# 划分树

# 参考

<http://blog.csdn.net/shiqi_614/article/details/8041390>

# 定义

划分树定义为，它的每一个节点保存区间[**left**,**right**]所有元素，元素顺序与原数组（输入）相同，但是，两个子树的元素为该节点所有元素排序后(rht-lft+1)/2个进入左子树，其余的到右子树，同时维护一个left\_child\_count，left\_child\_count [i]表示这个点之前（包括这个点）有多少个进入了左子树。

简单明了，如图所示：



Figure 1划分树



Figure 2利用划分数查找区间[4,7]上的第3个数

# 建树

## 时间复杂度

O(N\*logN)

## 算法

### 存储容器

const int MAX\_SIZE = 100001;

struct DataElement

{

int value[MAX\_SIZE];

int left\_child\_count[MAX\_SIZE];

};

DataElement tree[20]; // 为什么是20呢？如果题目中要求N<100000，那么划分树的层数为20就足够了，因为576 > 100000 > N。

int array[MAX\_SIZE];

存储容器有array和tree。

* array是用来对输入数据排序，在建树过程中会使用array获得平均值。
* tree是用来存储划分树的，该算法的灵魂。

### 输入数据

将输入的数据分别存储在array和tree[0]中。

### 对array进行排序

对array进行排序。

### 创建划分树

创建划分树：create\_tree(1,N);

create\_tree(left, right)

1. 选取第i层的left和right，计算mid=left+(right-left)/2
2. 遍历tree[i].value[N], j =1 🡪 N
   1. tree[i].left\_child\_count[i] = tree[i].left\_child\_count[i-1];
   2. if tree[i].value[j] <= array[mid]，则插入到左孩子，并且tree[i].left\_child\_count[i]++；否则插入到右孩子
3. 初始化第i+1层中的[left, mid]
4. 初始化第i+1层中的[mid+1, right]

## 实例

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **索引** |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 排序后的 |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 原数组 |  | 1 | 4 | 2 | 3 | 5 | 6 | 7 | 8 | 9 | 0 |
| 第0层 |  | 1 | mid=4 | 2 | 3 | 5 | 6 | 7 | 8 | 9 | 0 |
| 0 | 1 | 2 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 5 |
| 第1层 |  | 1 | 4 | mid=2 | 3 | 0 | 5 | 6 | mid=7 | 8 | 9 |
| 0 | 1 | 1 | 2 | 2 | 3 | 4 | 5 | 6 | 6 | 6 |
| 第2层 |  | mid=1 | 2 | 0 | 4 | mid=3 | 5 | mid=6 | 7 | mid=8 | 9 |
| 0 | 1 | 1 | 2 | 2 | 3 | 4 | 5 | 5 | 6 | 6 |
| 第3层 |  | 1 | mid=0 | 2 | 3 | 4 | mid=5 | 6 | 7 | 8 | 9 |
| 0 | 0 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 |
| 第4层 |  | 0 | 1 |  |  |  | 5 | 6 |  |  |  |
| 0 | 0 | 0 |  |  |  | 0 | 0 |  |  |  |

# 查找区间[from,to]内的第k个数

## 时间复杂度

O(logN)

## 算法

在原来的数组上查找区间[from, to]内的第k个数，就相当于在tree[0]上查找tree[0].value[from]到tree[0].value[to]内的第k个数，即query(from, to, k, 1, N, 0);

**query(from, to, k, left, right, h)**

1. 如果from == to，要找的就是tree[h].value[from]（query方法的出口）
2. 第h层tree，设from~to范围内分配到左孩子的节点数为**k1**

**k1** = tree[h].left\_child\_count[to] - tree[h].left\_child\_count[from - 1];

if **k1** < k

return query(next\_from, next\_to,**k-k1**,next\_left,next\_right,h+1)

else

return query(next\_from, next\_to,**k**,next\_left,next\_right,h+1)

为了简单明了，上式中的next\_from，next\_to，next\_left，next\_right在下面说明。

### 在左孩子中查找

**next\_from** = left + **[left,from)** 区间内分配到**左孩子**的个数

**next\_to** = next\_from + **[from,to]**中分配到**左孩子**的个数 – 1

= next\_from + **k1** – 1

**next\_left** =left

**next\_right** = mid

### 在右孩子中查找

**next\_from** = mid+1 + **[left, from)**区间内分配到**右孩子**的个数

**next\_to** = next\_from + **[from,to]**中分配到**右孩子**的个数 – 1

**next\_left** =mid + 1

**next\_right** = right

## 实例

### next\_from实例

* 情形1.1：

from=4，next\_from=from+(2)=6

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **索引** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **0层** | 1 | 4 | 2 | 3 | 5 | 6 | 7 | 8 | 9 | 0 |
| **1层** | 1 | 4 | 2 | 3 | 0 | 5 | 6 | 7 | 8 | 9 |

* 情形1.2：

from=4，next\_from=from+(2)=6

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **索引** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **0层** | 1 | 4 | 2 | 5 | 3 | 6 | 7 | 8 | 9 | 0 |
| **1层** | 1 | 4 | 2 | 3 | 0 | 5 | 6 | 7 | 8 | 9 |

* 情形2.1：

from=5，next\_from=from+(2)=6

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **索引** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **0层** | 1 | 4 | 2 | 3 | 5 | 6 | 7 | 8 | 9 | 0 |
| **1层** | 1 | 4 | 2 | 3 | 0 | 5 | 6 | 7 | 8 | 9 |

* 情形2.2：

from=5，next\_from=from+(1)=6

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **索引** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **0层** | 1 | 4 | 2 | 5 | 3 | 6 | 7 | 8 | 9 | 0 |
| **1层** | 1 | 4 | 2 | 3 | 0 | 5 | 6 | 7 | 8 | 9 |

* 情形3.1：

from=6， next\_from=from+(2)=7

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **索引** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **0层** | 1 | 4 | 2 | 3 | 5 | 6 | 7 | 8 | 9 | 0 |
| **1层** | 1 | 4 | 2 | 3 | 0 | 5 | 6 | 7 | 8 | 9 |

* 情形3.2：

from=6，next\_from=from+(2)=8

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **索引** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **0层** | 1 | 4 | 2 | 6 | 5 | 3 | 7 | 8 | 9 | 0 |
| **1层** | 1 | 4 | 2 | 3 | 0 | 6 | 5 | 7 | 8 | 9 |

* 情形4：

from=7，next\_from=from+(0)=7

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **索引** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| **0层** | 1 | 4 | 2 | 3 | 5 | 0 | 6 | 7 | 8 | 9 |
| **1层** | 1 | 4 | 2 | 3 | 0 | 5 | 6 | 7 | 8 | 9 |

### 例如：查找4~7中的第3个元素

from = 4

to = 7

k = 3

left = 1

right = 10

h = 0

query(4,7,3,1,10,0)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 索引 |  | left=1 | 2 | 3 | from=4 | 5 | 6 | to=7 | 8 | 9 | right=10 |
| 原数组 |  | 1 | 4 | 2 | 3 | 5 | 6 | 7 | 8 | 9 | 0 |
| 第0层 | **1--根节点--10** | | | | | | | | | | |
|  | 1 | 4 | 2 | 3 | 5 | 6 | 7 | 8 | 9 | 0 |
| 0 | 1 | 2 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 5 |

from = 6

to = 8

k = 2

left = 6

right = 10

h = 1

query(6,8,2,6,10,1)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 索引 |  |  |  |  |  | 5 | left=  from=6 | 7 | to=8 | 9 | right=  10 |
| 原数组 |  |  |  |  |  |  | 6 | 7 | 8 | 9 | 0 |
| 第1层 |  |  | | | | | **6--右孩子--10** | | | | |
|  |  |  |  |  |  | 5 | 6 | mid=7 | 8 | 9 |
|  |  |  |  |  | 3 | 4 | 5 | 6 | 6 | 6 |

from = 6

to = 8

k = 2

left = 6

right = 8

h = 2

query(6,8,2,6,8,2)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 索引 |  |  |  |  |  | 5 | left=  from=6 | 7 | right=  to=8 |  |  |
| 原数组 |  |  |  |  |  |  | 6 | 7 | 8 |  |  |
| 第2层 |  |  | | | | | **6—左孩子--8** | | |  | |
|  |  |  |  |  |  | 5 | mid=6 | 7 |  |  |
|  |  |  |  |  | 3 | 4 | 5 | 5 |  |  |

from = 6

to = 7

k = 2

left = 6

right = 7

h = 3

query(6,7,2,6,7,3)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 索引 |  |  |  |  |  | 5 | left=  from=6 | right=  to=7 |  |  |  |
| 原数组 |  |  |  |  |  |  | 6 | 7 |  |  |  |
| 第3层 |  |  | | | | | **6—左孩子--7** | |  |  | |
|  |  |  |  |  |  | mid=5 | 6 |  |  |  |
|  |  |  |  |  | 1 | 2 | 2 |  |  |  |

from = 7

to = 7

k = 1

left = 7

right = 7

h = 4

query(6,7,2,6,7,4)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 索引 |  |  |  |  |  |  | 6 | left=right= from=to=7 |  |  |  |
| 原数组 |  |  |  |  |  |  |  | 7 |  |  |  |
| 第4层 |  |  | | | | |  | **7—右孩子--7** |  |  | |
|  |  |  |  |  |  |  | 6 |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

总结如图：



# [示例源码](2-src/partition_tree.cc)

# 题目

## [hdoj 2665](../../7-hdoj/2-problems/1-2665/2665.docx)

## [poj 2104](../../6-poj/2-problems/1-2104/2104.docx)