= clone1[mid + 1])
    {
        mergeCount(low, mid, high);
    }
    else
    {
        for(int i = low; i <= high; i++)
        {
            origin[i] = clone1[i];
        }
    }
}
```