3.4 Medians and Order Statistics

- Decrease and Conquer原理
- Selection Problem

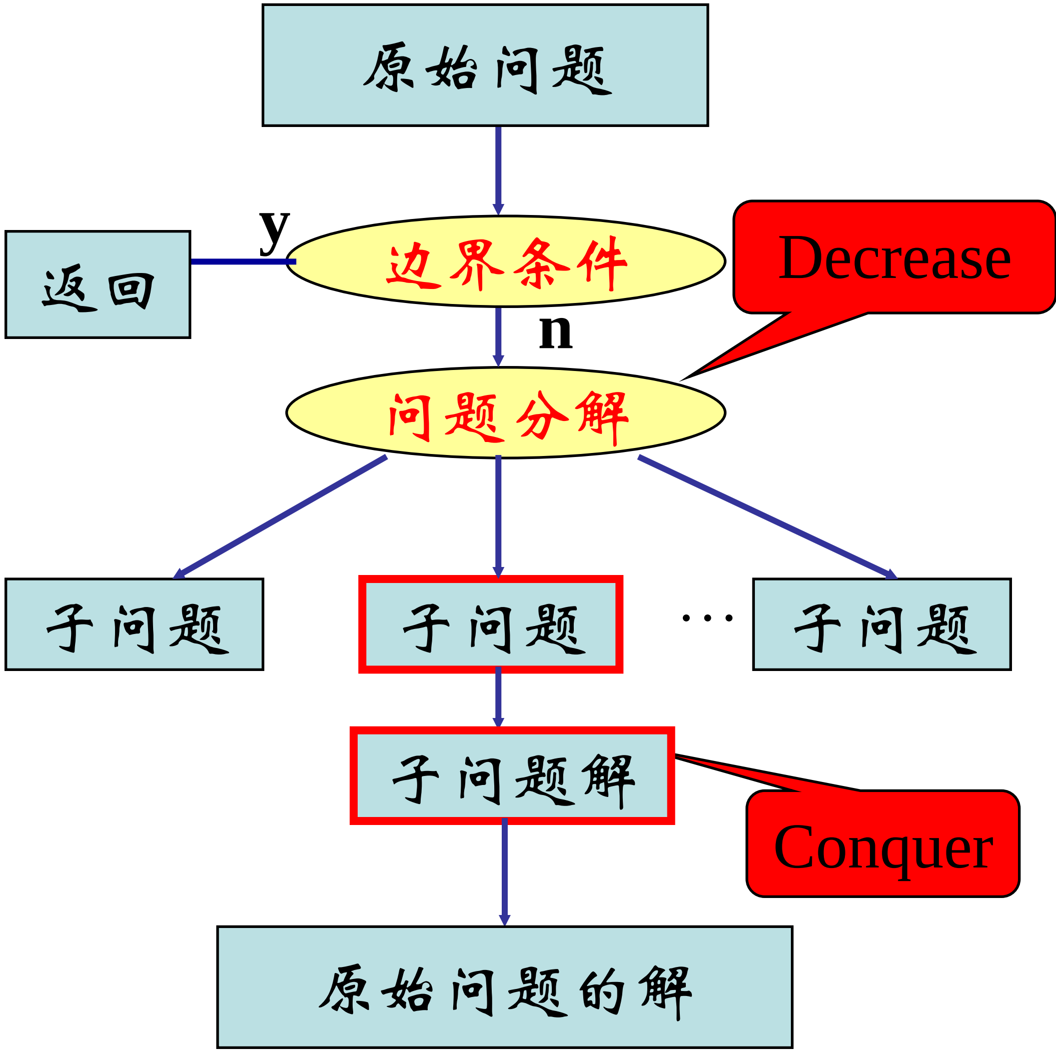
Decrease and Conquer原理

- 原始问题划分为若干子问题，将原始计算问题转化为其中某一个子问题的计算问题

例如：折半查找

$$T(n)=T(\lfloor n/2 \rfloor)+1$$

- 非常有效的一种方法，通常用于解决优化问题



Decrease and Conquer原理

- 与Divide and Conquer的不同
  - 分治方法：递归求解每一个子问题，然后通过合并各个子问题的解最后得到原始问题的解
  - 减治方法：仅通过求解某一个子问题的解得到原始问题的解

Medians and Order Statistics

- **The i^{th} order statistic problem**
  - Input: set S of n (distinct) elements, and a number k.
  - Output: element x in S that is greater than exactly k-1 elements in S.
  - Special order statistics
    - The 1^{st} order statistic is the *minimum* in S
    - The n^{th} order statistic is the *maximum* in S
    - The *median* in S is at
      - $(n+1)/2$ when n is odd
      - $n/2$ and $n/2+1$ when n is even

Selection Problem

- **Problem**
  - Input: set A of n (distinct) elements, and a number k.
  - Output: element x in A that is greater than exactly $k-1$ elements in A, i.e the k^{th} smallest element.

The straightforward algorithm:

step 1: Sort the n elements

step 2: Locate the k^{th} element in the sorted list.

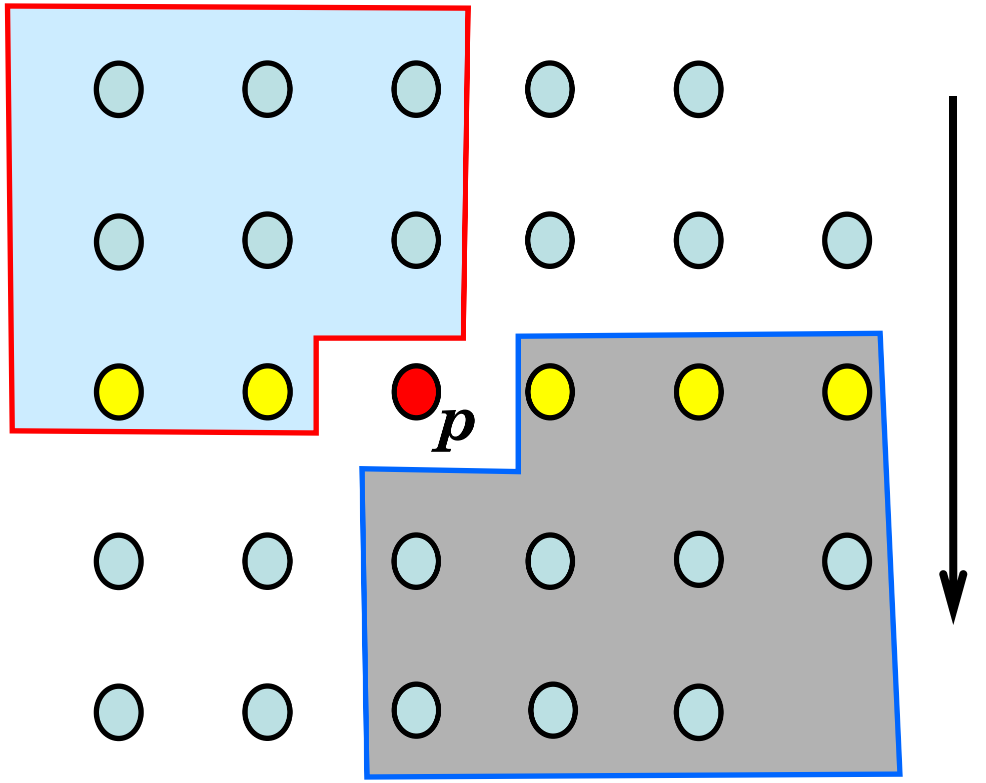
Time complexity: O($n$log$n$)

Algorithm of Selection Problem

- **Main Idea**
  - $S=\{a_1, a_2, \ldots, a_n\}$
  - Let $p \in S$, 用p 将S 划分为3 个子集合S_1, S_2, S_3:
    - $S_1=\{ a_i \mid a_i < p , 1 \le i \le n\}$
    - $S_2=\{ a_i \mid a_i = p , 1 \le i \le n\}$
    - $S_3=\{ a_i \mid a_i > p , 1 \le i \le n\}$
  - 3 种情况:
    - 若$|S_1| > k$, 则在集合S_1中搜索第k小的元素.
    - 否则, 若$|S_1| + |S_2| > k$, 则 p 是S中第k小的元素.
    - 否则, 在S_3中搜索第$(k - |S_1| - |S_2|)$小的元素.

Algorithm of Selection Problem

- 如何选择p?
  - 把n个元素划分成$\left\lceil \frac{n}{5} \right\rceil$个组（每组有5个元素.）
  - 排序每一个组
  - 找到元素p: 它是$\lceil n/5 \rceil$个组的中位数的中位数



Algorithm of Selection Problem

算法步骤:

$O(n)$

Step 1: 划分 S 为$\lceil n/5\rceil$个组. 每组包含5个元素. 若最后一组不足5个元素，则用∞补足.

Step 2: 排序每组5个元素， 并确定每一分组的中位数. **$O(n)$**

Step 3: 计算$\lceil n/5\rceil$个中位数的中位数p. **$T(n/5)$**

Step 4: p将 S 划分为三个子集合 S_1, S_2 及 S_3, S_1中元素均小于p， S_2中元素均等于p， S_3中元素均大于p.

$O(n)$

Step 5: 若$|S_1|\geq k$, 则递归地在S_1中搜索第 k小元素;

否则， 若$|S_1| + |S_2|\geq k$， 则 p 即为S中第 k小元素;

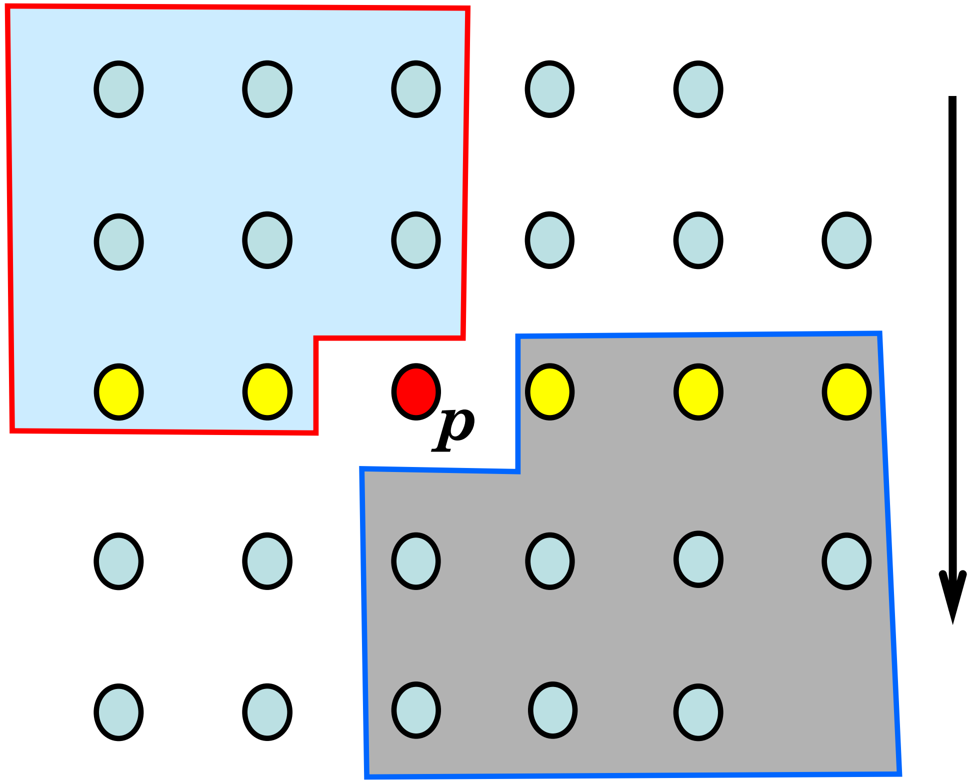
否则， 令 $k' = k - |S_1| - |S_2|$， 递归地在S_3中搜索第 k'小元素.

?

Performance Analysis (粗略)

Each 5-element subset is sorted in non-decreasing sequence.

“大约”至少 $|S|/4$ 个元素小于等于 p



“大约”至少 $|S|/4$ 个元素大于等于 p

Algorithm of Selection Problem

算法步骤:

$O(n)$

Step 1: 划分 S 为 $\lceil n/5 \rceil$ 个组. 每组包含5个元素. 若最后一组不足5个元素，则用∞补足.

Step 2: 排序每组5个元素，并确定每一分组的中位数. $O(n)$

Step 3: 计算 $\lceil n/5 \rceil$ 个中位数的中位数 p. $T(n/5)$

Step 4: p 将 S 划分为三个子集合 S_1, S_2 及 S_3, S_1 中元素均小于 p，S_2 中元素均等于 p，S_3 中元素均大于 p.

$O(n)$

Step 5: 若 $|S_1| \geq k$, 则递归地在 S_1 中搜索第 k 小元素;

否则，若 $|S_1| + |S_2| \geq k$，则 p 即为 S 中第 k 小元素;

否则，令 $k' = k - |S_1| - |S_2|$，递归地在 S_3 中搜索第 k' 小元素.

$T(3n/4)$

Performance Analysis (粗略)

- 算法复杂性分析

$T(n)=T(3n/4)+T(n/5) + O(n)$

Let $T(n) = a_0 + a_1 n + a_2 n^2 + \ldots \ , a_1 \neq 0$

$T(3n/4) = a_0 + (3/4)a_1 n + (9/16)a_2 n^2 + \ldots$

$T(n/5) = a_0 + (1/5)a_1 n + (1/25)a_2 n^2 + \ldots$

$T(3n/4 + n/5) = T(19n/20) = a_0 + (19/20)a_1 n + (361/400)a_2 n^2 + \ldots$

$T(3n/4) + T(n/5) = a_0 + a_0 +(19/20)a_1 n + (241/400)a_2 n^2 + \ldots$

$\leq a_0 + T(19n/20)$

$\Rightarrow T(n) \leq cn + T(19n/20)$

Performance Analysis (粗略)

$$\Rightarrow T(n) \le cn + T(19n/20)$$

$$\le cn + (19/20)cn + \mathrm{T}((19/20)^2 n)$$

$$\vdots$$

$$\le cn + (19/20)cn + (19/20)^2 cn + \ldots + (19/20)^p cn + T((19/20)^{p+1} n)\ ,$$

$$(19/20)^{p+1} n \le 1 \le (19/20)^p n$$

$$= cn\ (1 + 19/20 + (19/20)^2 + \ldots + (19/20)^p) + b$$

$$= \frac{1-(\frac{19}{20})^{p+1}}{1-\frac{19}{20}} cn + b$$

$$\le 20\ cn + b$$

$$= O(n)$$

- **Time complexity analysis**
  - 比用于划分的元素x更大的元素数量至少是：

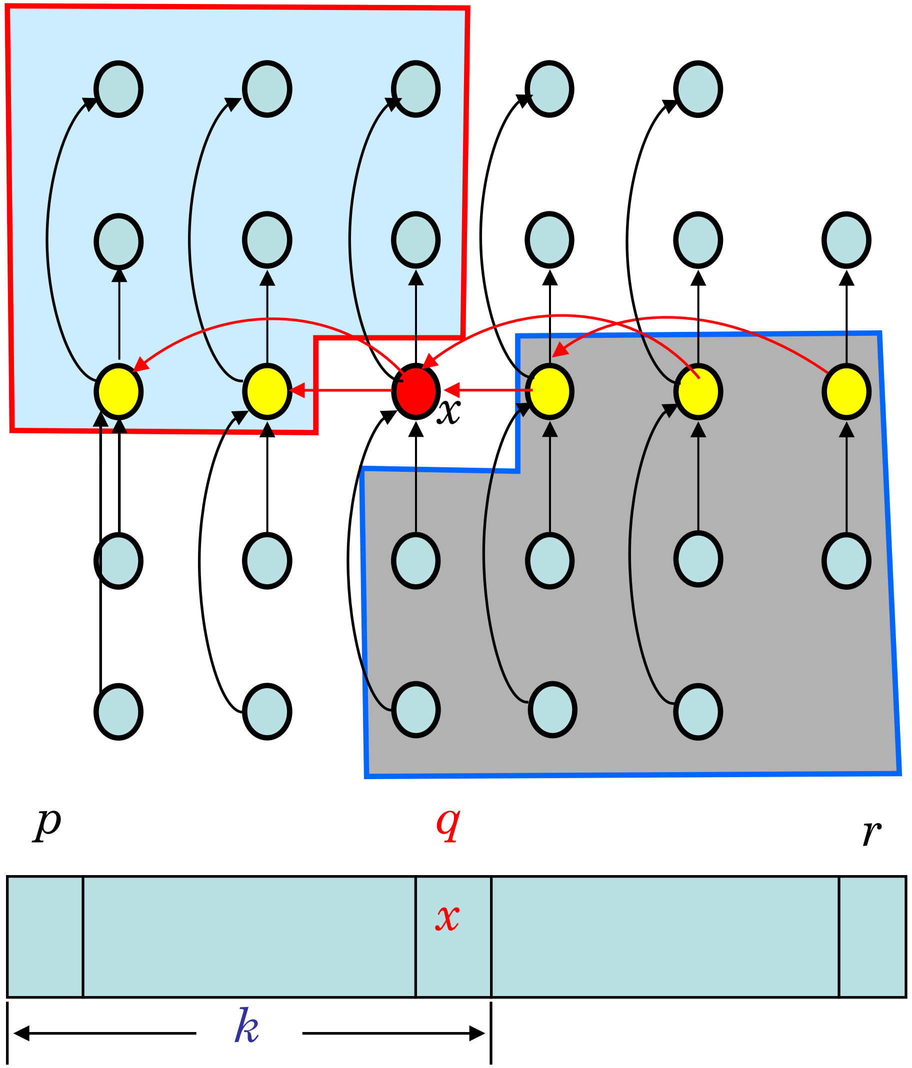
$$3\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{5}\right\rceil \right\rceil -2\right)\geq \frac{3n}{10}-6$$

  - 因此，最坏情况下，比x更小的元素的数量最多是：

n-((3*n*/10)-6) =7n/10+6.

  - 同理，比x更大的元素的数量最多也是：

7n/10+6



1. Divide n elements in A into $\lceil n/5 \rceil$ groups of 5 elements each, at most one group has $(n \mod 5)$ elements.	$O(n)$
2. Find median of each group by sorting first.	$O(n)$
3. Use Select recursively to find the median x of the $\lceil n/5 \rceil$ medians. In case of having two medians, take the lower.	$T(\lceil n/5 \rceil)$
4. Exchange x with the last element in A and apply Partition subroutine. Let k be the number of elements on the low side of the partition including x.	$O(n)$
5. If $i = k$, return x. Otherwise, use Select recursively to find the i^{th} smallest element on the low side if $i \le k$, or the $(i-k)^{th}$ smallest element on the high side if $i > k$.	$T(7n/10+6)$

$$T(n) \le \begin{cases} \theta(1) & \text{if } n \le 140 \\ T(\lceil n/5 \rceil) + T(7n/10+6) + O(n) & \text{if } n > 140 \end{cases}$$

- **Now we have**

$$T(n) \le \begin{cases} \theta(1) & \text{if } n \le 140 \\ T(\lceil n/5 \rceil) + T(7n/10+6) + O(n) & \text{if } n > 140 \end{cases}$$

- **Using inductive method, we can prove $T(n) \le cn$ for some c and $n>140$.**
- **Thus, the worst case time complexity is $T(n)=O(n)$.**

3.5 Finding the closest pair of points

优化combine阶段，降低$T(n)=aT(n/b)+f(n)$中的$f(n)$

问题定义

输入：Euclidean空间上的n个点的集合Q

输出：$A, B \in Q$,

$Dis(A, B)=Min\{Dis(P_i, P_j) \mid P_i, P_j \in Q\}$

Dis(P_i, P_j)是Euclidean距离：

如果$P_i=(x_i, y_i)$, $P_j=(x_j, y_j)$，则

$$Dis(P_i,P_j)=\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}$$

一维空间算法

- 利用排序的算法
  - 算法
    - 把Q中的点排序

    - 通过有序集合的线性扫描找出最近点对
  - 时间复杂性
    - $T(n)=O(n\log n)$

一维空间算法(续)

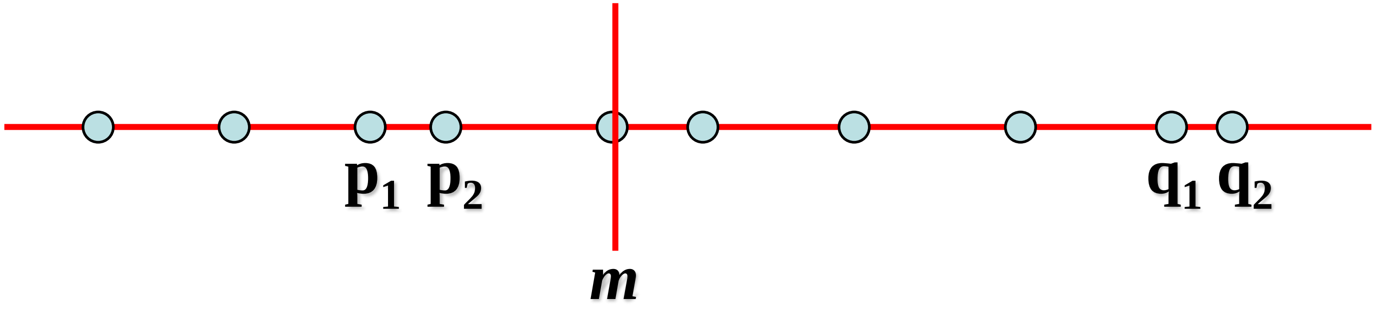
- Divide-and-conquer算法

边界条件:

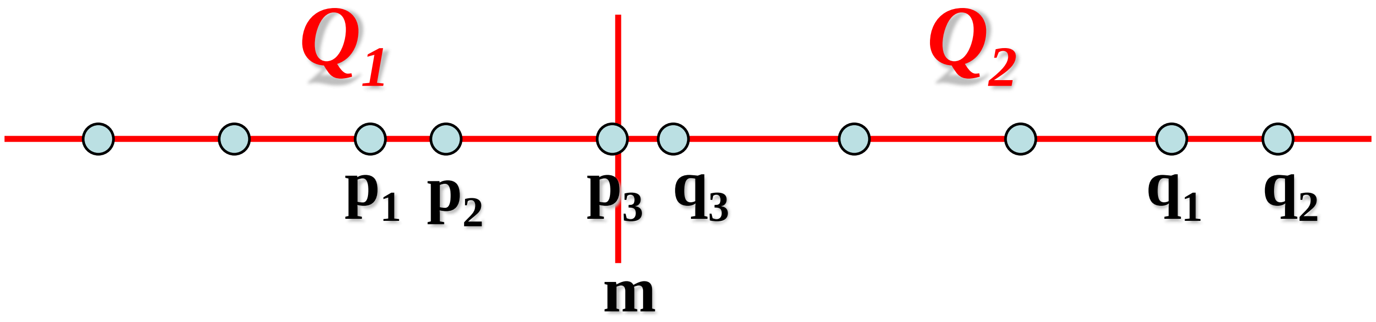
1. 如果Q中仅包含2个点，则返回这个点对;

Divide:

2. 求Q中点的中位数m;



3\. 用Q中点坐标中位数m把Q划分为两个大小相等的子集合$Q_1=\{x\in Q \mid x\leq m\}$，$Q_2=\{x\in Q \mid x>m\}$

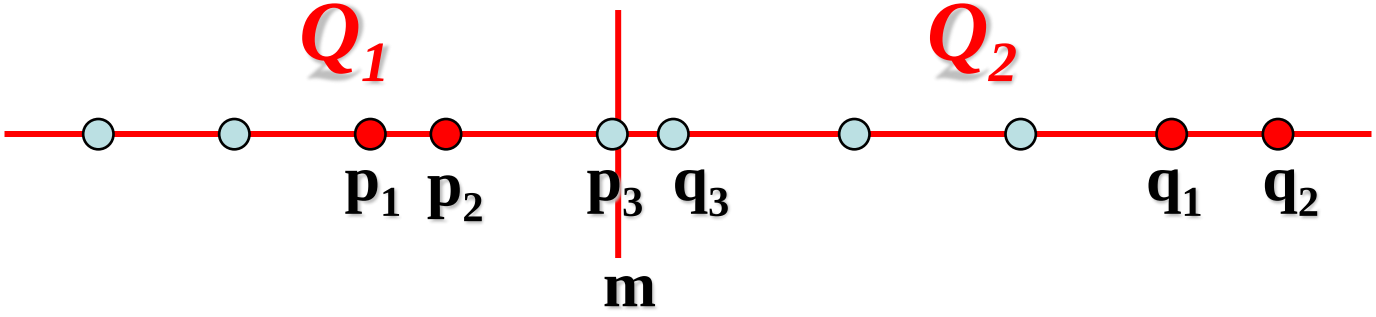


Conquer:

4\. 递归地在Q_1和Q_2中找出最接近点对

(p_1, p_2)和(q_1, q_2)

Merge:



5. 在(p_1, p_2)、(q_1, q_2)和某个(p_3, q_3)之间选择最接近点对(x, y)，其中p_3是Q_1中最大点，q_3是Q_2中最小点。

(x, y)是Q中最接近点对

- 时间复杂性
  - Divide阶段需要$O(n)$时间
  - Conquer阶段需要$2T(n/2)$时间
  - Merge阶段需要$O(n)$时间
  - 递归方程

$$T(n) = O(1) \qquad n = 2$$

$$T(n) = 2T(n/2) + O(n) \qquad n \geq 3$$

  - 用Master定理求解$T(n)$

$$T(n) = O(nlogn)$$

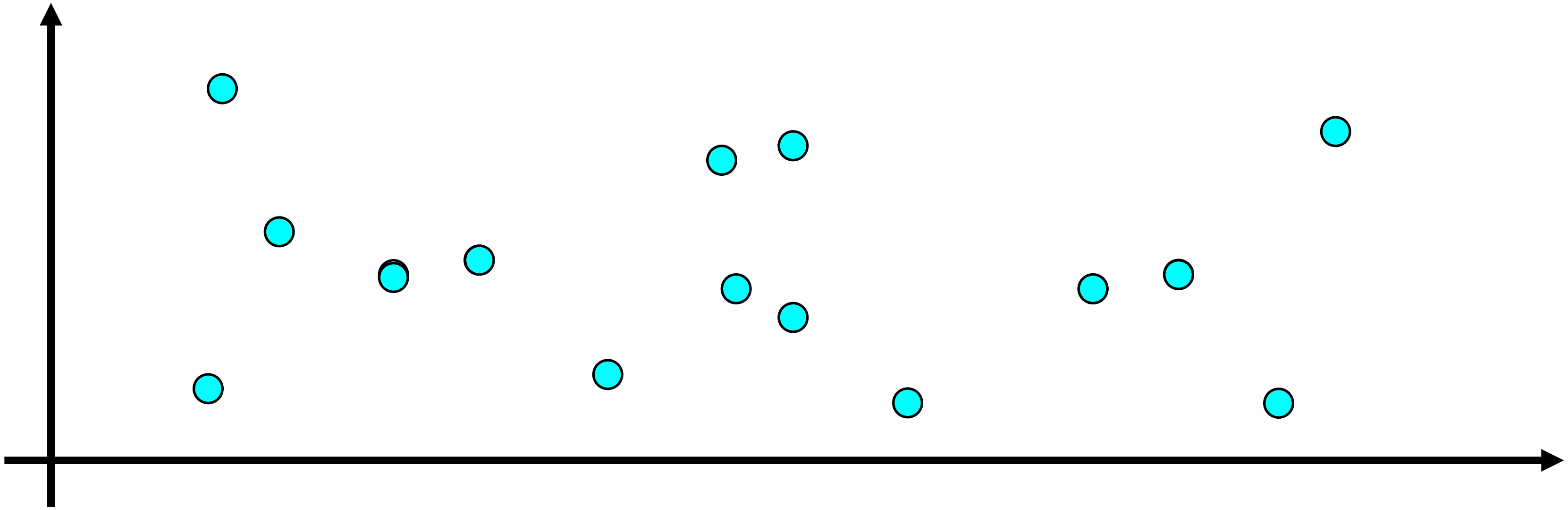
二维空间算法

- Divide-and-conquer算法

Assume: Q中点已经分别按x坐标和y坐标排序后存储在X和Y中.

边界条件:

1. 如果Q中仅包含3个点,则返回最近点对,结束;

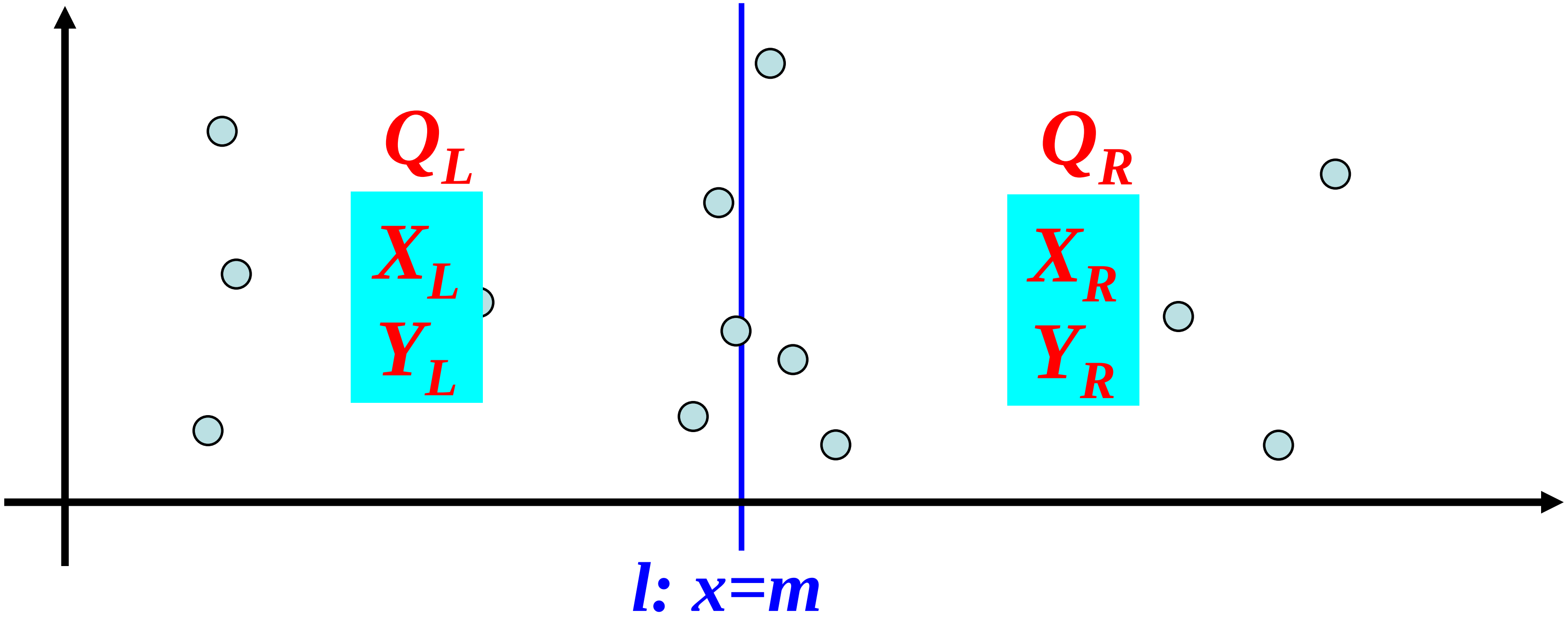


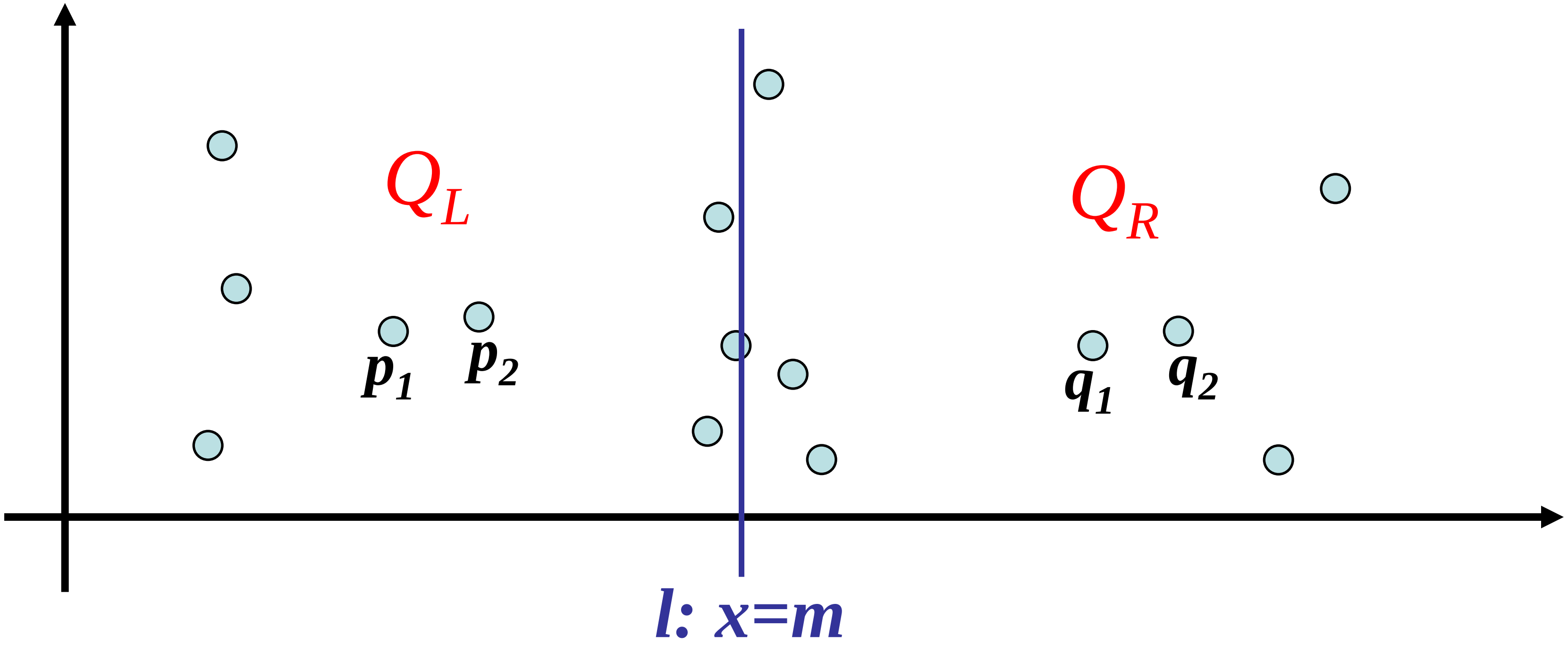
Divide:

2. 计算Q中各点x-坐标的中位数m；

3. 用垂线l: $x=m$把Q划分成两个大小相等的子集合Q_L 和 Q_R， Q_L中点在l左边, Q_R 中点在l右边;

4. 把X划分为X_L和X_R； 把Y划分为Y_L和Y_R ;



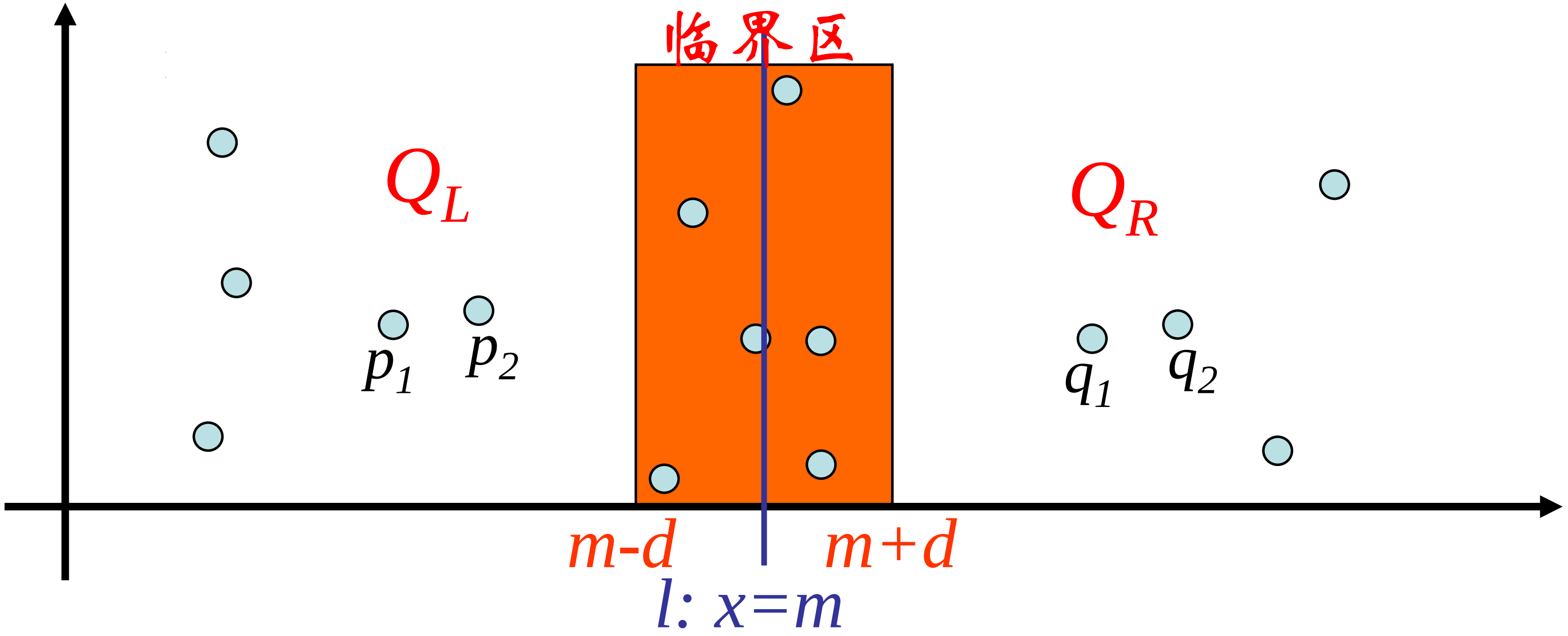


Conquer:

5. 递归地在Q_L、Q_R中找出最近点对:

$$(p_1, p_2)\in Q_L\,,\ (q_1, q_2)\in Q_R$$

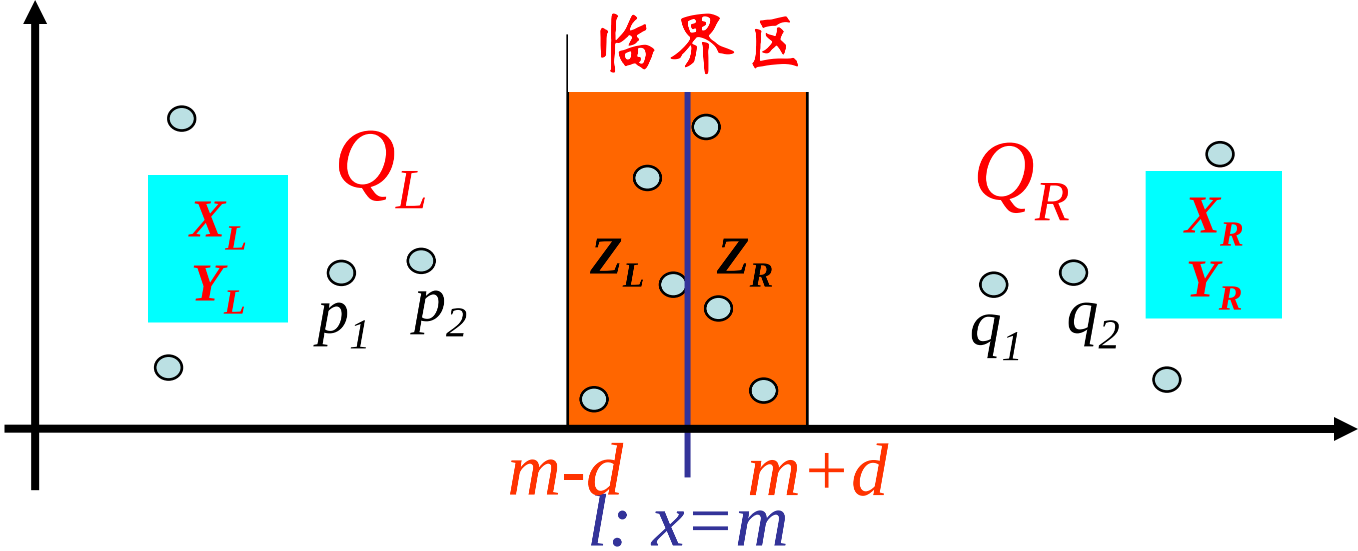
6. $d=\min\{Dis(p_1, p_2),\ Dis(q_1, q_2)\}$;



Merge:

1. 在临界区查找距离小于d的最近点对(p_l, q_r), $p_l \in Q_L$, $q_r \in Q_R$;
2. 若找到，则(p_l, q_r)是Q中最近点对，否则(p_1, p_2)和(q_1, q_2) 中距离最小者为Q中最近点对.

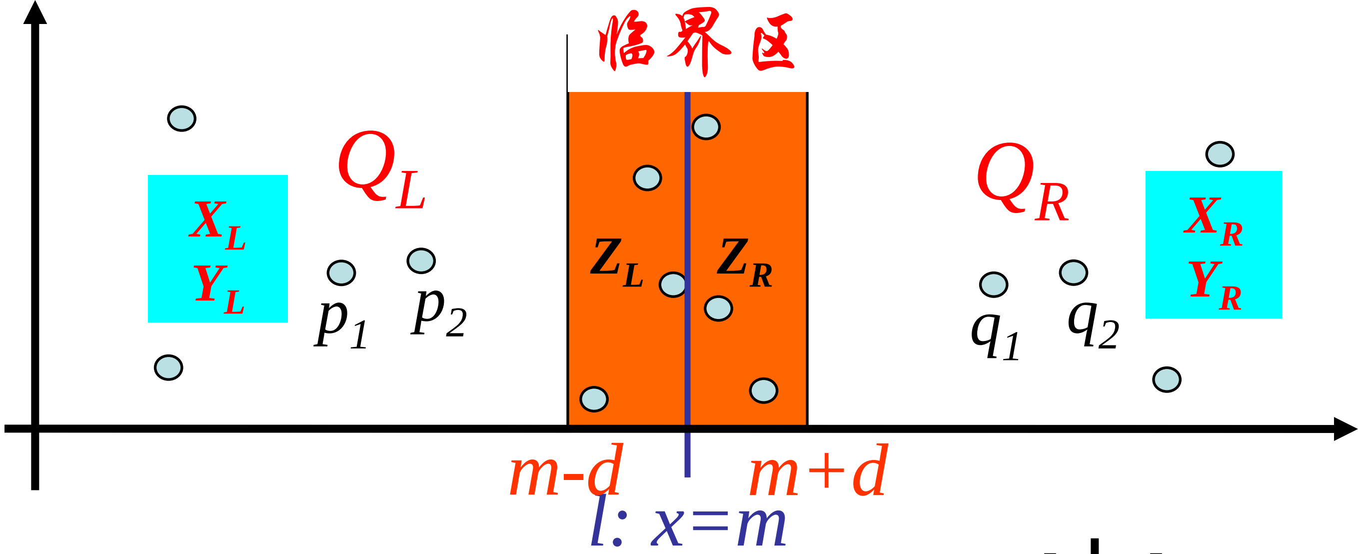
关键是(p_l, q_r)的搜索方法及其搜索时间



- $(p_l\,, q_r)$搜索算法
  1. $Z_L = \{Q_L$ 中左临界区点$\}$；
     $Z_R = \{Q_R$ 中右临界区点$\}$；

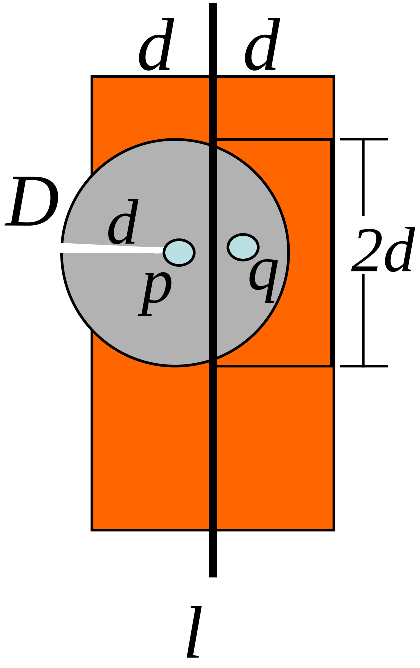
- 时间复杂性

  $O(6n)=O(n)$

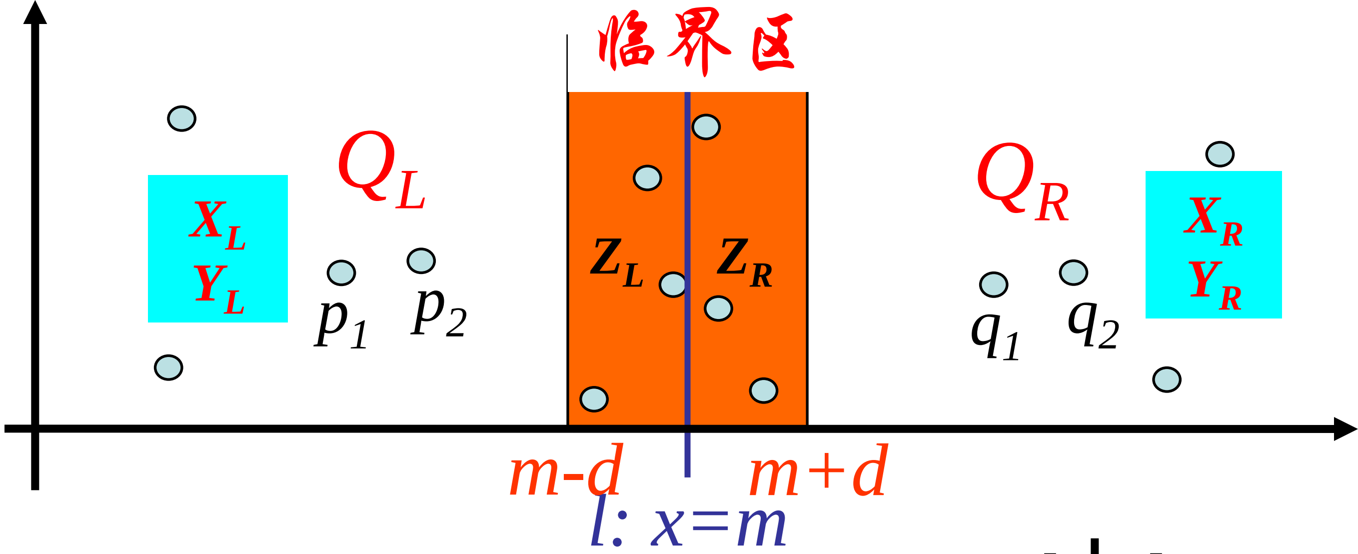


- (p_l , q_r)搜索算法

```
1. $Z_L = \{Q_L$ 中左临界区点$\}$;
   $Z_R = \{Q_R$ 中右临界区点$\}$;
2. For  $\forall p(x_p, y_p)\in Z_L$  Do
3.     For $\forall q(x_q, y_q)\in Z_R$ Do     $(y_p-d\le y_q\le y_p+d)$
                  \*这样点至多6个*\
4.         If  $Dis(p, q)<d$
5.         Then $d=Dis(p, q)$, 记录$(p, q)$;
6. 如果$d$发生过变化, 与最后的$d$对应的点对即为$(p_l , q_r)$,
   否则不存在$(p_l, q_r)$.
```

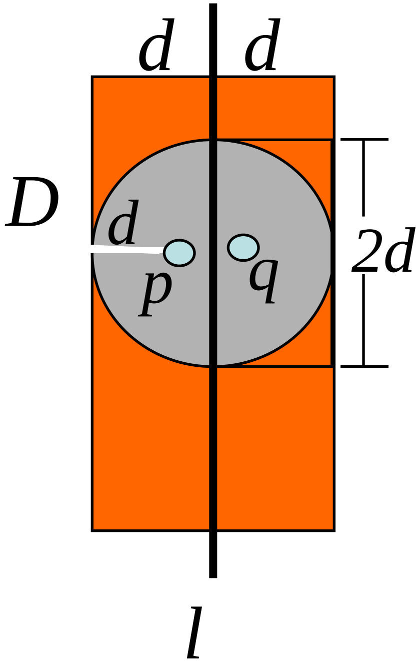


- 时间复杂性
  $O(6n)=O(n)$



- $(p_l\,, q_r)$搜索算法

```
1. $Z_L=\{Q_L$ 中左临界区点$\}$;
   $Z_R=\{Q_R$ 中右临界区点$\}$;
2. For $\forall p(x_p, y_p)\in Z_L$ Do
3.     For $\forall q(x_q, y_q)\in Z_R$ Do     $(y_p-d\le y_q\le y_p+d)$
                \*这样点至多6个*\
4.         If $Dis(p, q)<d$
5.         Then $d=Dis(p, q)$, 记录$(p, q)$;
6. 如果$d$发生过变化, 与最后的$d$对应的点对即为$(p_l\,, q_r)$,
   否则不存在$(p_l, q_r)$.
```



- 时间复杂性
  - Divide阶段需要$O(n)$时间
  - Conquer阶段需要$2T(n/2)$时间
  - Merge阶段需要$O(n)$时间
  - 递归方程

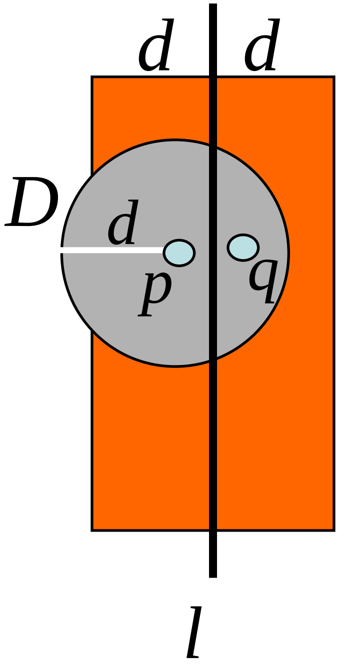
    $T(n)= O(1) \qquad n \leq 3$

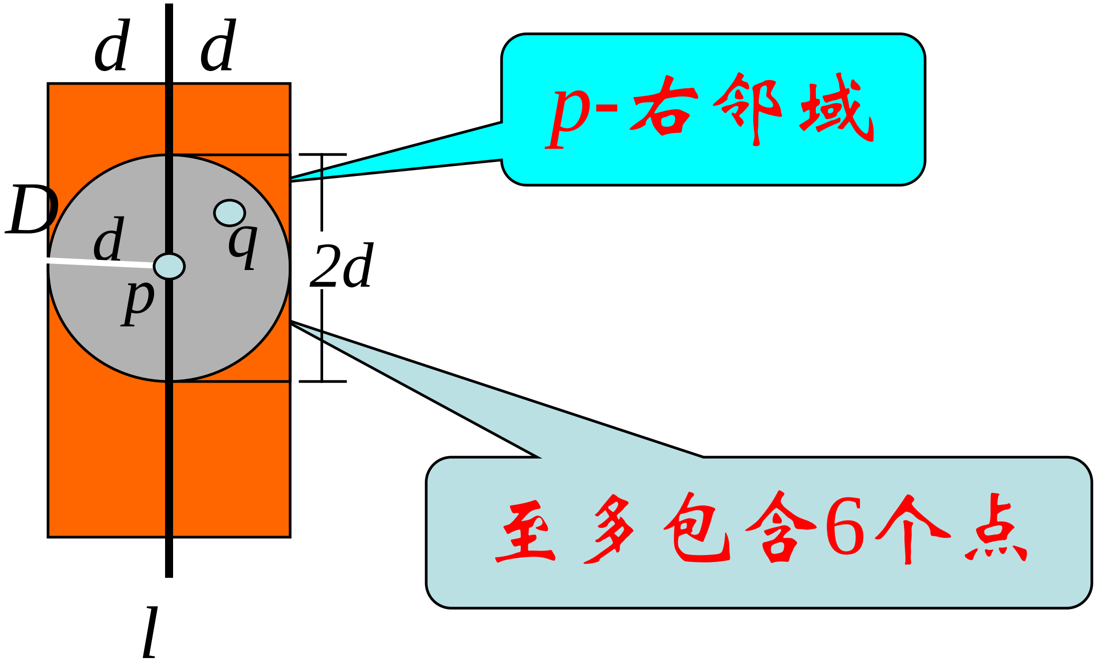
    $T(n) = 2T(n/2) + O(n) \qquad n > 3$
  - 用Master定理求解$T(n)$

    $T(n) = O(n\log n)$

(p_l, q_r)的搜索时间：

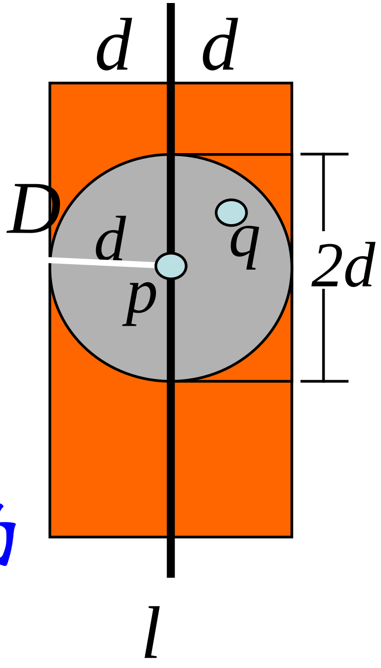
- 若(p, q)是最近点对而且$p \in Q_L$, $q \in Q_R$, $dis(p, q) < d$，(p, q)只能在下图的区域D.
- 若p在分割线l上，包含(p, q)的区域D最大，嵌于$d \times 2d$的矩形(p-右邻域)中，如下图所示.





p-右邻域

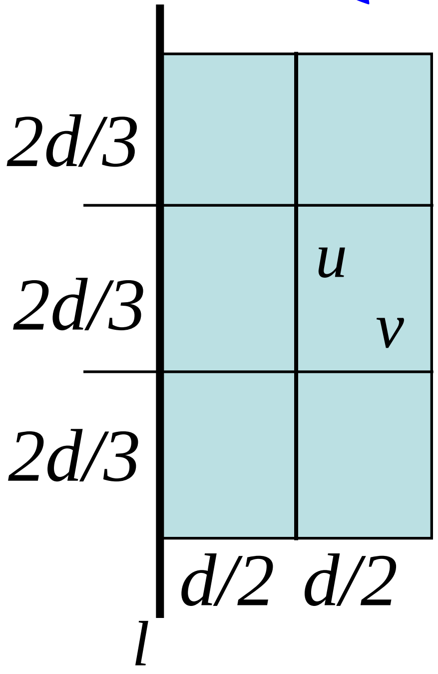
至多包含6个点



定理1. 对于左临界区中的每个点p，

p-右邻域中至多包含6个点。

证明：把p-右邻域划分为6个$(d/2)\times(2d/3)$的矩形。



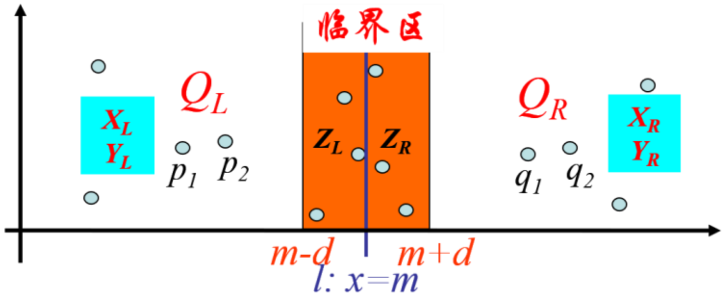
若p-右邻域中点数大于6，由鸽巢原理，至少有一个矩形中有两个点，设为u、v，则

$$(x_u-x_v)^2+(y_u-y_v)^2\le(d/2)^2+(2d/3)^2=25d^2/36$$

即$Dis(u, v)\le5d/6<d$，与d的定义矛盾。

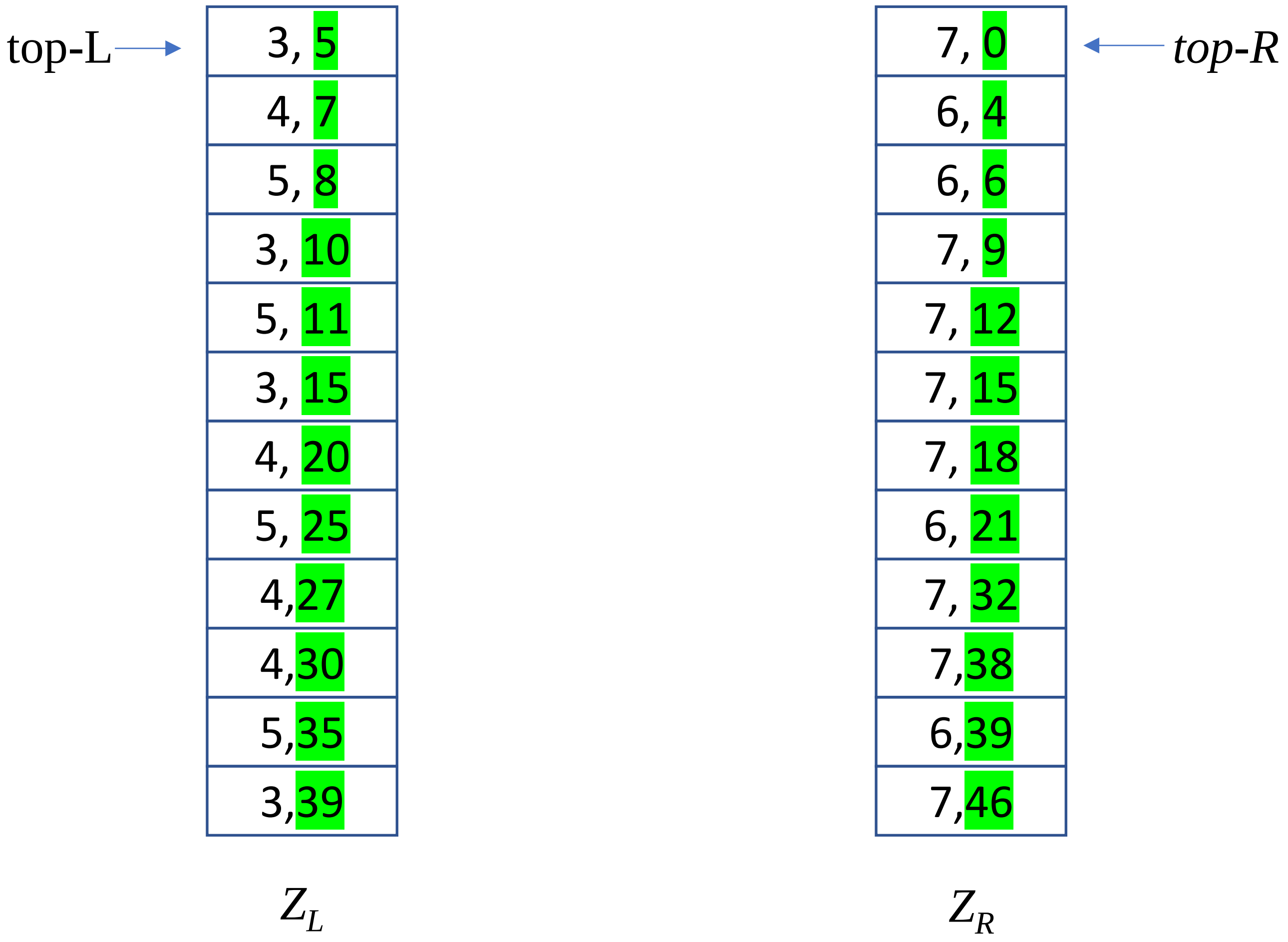
- (p_l, q_r)搜索算法

```
Input: $Y_L$, $Y_R$, $d$
Output: $result$
扫描$Y_L$得到$Q_L$中左临界区点，保持y坐标排序，得到$Z_L$
扫描$Y_R$得到$Q_R$中左临界区点，保持y坐标排序，得到$Z_R$
$result$=null;
$top\text{-}R=0$;
for $top\text{-}L=0$ to $Z_L.length-1$
    while $Z_L[top\text{-}L].y > Z_R[top\text{-}R].y + d$
        if $top\text{-}R=Z_R.length-1$
            return $result$;
        $top\text{-}R \leftarrow top\text{-}R+1$;
    if $Z_L[top\text{-}L].y \geq Z_R[top\text{-}R].y - d$ and $Z_L[top\text{-}L].y \leq Z_R[top\text{-}R].y + d$
        for $i=0$ to $min\{5, Z_R.length - top\text{-}R - 1\}$
            if $dist(Z_L[top\text{-}L], Z_R[top\text{-}R+i])<d$
                $result=(Z_L[top\text{-}L], Z_R[top\text{-}R+i])$;
                $d=dist(Z_L[top\text{-}L], Z_R[top\text{-}R+i])$;
return $result$;
```



m=5, *d*=4

Result=null



3, 5
4, 7
5, 8
3, 10
5, 11
3, 15
4, 20
5, 25
4,27
4,30
5,35
3,39

Z_L

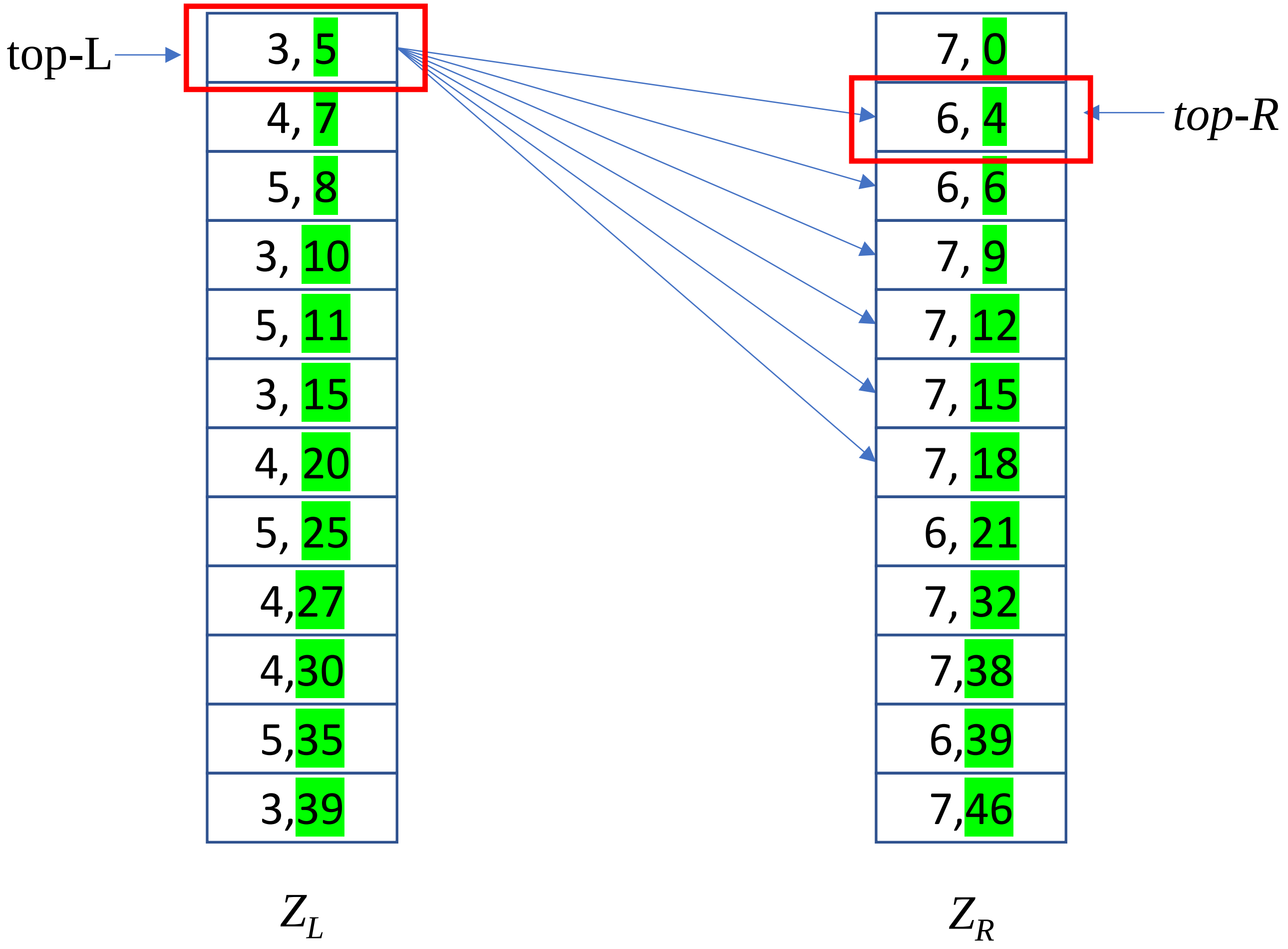
7, 0
6, 4
6, 6
7, 9
7, 12
7, 15
7, 18
6, 21
7, 32
7,38
6,39
7,46

Z_R

$Z_L[top\text{-}L].y > Z_R[top\text{-}R].y + d$, 即5>0+4, Z_L中后续元素y值都不比$Z_L[top\text{-}L].y$小， 因此它们与$Z_R[top\text{-}R]$的距离必然大于*d*, *top-R*增加1

$m=5, d=4$ 更新为 $d=\sqrt{10}$

$Result=dist((3,5),(6,4))$

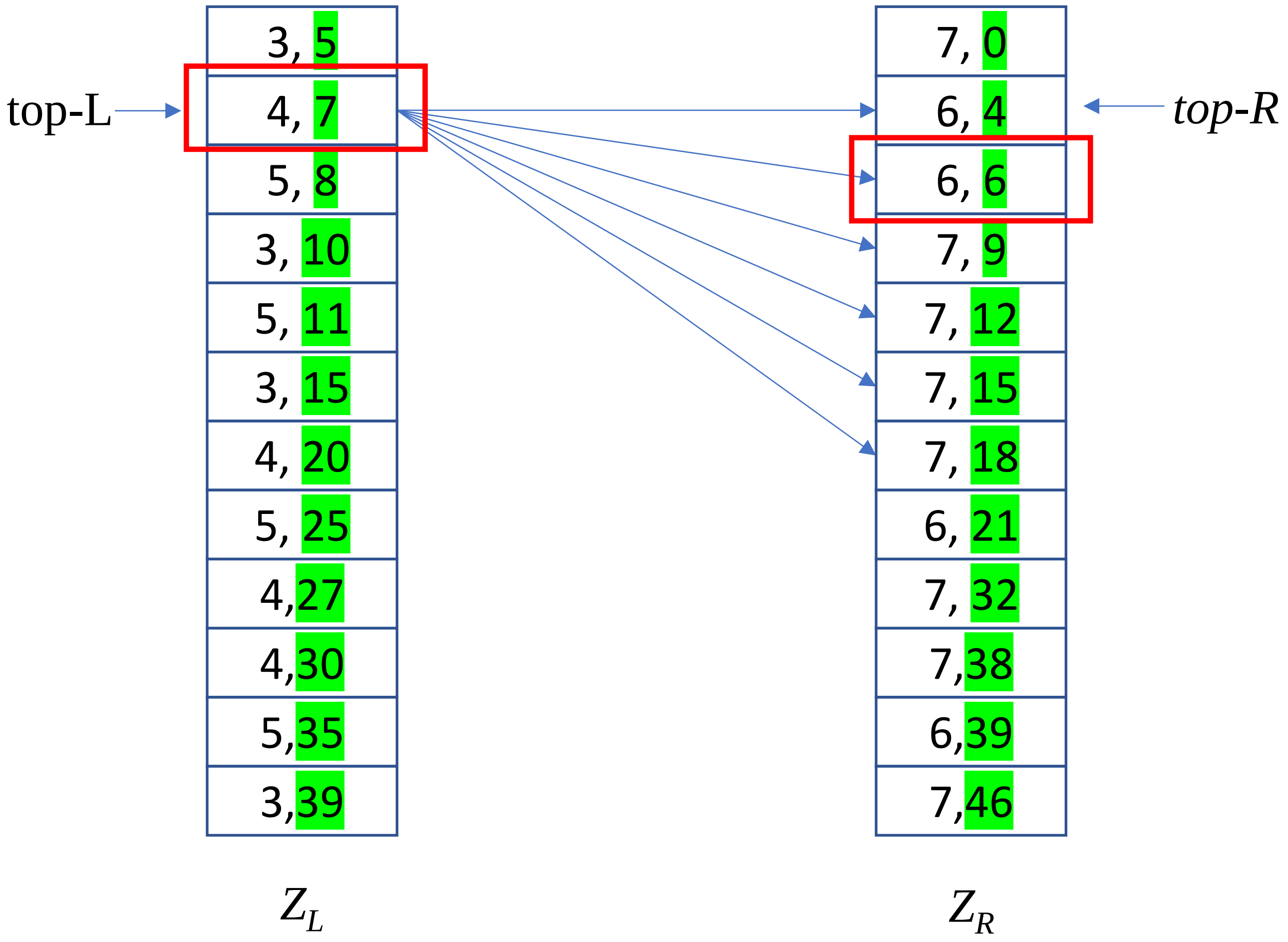


3, 5	7, 0
4, 7	6, 4
5, 8	6, 6
3, 10	7, 9
5, 11	7, 12
3, 15	7, 15
4, 20	7, 18
5, 25	6, 21
4,27	7, 32
4,30	7,38
5,35	6,39
3,39	7,46
Z_L	Z_R

$Z_L[top\text{-}L].y \geq Z_R[top\text{-}R].y - d$ 并且 $Z_L[top\text{-}L].y \leq Z_R[top\text{-}R].y + d$, 为当前 $Z_L[top\text{-}L]$ 连续检验最多6个 Z_R 中的元素（如果 Z_R 中有足够多的剩余元素），随后 $top\text{-}L$ 增长1, $top\text{-}R$ 不变

m=5, $d = \sqrt{10}$更新为$d = \sqrt{5}$

Result=dist((4,7),(6,6))



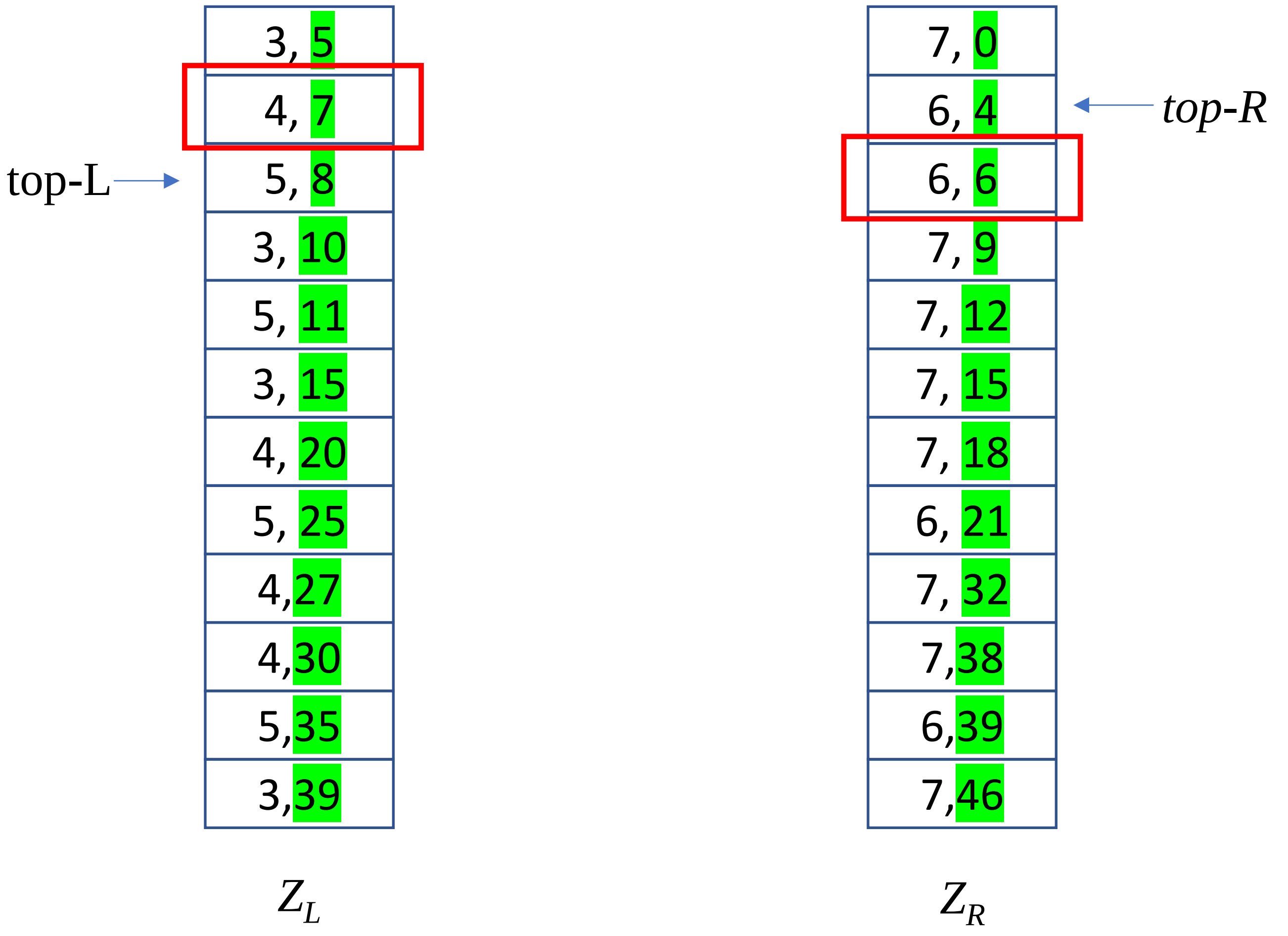
Z_L
3, 5
4, 7
5, 8
3, 10
5, 11
3, 15
4, 20
5, 25
4,27
4,30
5,35
3,39

Z_R
7, 0
6, 4
6, 6
7, 9
7, 12
7, 15
7, 18
6, 21
7, 32
7,38
6,39
7,46

$Z_L[top\text{-}L].y \geq Z_R[top\text{-}R].y - d$ 并且 $Z_L[top\text{-}L].y \leq Z_R[top\text{-}R].y + d$, 为当前$Z_L[top\text{-}L]$ 连续检验最多6个Z_R中的元素（如果Z_R中有足够多的剩余元素），随后*top-L*增长1, *top-R*不变

$m=5, d=\sqrt{5}$

$Result=dist((4,7),(6,6))$



3, 5
4, 7
5, 8
3, 10
5, 11
3, 15
4, 20
5, 25
4,27
4,30
5,35
3,39

Z_L

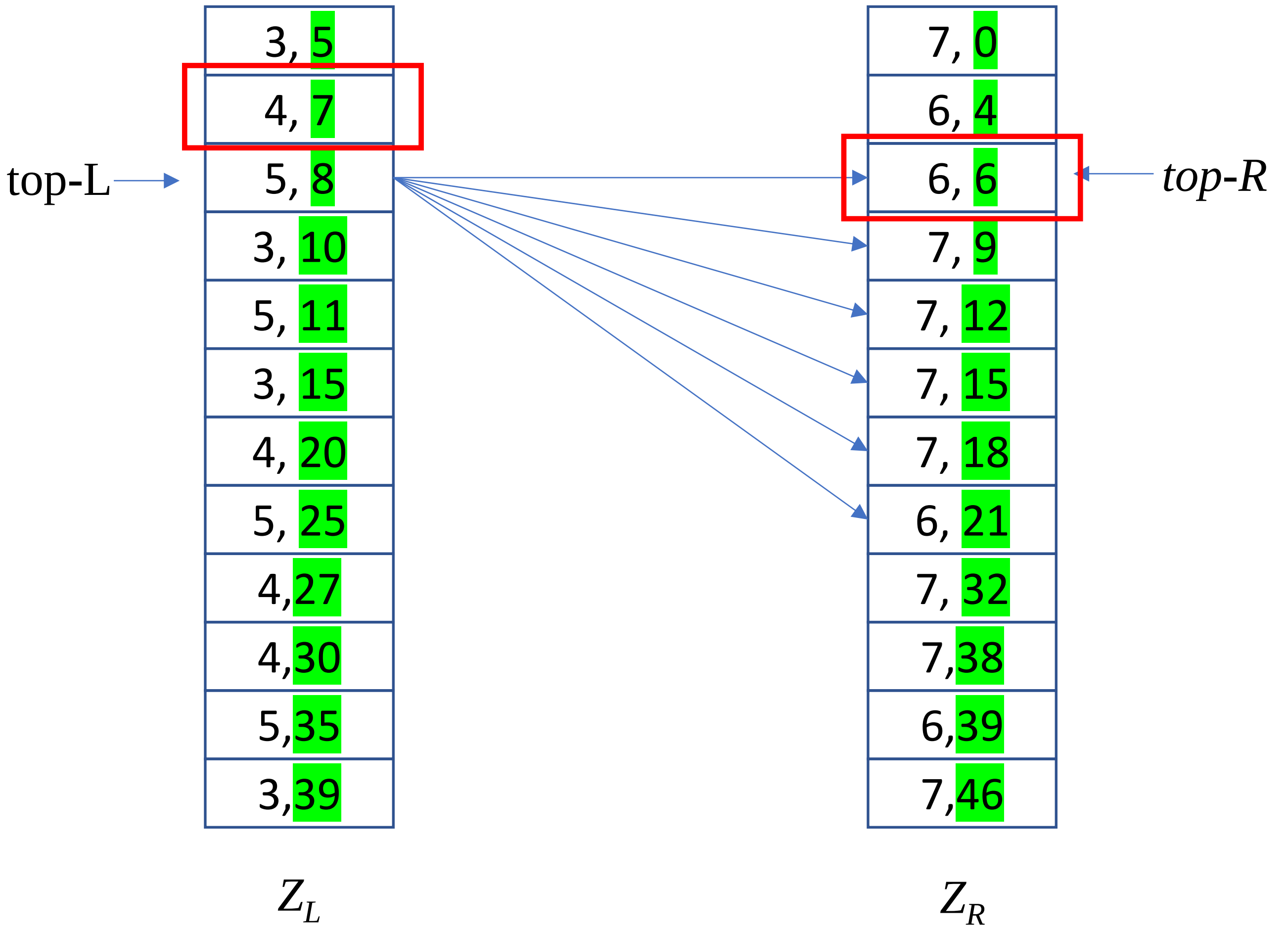
7, 0
6, 4
6, 6
7, 9
7, 12
7, 15
7, 18
6, 21
7, 32
7,38
6,39
7,46

Z_R

$Z_L[top\text{-}L].y > Z_R[top\text{-}R].y + d$, Z_L中后续元素y值都不比$Z_L[top\text{-}L].y$小， 因此它们与$Z_R[top\text{-}R]$的距离必然大于d, $top\text{-}R$增加1

$m=5, d=\sqrt{5}$

$Result=dist((4,7),(6,6))$

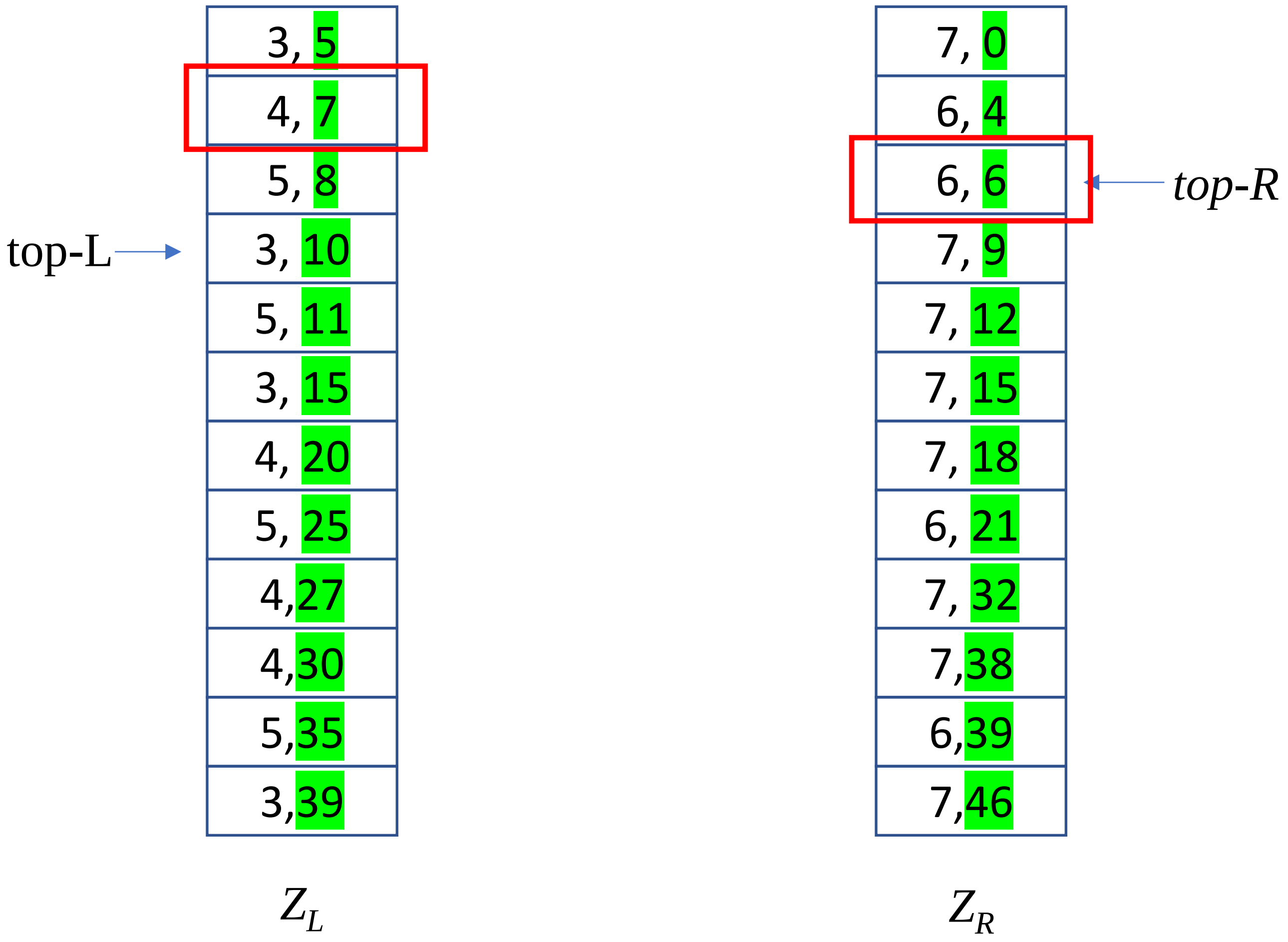


3, 5	7, 0
4, 7	6, 4
5, 8	6, 6
3, 10	7, 9
5, 11	7, 12
3, 15	7, 15
4, 20	7, 18
5, 25	6, 21
4,27	7, 32
4,30	7,38
5,35	6,39
3,39	7,46
Z_L	Z_R

$Z_L[top\text{-}L].y \geq Z_R[top\text{-}R].y - d$ 并且 $Z_L[top\text{-}L].y \leq Z_R[top\text{-}R].y + d$, 为当前$Z_L[top\text{-}L]$ 连续检验最多6个Z_R中的元素（如果Z_R中有足够多的剩余元素），随后*top-L*增长1, *top-R*不变

$m=5,d=\sqrt{5}$

$Result=dist((4,7),(6,6))$

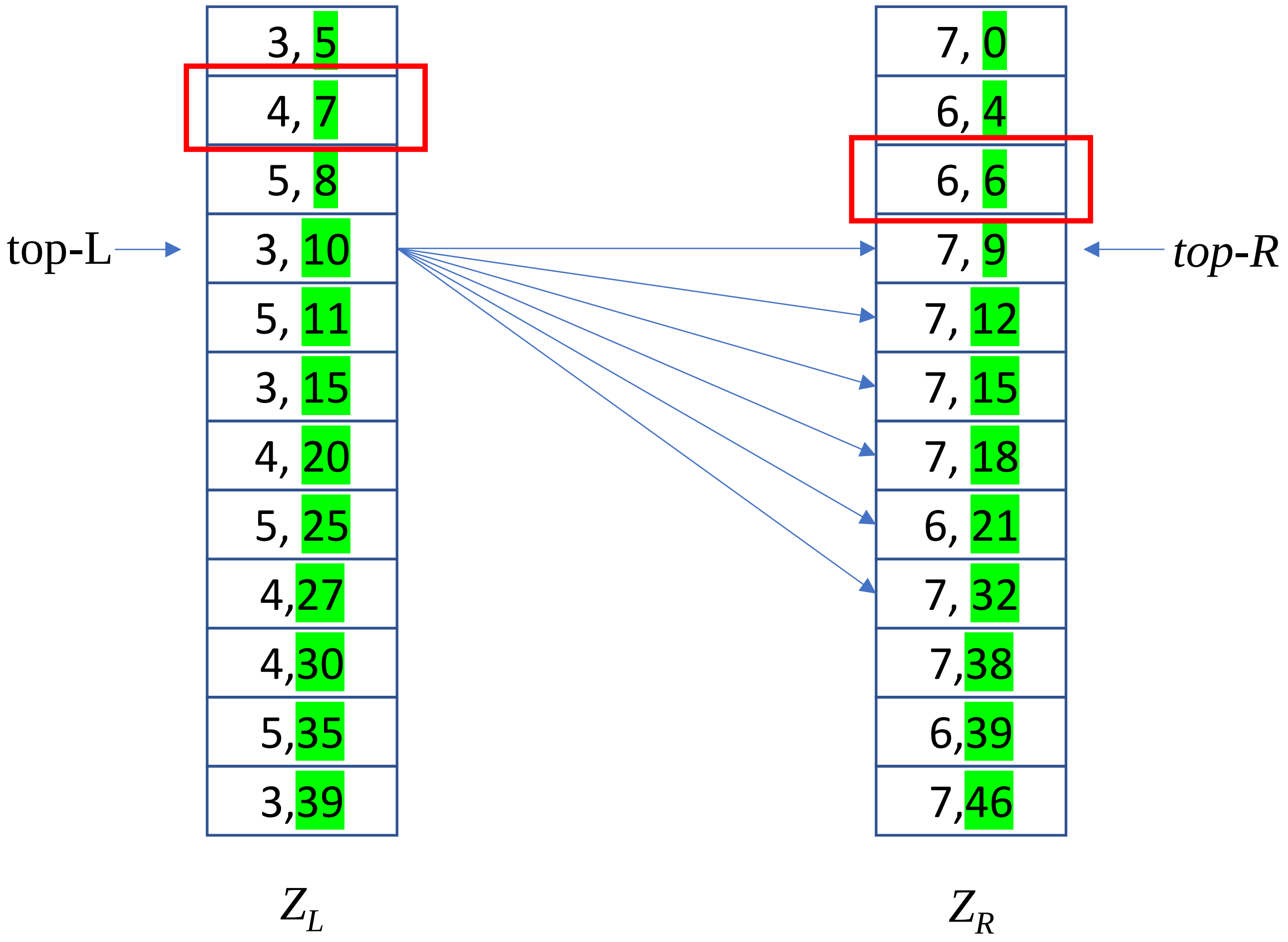


3, 5	7, 0
4, 7	6, 4
5, 8	6, 6
3, 10	7, 9
5, 11	7, 12
3, 15	7, 15
4, 20	7, 18
5, 25	6, 21
4,27	7, 32
4,30	7,38
5,35	6,39
3,39	7,46
Z_L	Z_R

$Z_L[top\text{-}L].y > Z_R[top\text{-}R].y + d$, Z_L中后续元素y值都不比$Z_L[top\text{-}L].y$小，因此它们与$Z_R[top\text{-}R]$的距离必然大于d, top-R增加1

$m=5,\ d=\sqrt{5}$

$Result=dist((4,7),(6,6))$

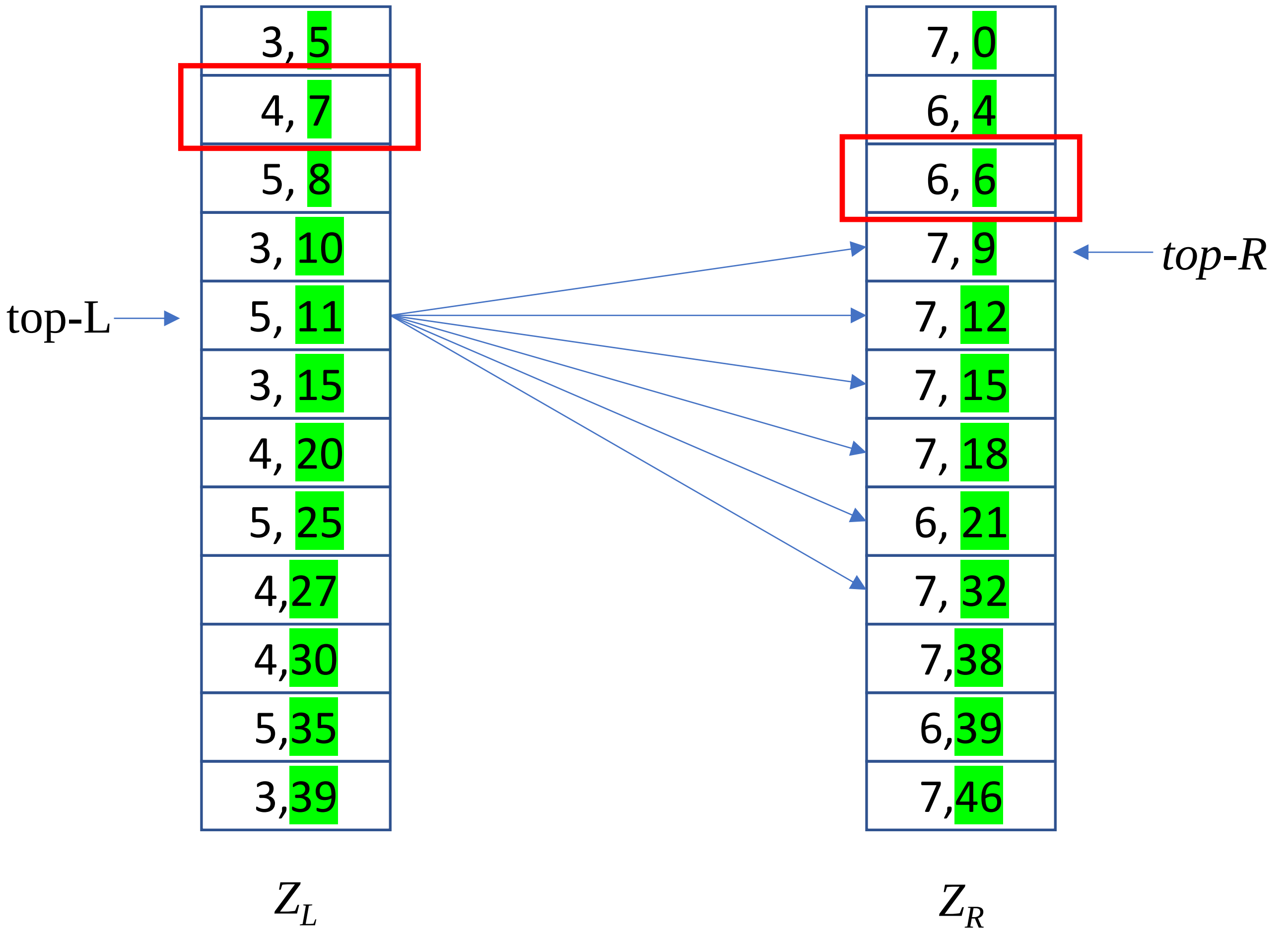


3, 5	7, 0
4, 7	6, 4
5, 8	6, 6
3, 10	7, 9
5, 11	7, 12
3, 15	7, 15
4, 20	7, 18
5, 25	6, 21
4,27	7, 32
4,30	7,38
5,35	6,39
3,39	7,46
Z_L	Z_R

$Z_L[top\text{-}L].y \ge Z_R[top\text{-}R].y - d$ 并且 $Z_L[top\text{-}L].y \le Z_R[top\text{-}R].y + d$, 为当前$Z_L[top\text{-}L]$ 连续检验最多6个Z_R中的元素（如果Z_R中有足够多的剩余元素），随后top-L增长1, top-R不变

$m=5,\ d=\sqrt{5}$

$Result=dist((4,7),(6,6))$

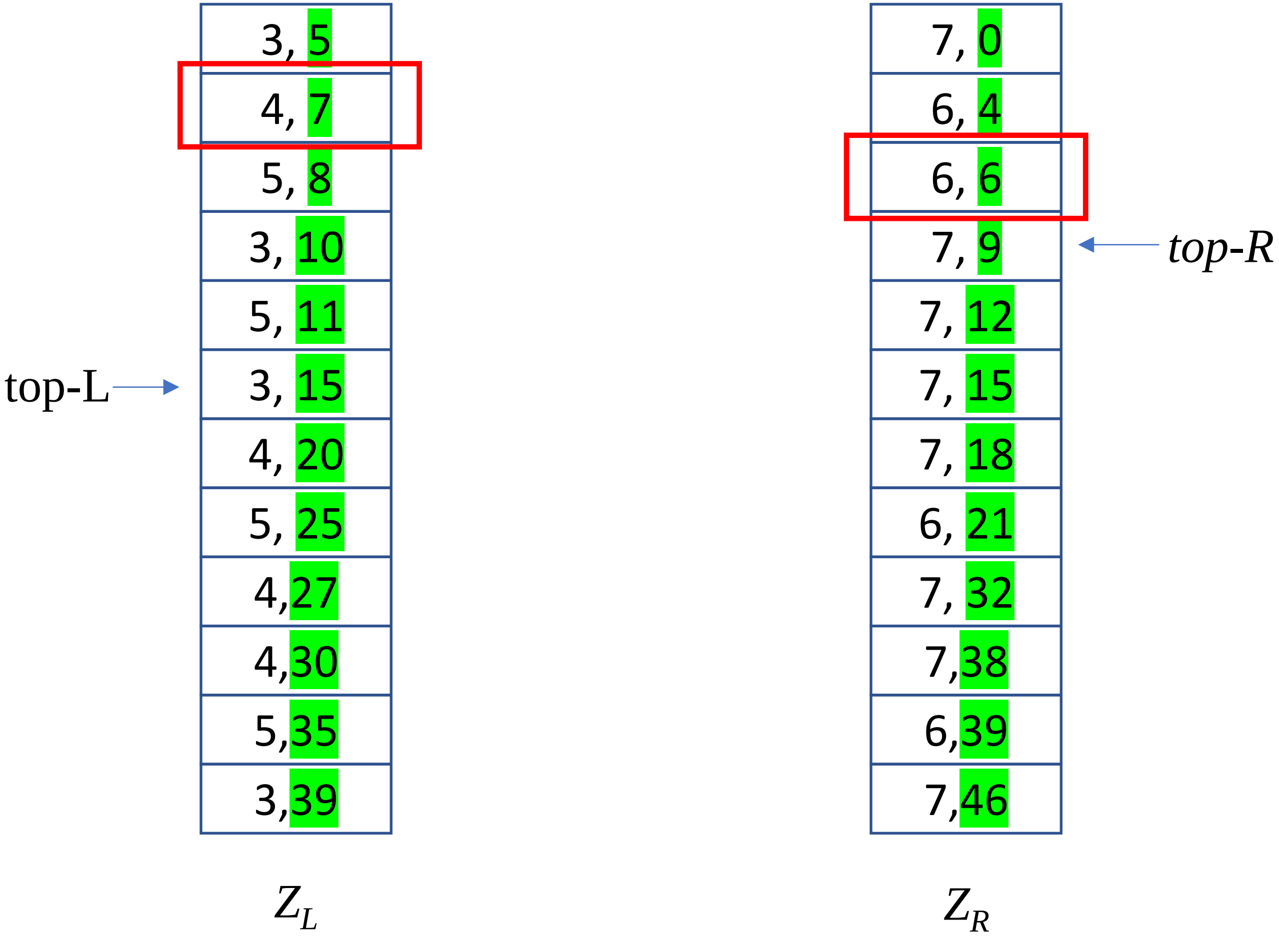


3, 5	7, 0
4, 7	6, 4
5, 8	6, 6
3, 10	7, 9
5, 11	7, 12
3, 15	7, 15
4, 20	7, 18
5, 25	6, 21
4,27	7, 32
4,30	7,38
5,35	6,39
3,39	7,46
Z_L	Z_R

$Z_L[top\text{-}L].y \geq Z_R[top\text{-}R].y - d$ 并且 $Z_L[top\text{-}L].y \leq Z_R[top\text{-}R].y + d$, 为当前$Z_L[top\text{-}L]$ 连续检验最多6个Z_R中的元素（如果Z_R中有足够多的剩余元素），随后top-L增长1, top-R不变

$m=5,\ d=\sqrt{5}$

$Result=dist((4,7),(6,6))$

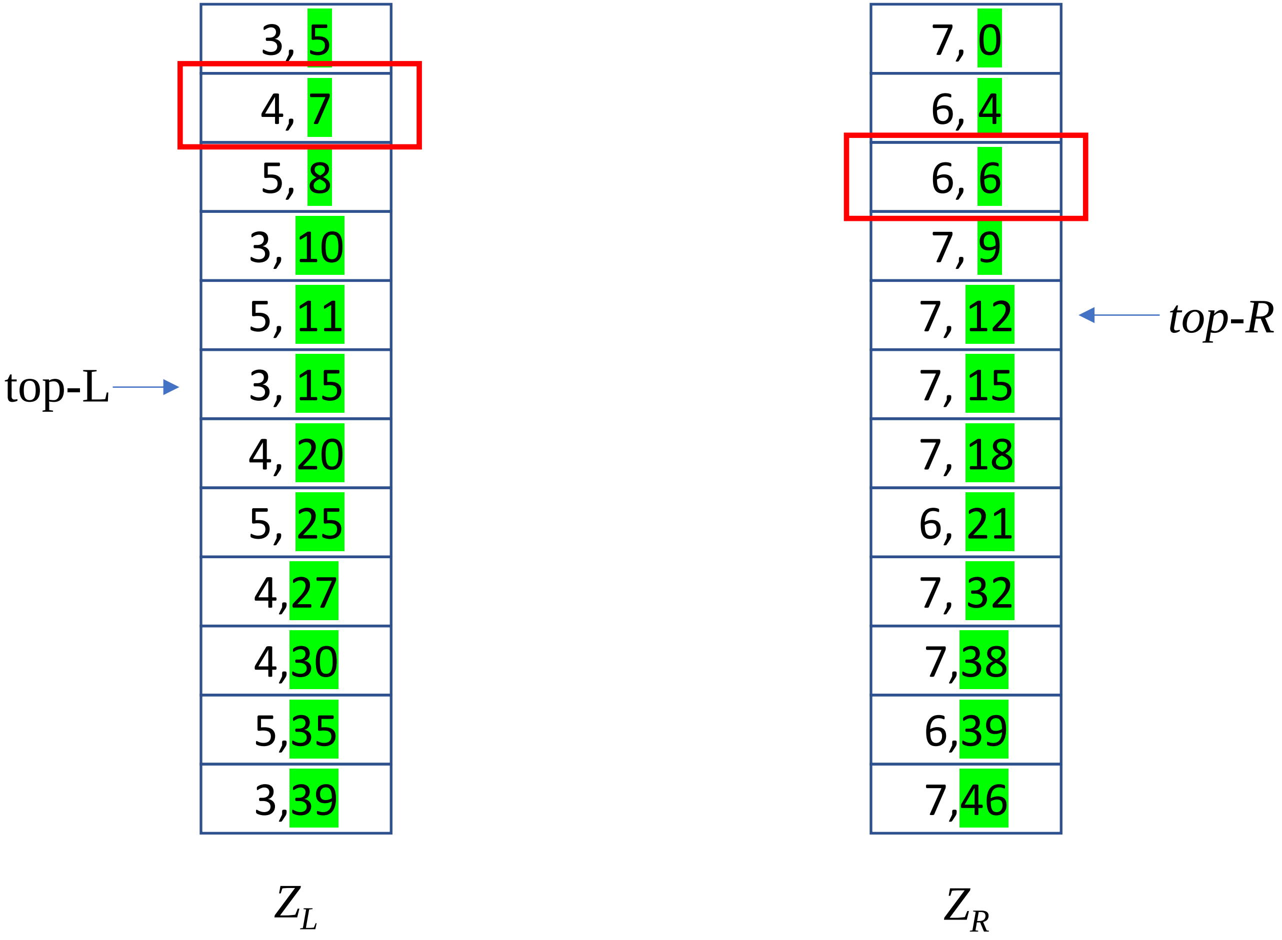


Z_L	Z_R
3, 5	7, 0
4, 7	6, 4
5, 8	6, 6
3, 10	7, 9
5, 11	7, 12
3, 15	7, 15
4, 20	7, 18
5, 25	6, 21
4,27	7, 32
4,30	7,38
5,35	6,39
3,39	7,46

$Z_L[top\text{-}L].y > Z_R[top\text{-}R].y + d$, Z_L中后续元素y值都不比$Z_L[top\text{-}L].y$小，因此它们与$Z_R[top\text{-}R]$的距离必然大于d, top-R增加1

$m=5, d=\sqrt{5}$

$Result=dist((4,7),(6,6))$

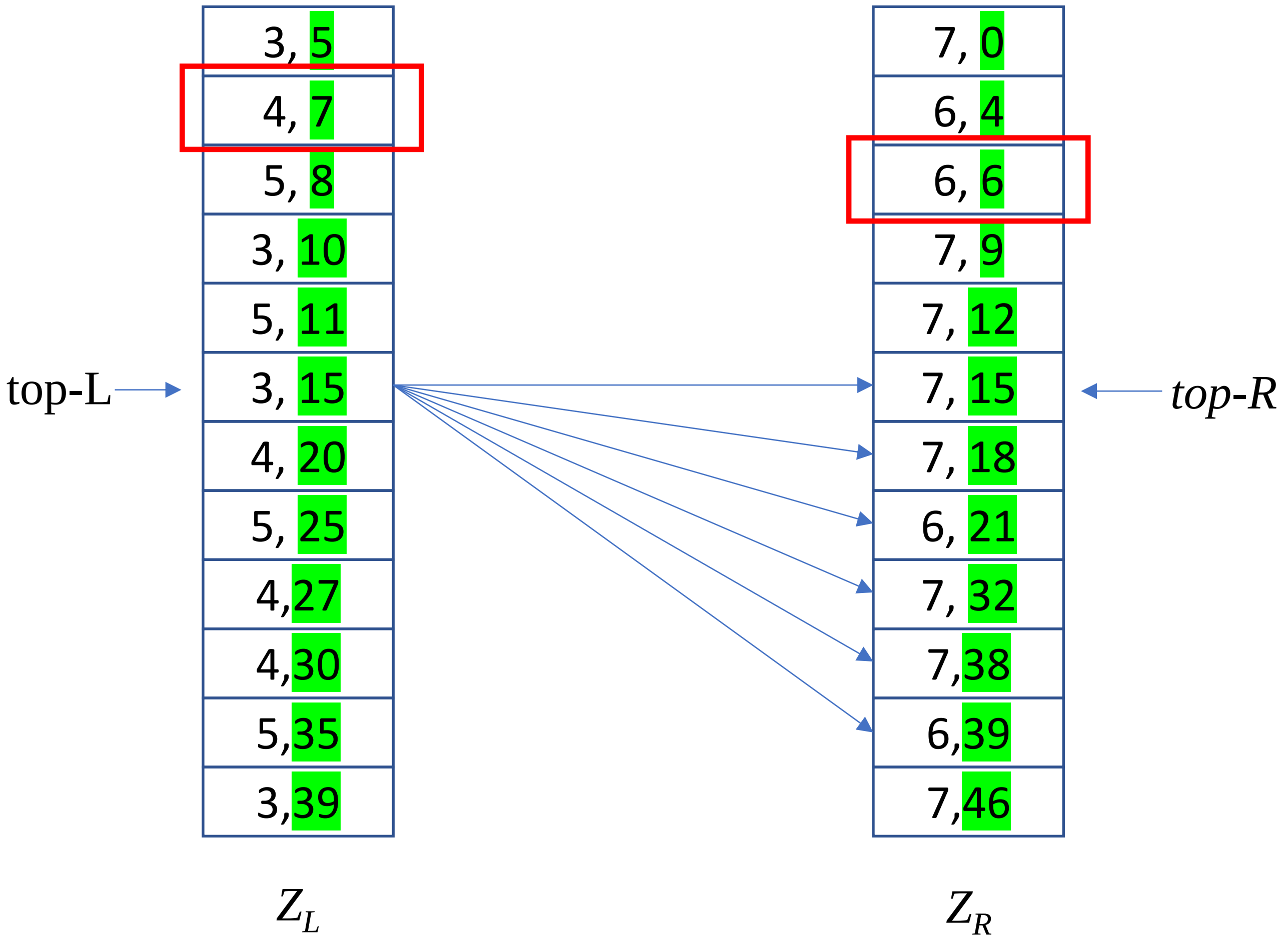


Z_L	Z_R
3, 5	7, 0
4, 7	6, 4
5, 8	6, 6
3, 10	7, 9
5, 11	7, 12
3, 15	7, 15
4, 20	7, 18
5, 25	6, 21
4,27	7, 32
4,30	7,38
5,35	6,39
3,39	7,46

$Z_L[top\text{-}L].y > Z_R[top\text{-}R].y + d$, Z_L中后续元素y值都不比$Z_L[top\text{-}L].y$小，因此它们与$Z_R[top\text{-}R]$的距离必然大于d, $top\text{-}R$增加1

$m=5, d=\sqrt{5}$

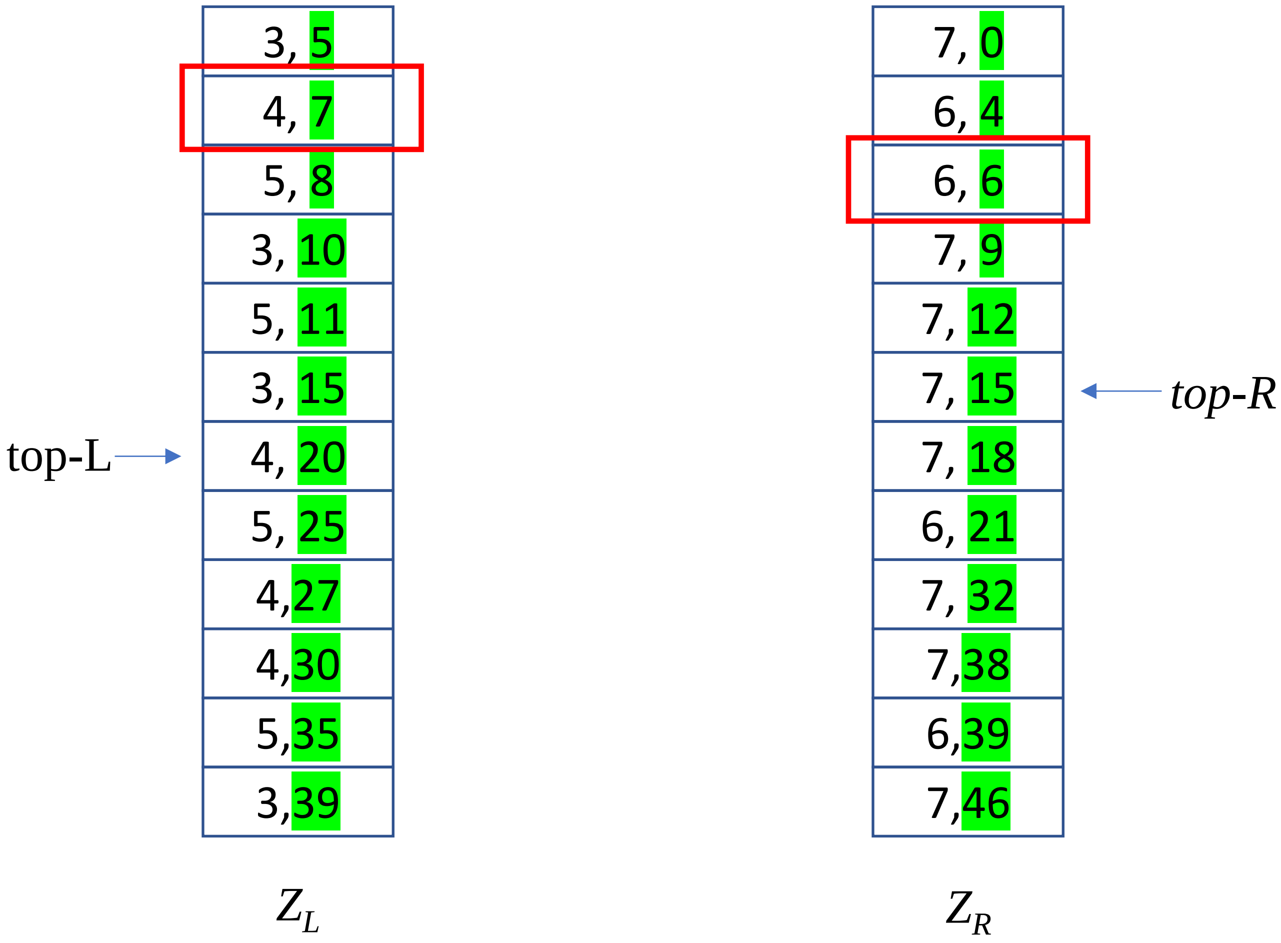
$Result=dist((4,7),(6,6))$



$Z_L[top\text{-}L].y \ge Z_R[top\text{-}R].y - d$ 并且 $Z_L[top\text{-}L].y \le Z_R[top\text{-}R].y + d$, 为当前$Z_L[top\text{-}L]$ 连续检验最多6个Z_R中的元素（如果Z_R中有足够多的剩余元素），随后*top-L*增长1, *top-R*不变

$m=5,\ d=\sqrt{5}$

$Result=dist((4,7),(6,6))$



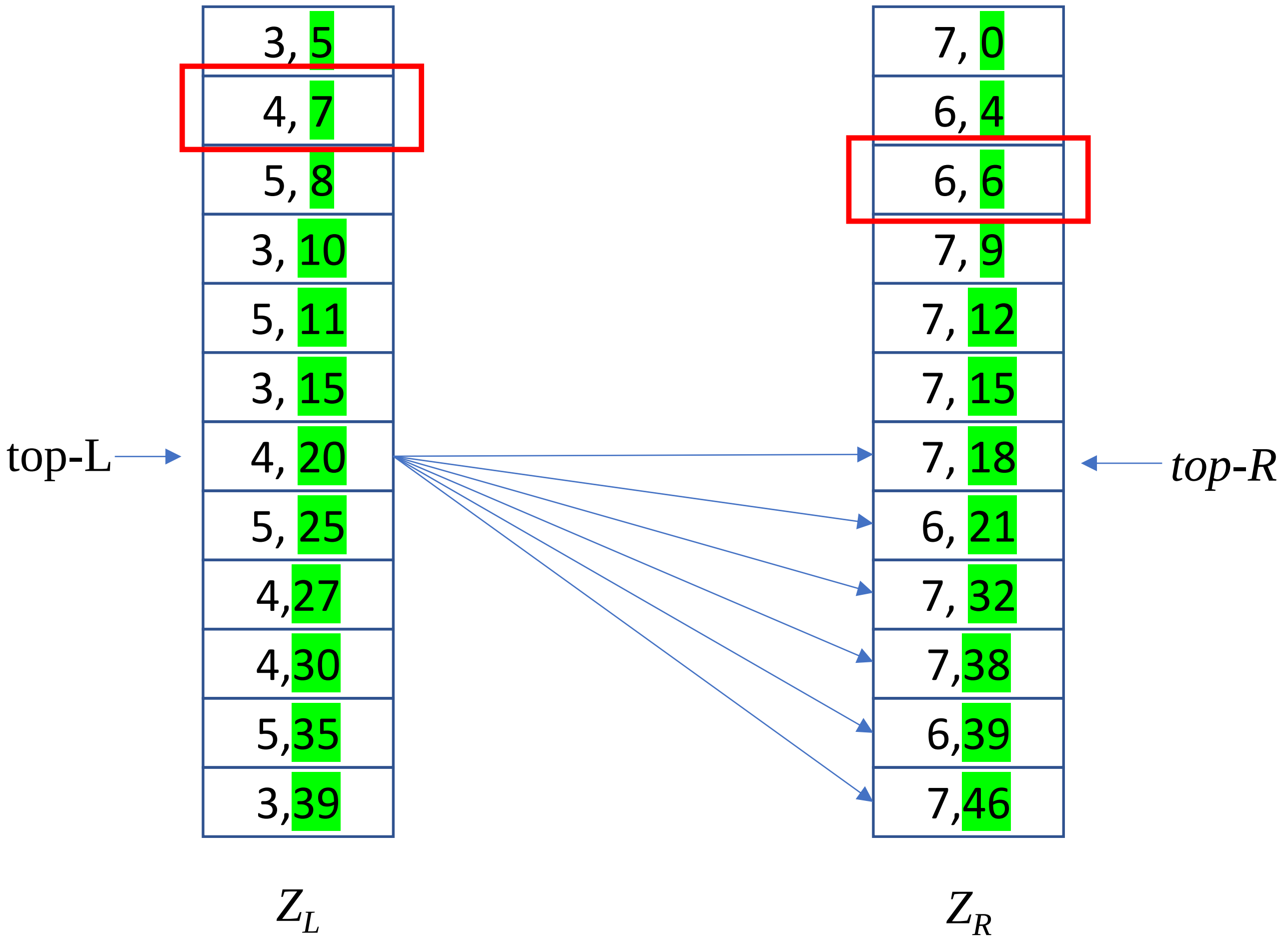
3, 5
4, 7
5, 8
3, 10
5, 11
3, 15
4, 20
5, 25
4,27
4,30
5,35
3,39

7, 0
6, 4
6, 6
7, 9
7, 12
7, 15
7, 18
6, 21
7, 32
7,38
6,39
7,46

$Z_L[top\text{-}L].y > Z_R[top\text{-}R].y + d$, Z_L中后续元素y值都不比$Z_L[top\text{-}L].y$小，因此它们与$Z_R[top\text{-}R]$的距离必然大于d, top-R增加1

$m=5, d=\sqrt{5}$

$Result=dist((4,7),(6,6))$

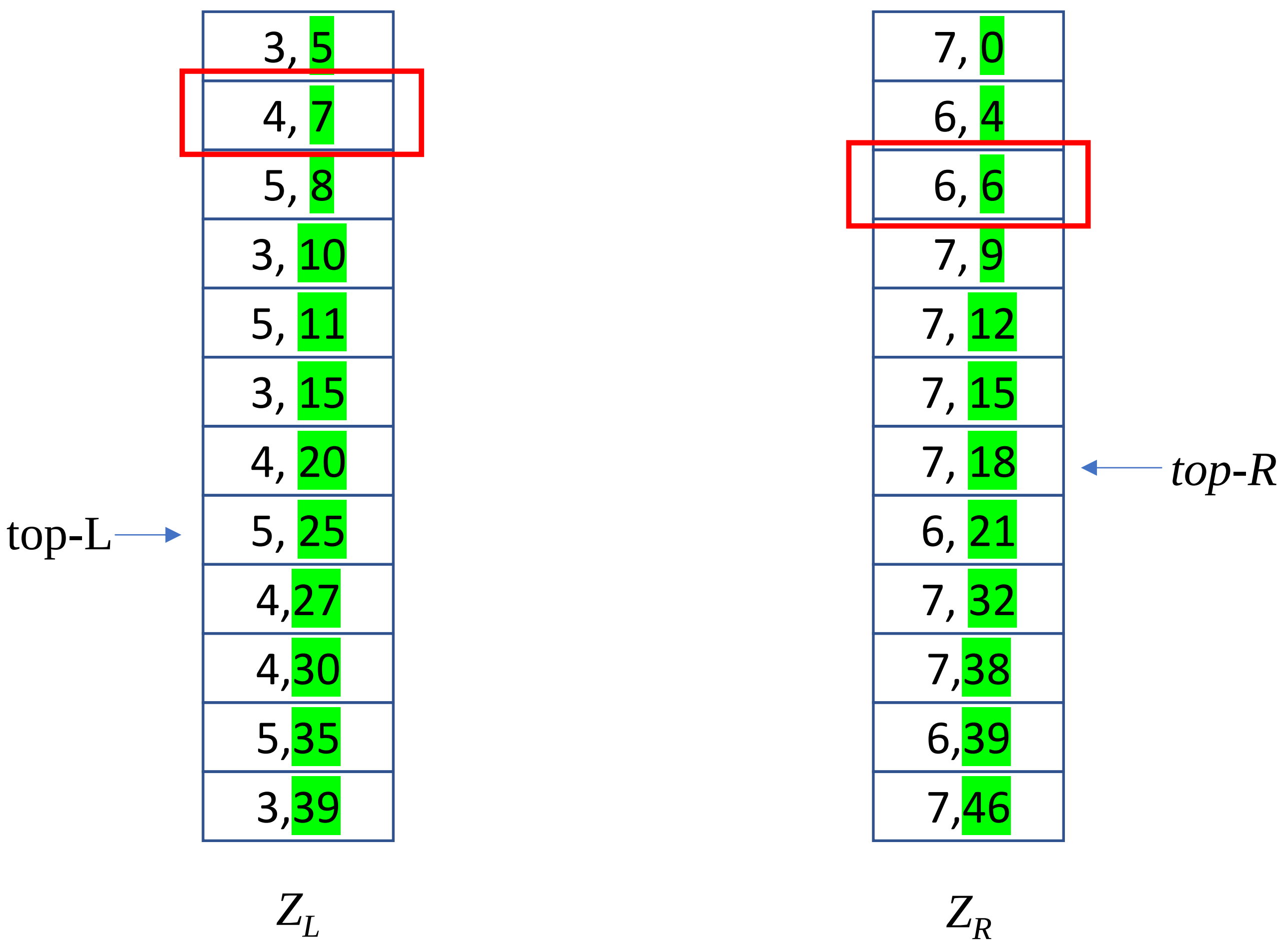


3, 5	7, 0
4, 7	6, 4
5, 8	6, 6
3, 10	7, 9
5, 11	7, 12
3, 15	7, 15
4, 20	7, 18
5, 25	6, 21
4,27	7, 32
4,30	7,38
5,35	6,39
3,39	7,46
Z_L	Z_R

$Z_L[top\text{-}L].y \ge Z_R[top\text{-}R].y - d$ 并且 $Z_L[top\text{-}L].y \le Z_R[top\text{-}R].y + d$, 为当前$Z_L[top\text{-}L]$ 连续检验最多6个Z_R中的元素（如果Z_R中有足够多的剩余元素），随后*top-L*增长1, *top-R*不变

$m=5$, $d=\sqrt{5}$

$Result=dist((4,7),(6,6))$



3, 5
4, 7
5, 8
3, 10
5, 11
3, 15
4, 20
5, 25
4,27
4,30
5,35
3,39

Z_L

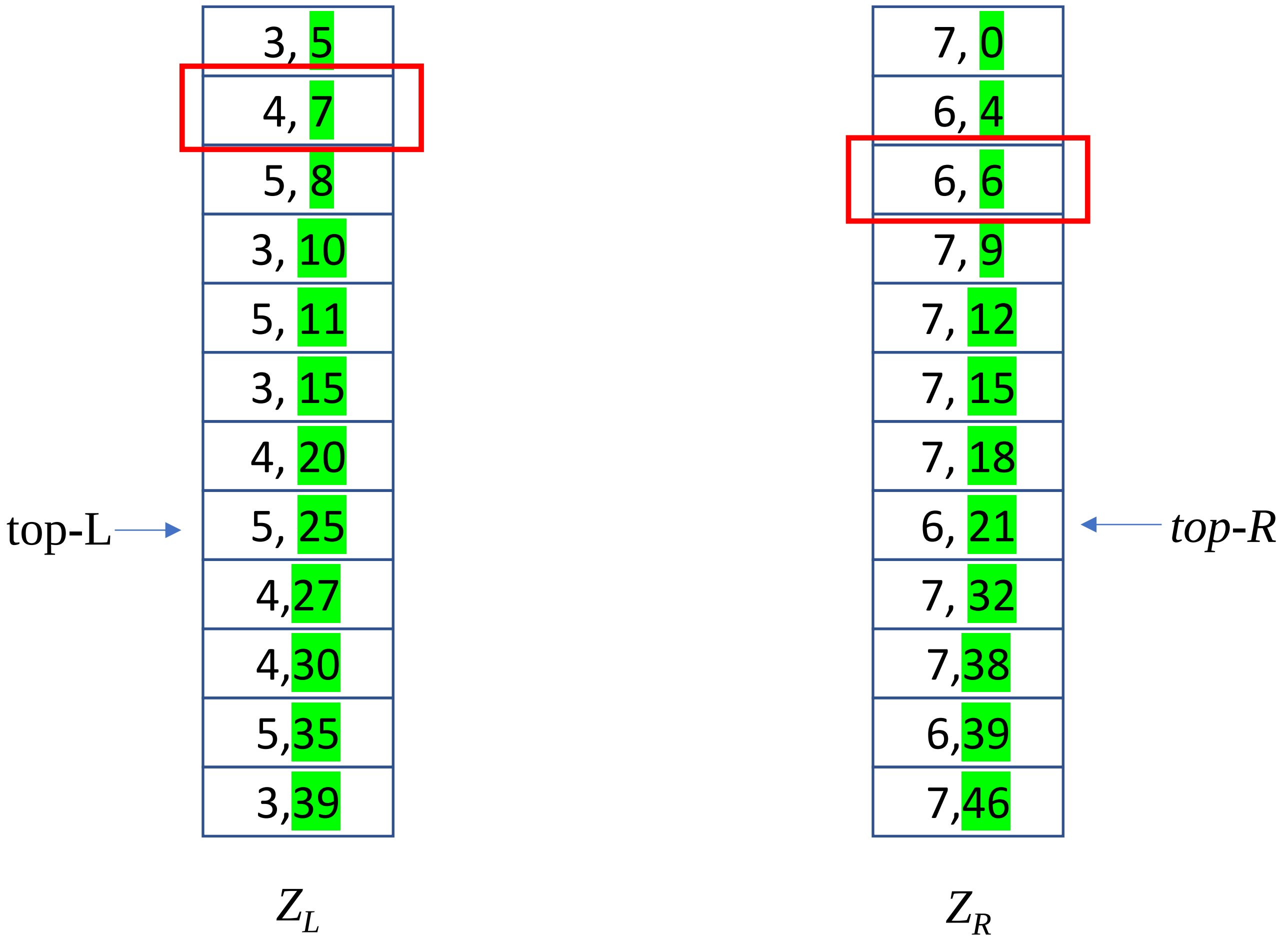
7, 0
6, 4
6, 6
7, 9
7, 12
7, 15
7, 18
6, 21
7, 32
7,38
6,39
7,46

Z_R

$Z_L[top\text{-}L].y > Z_R[top\text{-}R].y + d$, Z_L中后续元素y值都不比$Z_L[top\text{-}L].y$小，因此它们与$Z_R[top\text{-}R]$的距离必然大于d, $top\text{-}R$增加1

$m=5, d=\sqrt{5}$

$Result=dist((4,7),(6,6))$



3, 5
4, 7
5, 8
3, 10
5, 11
3, 15
4, 20
5, 25
4,27
4,30
5,35
3,39

Z_L

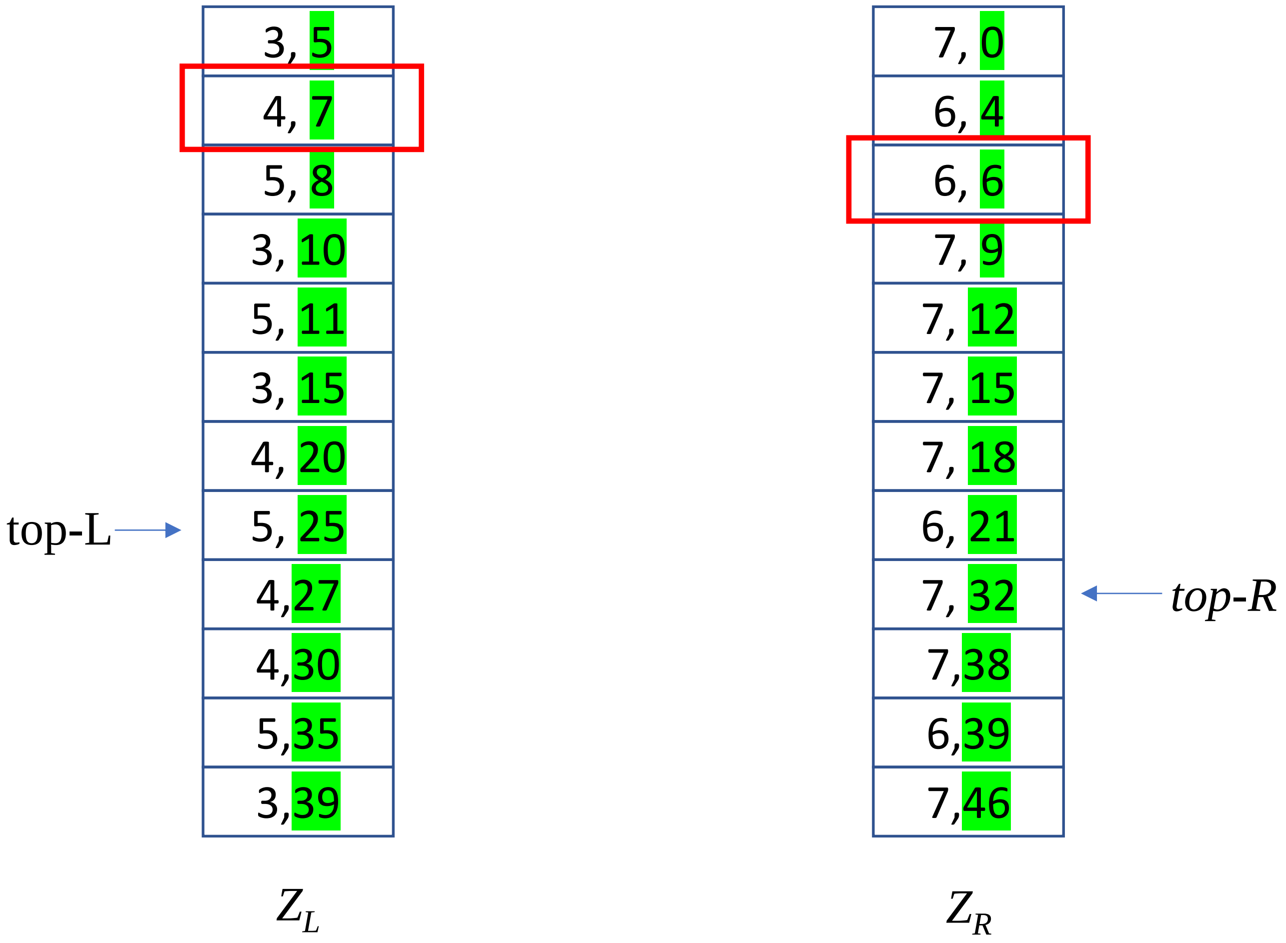
7, 0
6, 4
6, 6
7, 9
7, 12
7, 15
7, 18
6, 21
7, 32
7,38
6,39
7,46

Z_R

$Z_L[top\text{-}L].y > Z_R[top\text{-}R].y + d$, Z_L中后续元素y值都不比$Z_L[top\text{-}L].y$小，因此它们与$Z_R[top\text{-}R]$的距离必然大于d, *top-R*增加1

$m=5$, $d=\sqrt{5}$

$Result=dist((4,7),(6,6))$

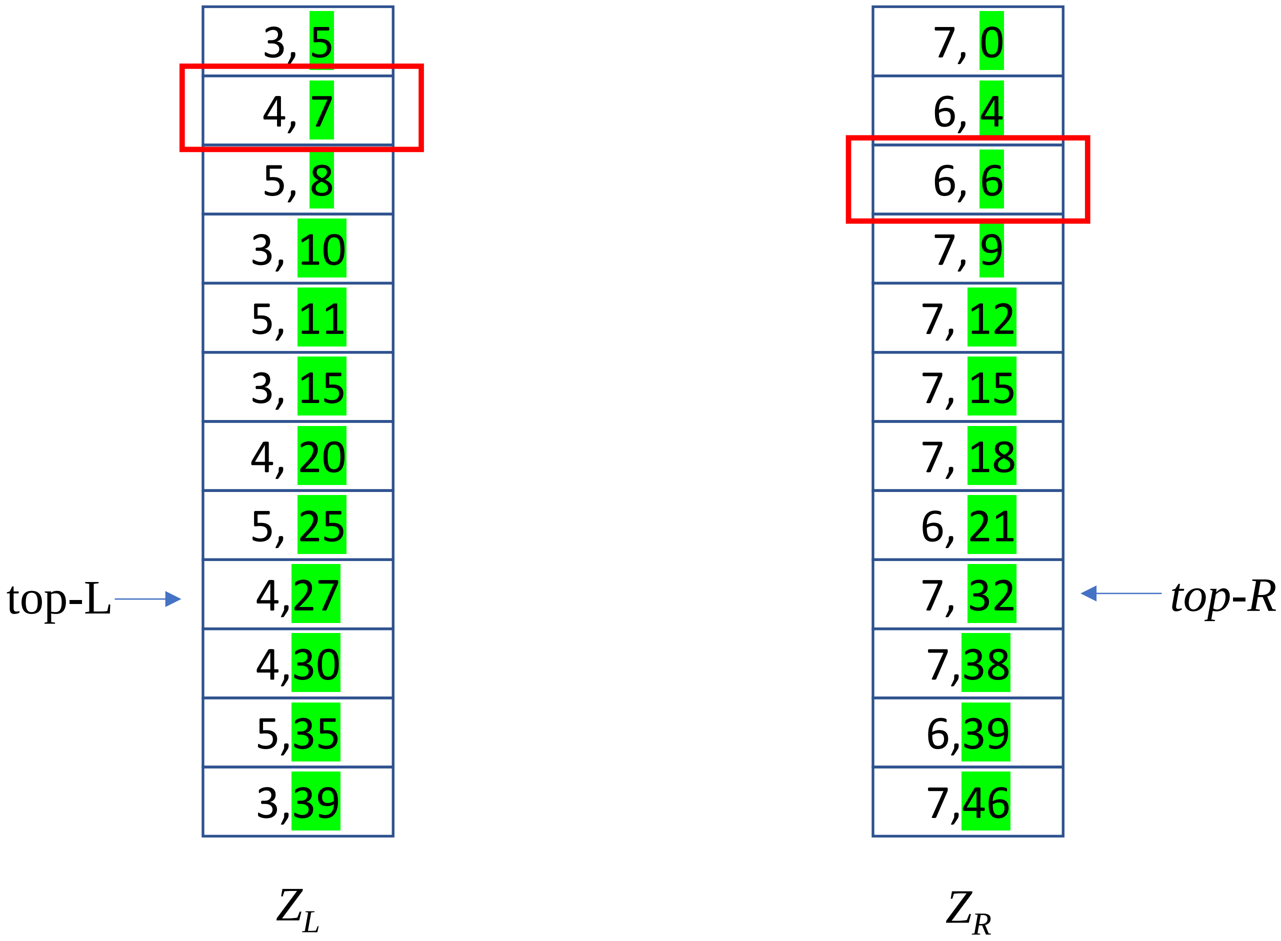


Z_L	Z_R
3, 5	7, 0
4, 7	6, 4
5, 8	6, 6
3, 10	7, 9
5, 11	7, 12
3, 15	7, 15
4, 20	7, 18
5, 25	6, 21
4,27	7, 32
4,30	7,38
5,35	6,39
3,39	7,46

$Z_L[top\text{-}L].y < Z_R[top\text{-}R].y - d$, Z_R中后续元素y值都不比$Z_R[top\text{-}R].y$小，因此它们与$Z_L[top\text{-}L]$的距离必然大于d, top-L增加1

$m=5, d=\sqrt{5}$

$Result=dist((4,7),(6,6))$

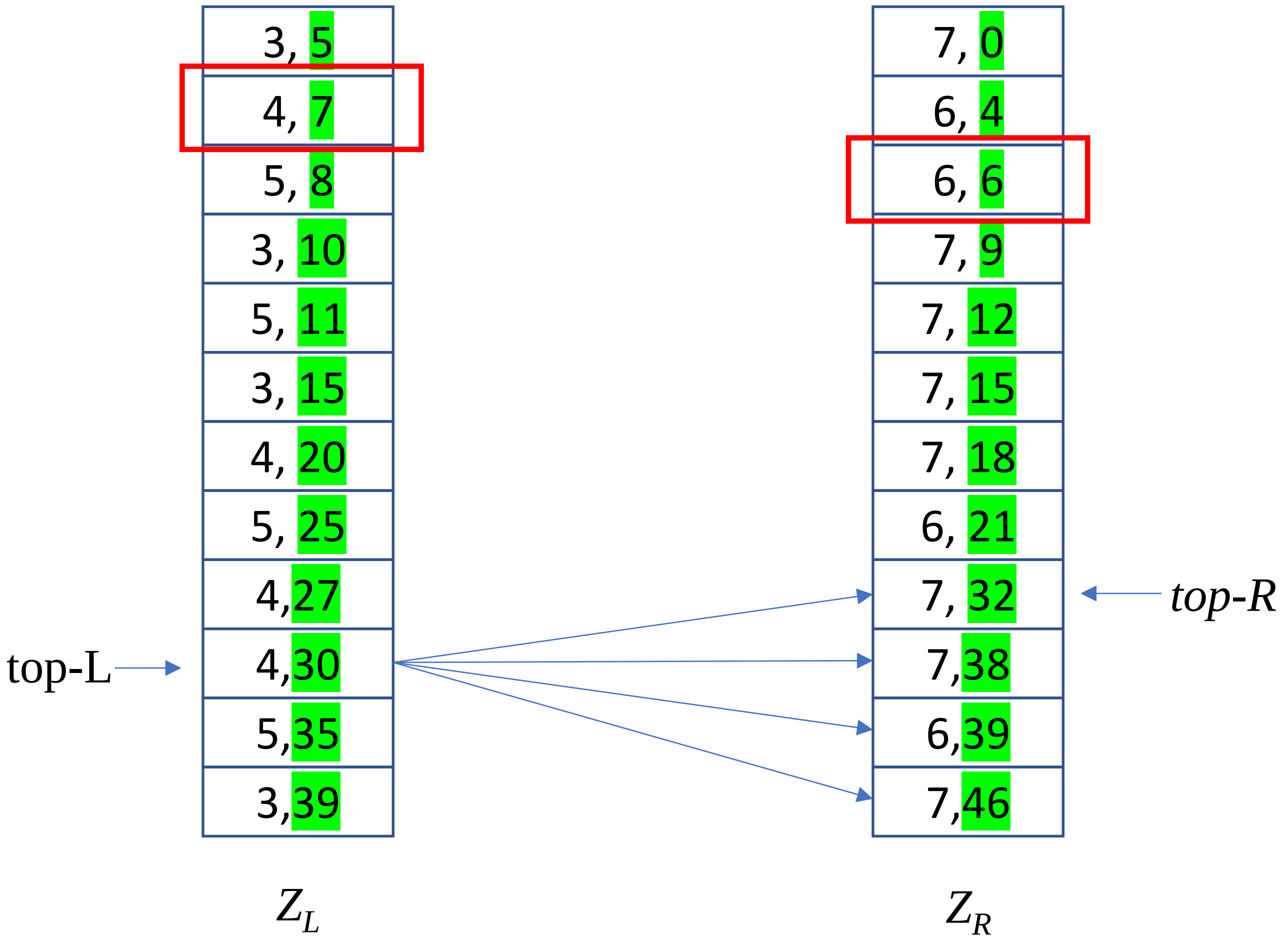


3, 5	7, 0
4, 7	6, 4
5, 8	6, 6
3, 10	7, 9
5, 11	7, 12
3, 15	7, 15
4, 20	7, 18
5, 25	6, 21
4,27	7, 32
4,30	7,38
5,35	6,39
3,39	7,46
Z_L	Z_R

$Z_L[top\text{-}L].y < Z_R[top\text{-}R].y - d$, Z_R中后续元素y值都不比$Z_R[top\text{-}R].y$小，因此它们与$Z_L[top\text{-}L]$的距离必然大于d, top-L增加1

$m=5, d=\sqrt{5}$

$Result=dist((4,7),(6,6))$

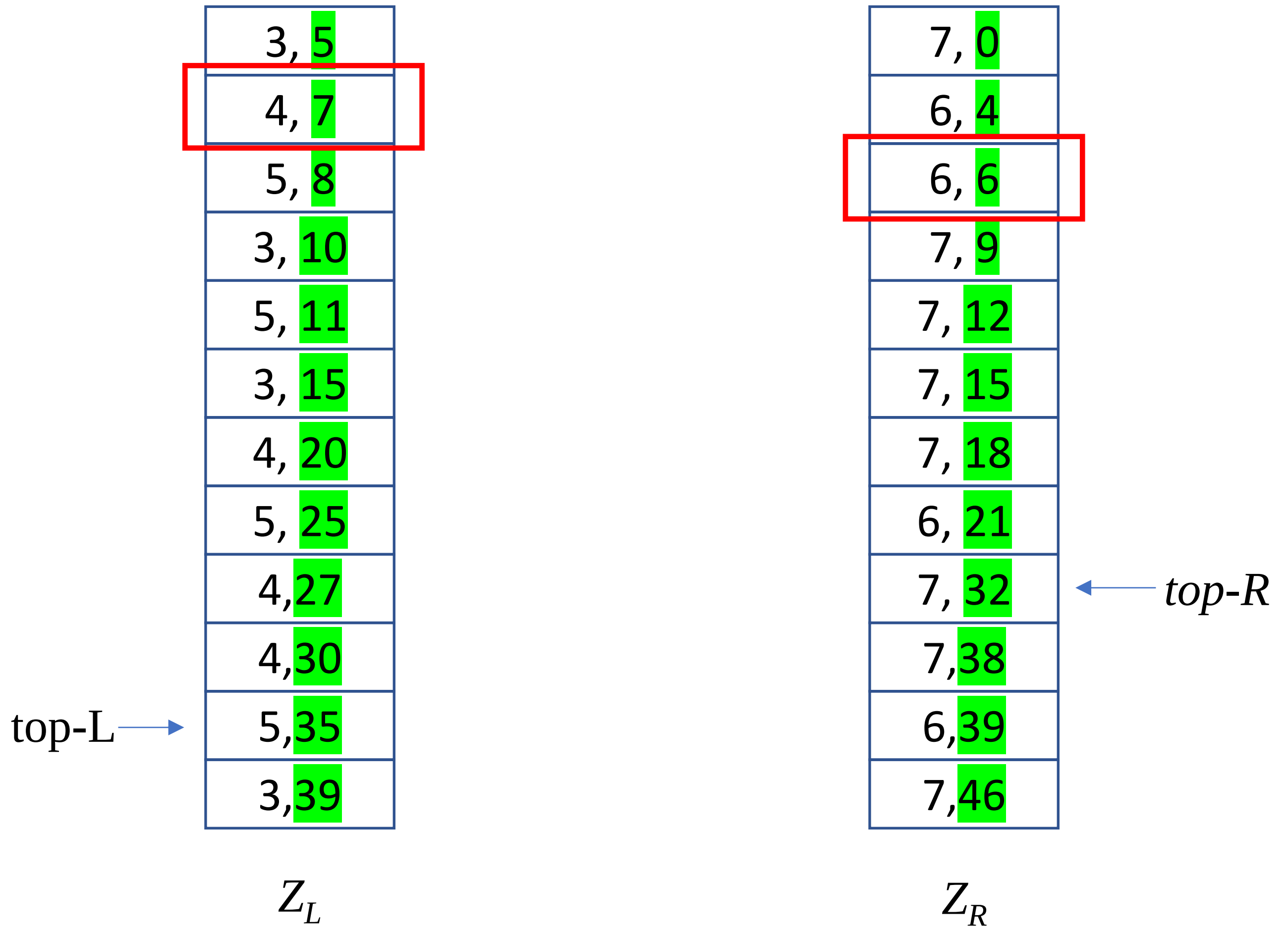


3, 5	7, 0
4, 7	6, 4
5, 8	6, 6
3, 10	7, 9
5, 11	7, 12
3, 15	7, 15
4, 20	7, 18
5, 25	6, 21
4,27	7, 32
4,30	7,38
5,35	6,39
3,39	7,46
Z_L	Z_R

$Z_L[top\text{-}L].y \ge Z_R[top\text{-}R].y - d$ 并且 $Z_L[top\text{-}L].y \le Z_R[top\text{-}R].y + d$, 为当前$Z_L[top\text{-}L]$ 连续检验最多6个Z_R中的元素（如果Z_R中有足够多的剩余元素），随后top-L增长1, top-R不变

$m=5, d=\sqrt{5}$

$Result=dist((4,7),(6,6))$

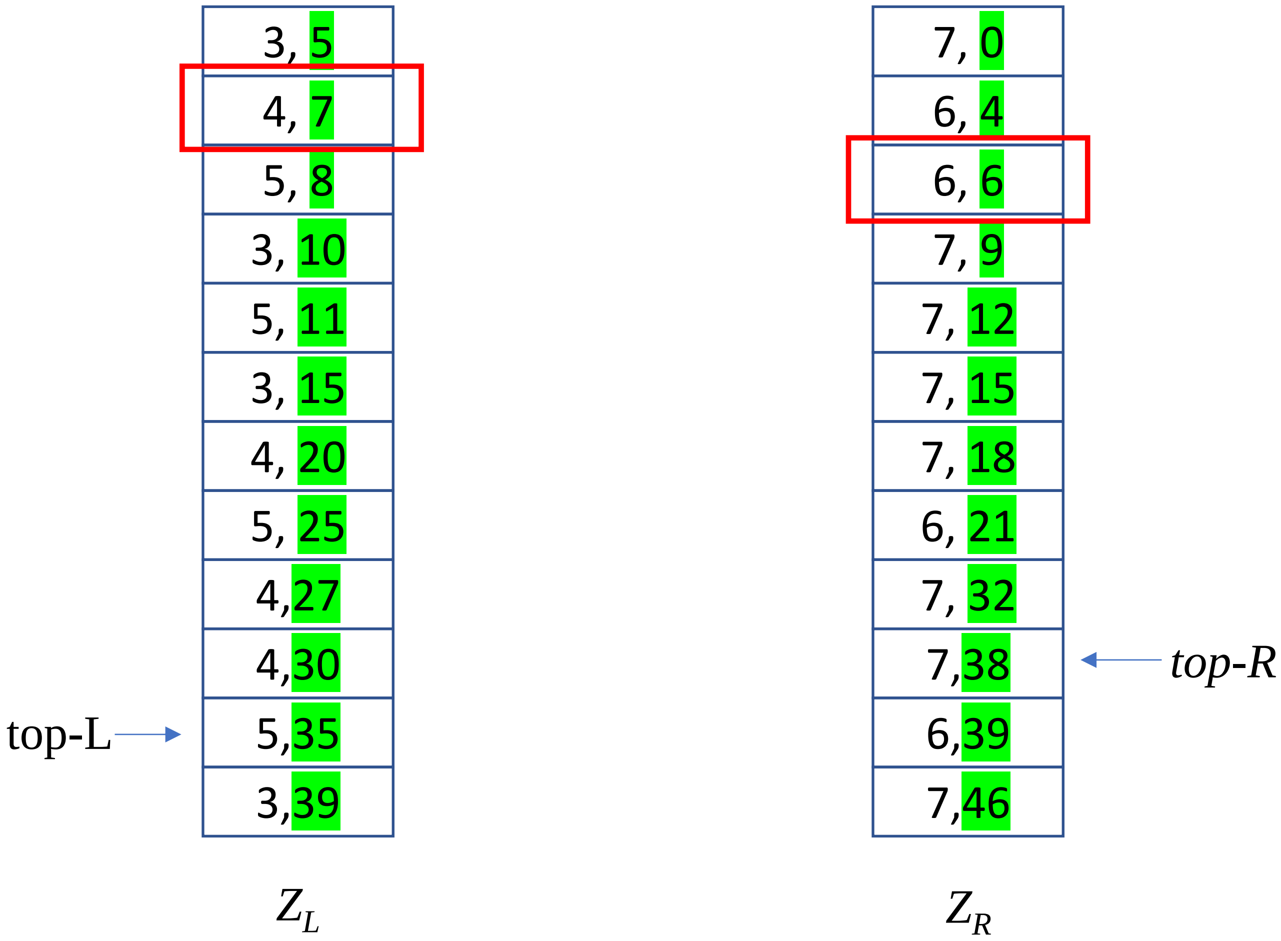


3, 5	7, 0
4, 7	6, 4
5, 8	6, 6
3, 10	7, 9
5, 11	7, 12
3, 15	7, 15
4, 20	7, 18
5, 25	6, 21
4,27	7, 32
4,30	7,38
5,35	6,39
3,39	7,46
Z_L	Z_R

$Z_L[top\text{-}L].y > Z_R[top\text{-}R].y + d$, Z_L中后续元素y值都不比$Z_L[top\text{-}L].y$小，因此它们与$Z_R[top\text{-}R]$的距离必然大于d, top-R增加1

$m=5, d=\sqrt{5}$

Result=dist((4,7),(6,6))

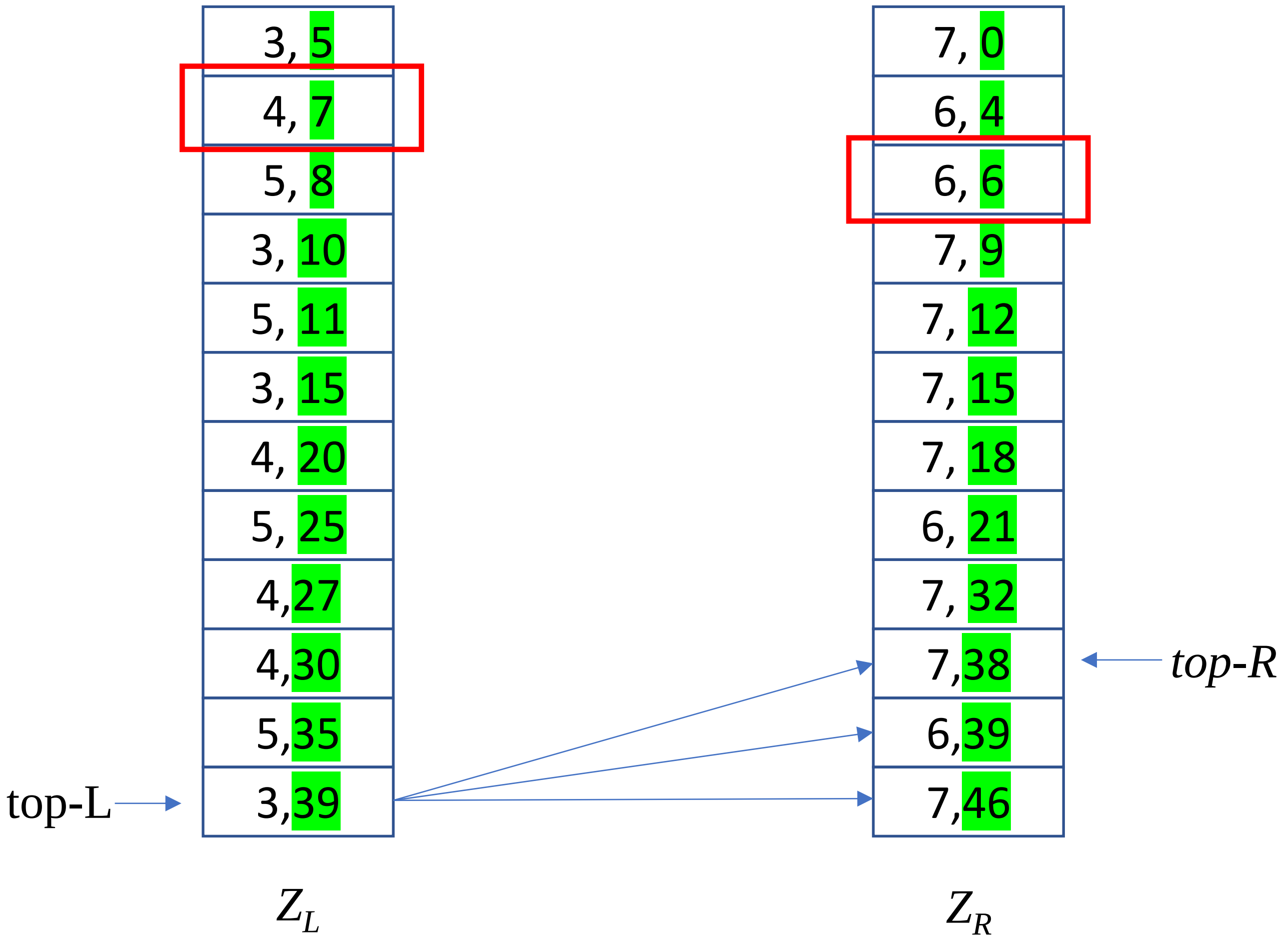


3, 5	7, 0
4, 7	6, 4
5, 8	6, 6
3, 10	7, 9
5, 11	7, 12
3, 15	7, 15
4, 20	7, 18
5, 25	6, 21
4,27	7, 32
4,30	7,38
5,35	6,39
3,39	7,46
Z_L	Z_R

Z_L[*top-L*].*y* < Z_R[*top-R*].*y* - *d*, Z_R中后续元素*y*值都不比Z_R[*top-R*].*y*小，因此它们与Z_L[*top-L*]的距离必然大于*d*, *top-L*增加1

$m=5,\ d=\sqrt{5}$

$Result=dist((4,7),(6,6))$



$Z_L[top\text{-}L].y \ge Z_R[top\text{-}R].y - d$ 并且 $Z_L[top\text{-}L].y \le Z_R[top\text{-}R].y + d$, 为当前$Z_L[top\text{-}L]$ 连续检验最多6个Z_R中的元素（如果Z_R中有足够多的剩余元素），随后top-L增长1, top-R不变. 算法结束.

- (p_l, q_r)搜索算法时间复杂性
  - 获取Z_L和Z_R需要$O(n)$时间
  - 每次使得*top-L*增加1或者*top-R*增加1需要消耗常数时间
  - 算法结束时, *top-L*和*top-R*总共增长$O(n)$, 外层for循环耗时$O(n)$
  - 算法时间复杂性为$O(n)$

(p_l, q_r)搜索算法

```
Input: $Y_L$, $Y_R$, $d$
Output: $result$
扫描$Y_L$得到$Q_L$中左临界区点，保持y坐标排序，得到$Z_L$
扫描$Y_R$得到$Q_R$中左临界区点，保持y坐标排序，得到$Z_R$
$result$=null;
$top\text{-}R$=0;
for $top\text{-}L$=0 to $Z_L.length$-1
    while $Z_L[top\text{-}L].y > Z_R[top\text{-}R].y + d$
        if $top\text{-}R = Z_R.length-1$
            return $result$;
        $top\text{-}R \leftarrow top\text{-}R+1$;
    if $Z_L[top\text{-}L].y \geq Z_R[top\text{-}R].y - d$ and $Z_L[top\text{-}L].y \leq Z_R[top\text{-}R].y + d$
        for $i$=0 to $min\{5, Z_R.length - top\text{-}R - 1\}$
            if $dist(Z_L[top\text{-}L], Z_R[top\text{-}R+i]) < d$
                $result=(Z_L[top\text{-}L], Z_R[top\text{-}R+i])$;
                $d=dist(Z_L[top\text{-}L], Z_R[top\text{-}R+i])$;
return $result$;
```

- ~~**Assume:**~~
  ~~Q中点已经分别按x坐标和y坐标排序后存储在X和Y中.~~

1. X=按x排序Q中点;
2. Y=按y排序Q中点;
3. FindCPP(X, Y).

时间复杂性=$O(n\log n)$+T(FindCPP)=$O(n\log n)$