平衡树

郭炼

哈尔滨工业大学
计算学部

2020 年 8 月 6 号

哈爾濱工業大學

目录

二叉搜索树定义

定义

二叉搜索树（Binary Search Tree，BST）是一种二叉树的树形数据结构，其定义如下：

1. 空树是二叉搜索树。
2. 若二叉搜索树的左子树不为空，则其左子树上所有点的附加权值均小于其根结点的值。
3. 若二叉搜索树的右子树不为空，则其右子树上所有点的附加权值均大于其根结点的值。
4. 二叉搜索树的左右子树均为二叉搜索树。

插入一个元素

我们可以根据当前 BST 的情况来执行插入操作：

- 若当前子树为空树，直接创建一个包含一个结点的 BST。
- 如果插入的权值等于子树根结点的权值，则退出插入。
- 如果插入的权值小于子树根结点的权值，则将权值插入根结点的左子树。
- 如果插入的权值大于子树根结点的权值，则将权值插入根结点的右子树。

查找一个元素

BST 的查找过程与二分查找类似，不过将每次取 mid 的过程修改为取子树的根结点

- 若当前子树为空树，返回 Null。
- 如果插入的权值等于子树根结点的权值，则该结点。
- 如果插入的权值小于子树根结点的权值，则在左子树进行查找。
- 如果插入的权值大于子树根结点的权值，则在右子树进行查找。

删除一个元素

删除元素时，我们需要首先找到该结点，之后根据树的形态来进行删除。

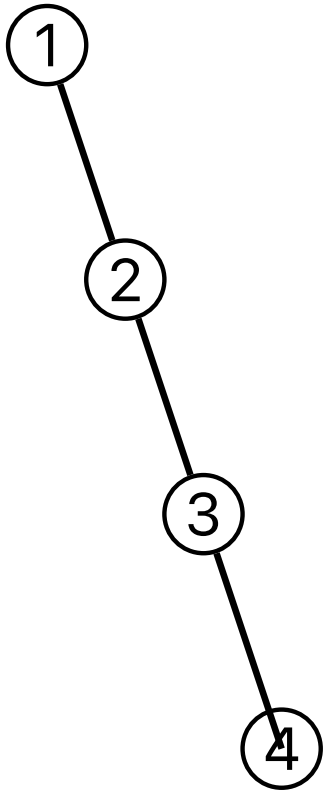
- 若该结点为叶子结点，直接删除该结点即可。
- 若该结点为链结点，即只有一个儿子的结点，返回这个儿子。
- 若该结点有两个非空子结点，一般是用它左子树的最大值或右子树的最小值代替它，然后将它删除。

缺点

- 朴素的 BST 直观且容易实现，对于随机的数据比较好。

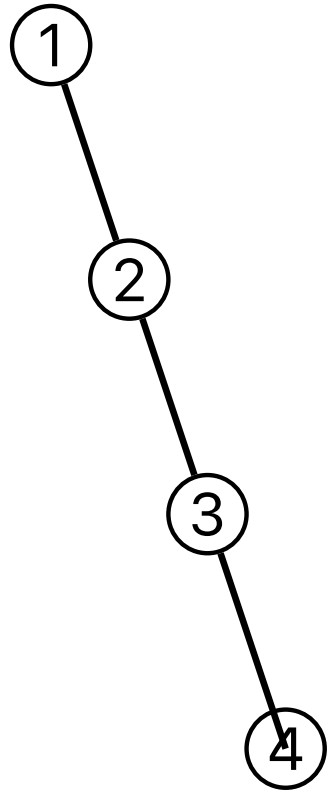
缺点

- 朴素的 BST 直观且容易实现，对于随机的数据比较好。
- 对于部分输入，BST 会不平衡。如依次插入一个有序序列，复杂度会达到 $O(n^2)$。



缺点

- 朴素的 BST 直观且容易实现，对于随机的数据比较好。
- 对于部分输入，BST 会不平衡。如依次插入一个有序序列，复杂度会达到 $O(n^2)$。
- 所以人们设计了一系列平衡树，如替罪羊树，Treap，Splay，AVL 树，红黑树等。



替罪羊树

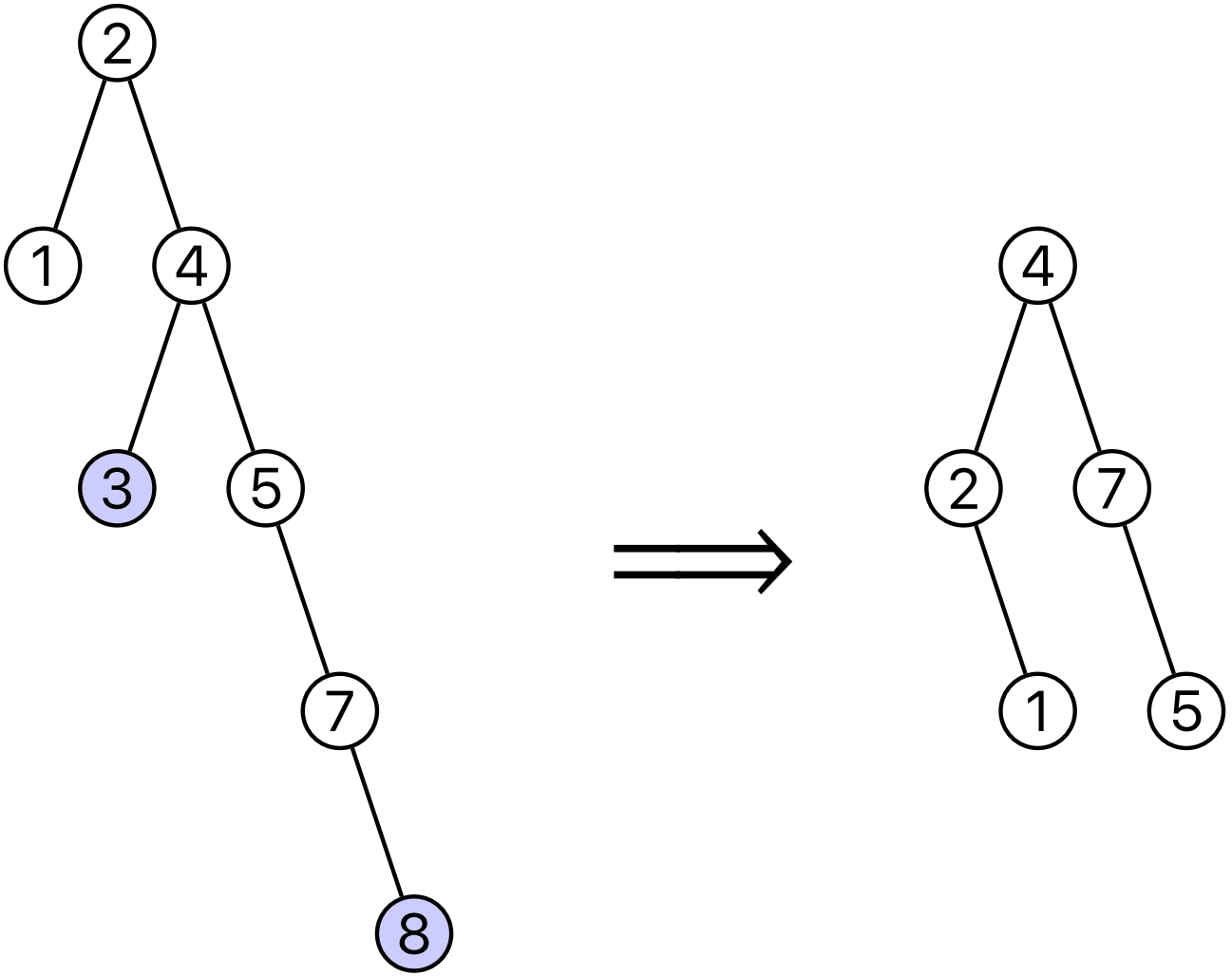
替罪羊树是一种依靠重构操作维持平衡的重量平衡树。替罪羊树会在插入、删除操作时，检测途经的结点，若发现失衡，则将以该结点为根的子树重构。

替罪羊树的重点是**重构**操作。

重构

- 我们引入一个平衡系数 $\alpha \in (0.5, 1)$，一般取 $[0.7, 0.8]$。若一个结点的某个子结点的大小占整棵子树大小的比例超过 α，则重构。
- 此外，替罪羊树的删除是惰性删除。如果有效结点占比低于 α，那么也要重构。
- 重构是将子树拍扁，之后按照二分的方法重新构建子树。

示例



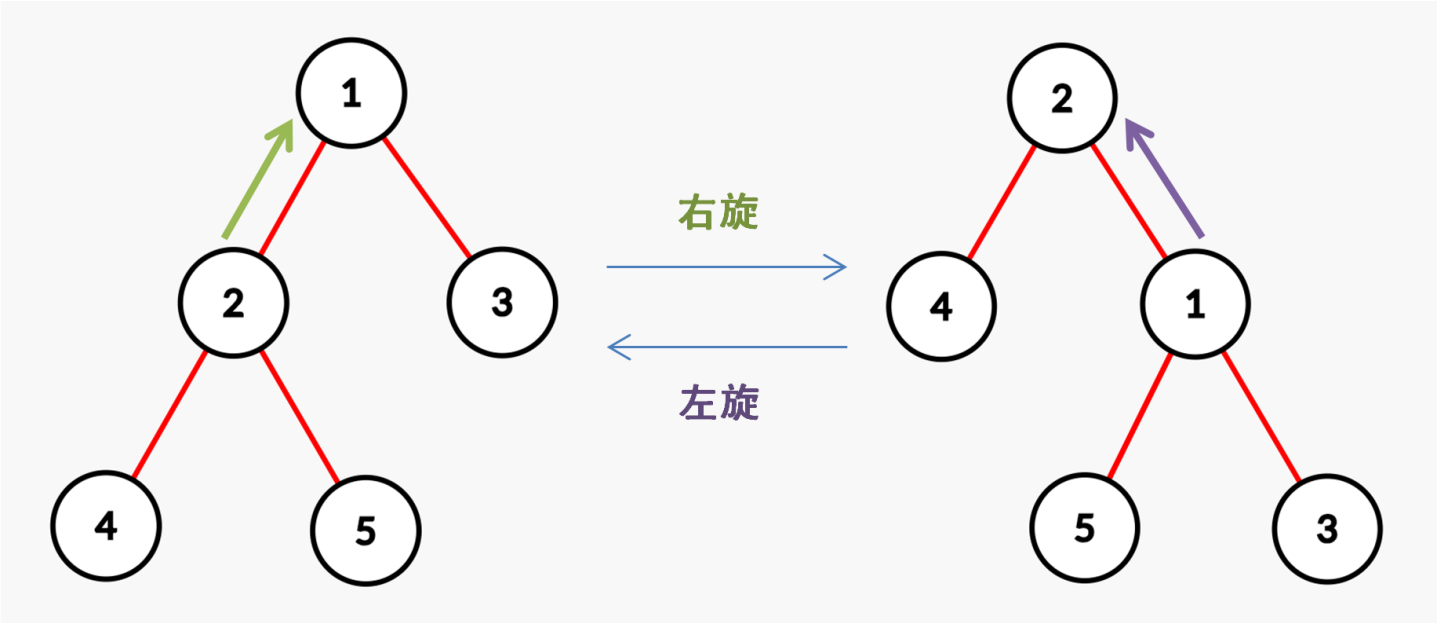
插入与删除

- 插入时与 BST 类似，但是需要判断是否需要重构。
- 替罪羊树的删除是惰性删除，不需要真实删除此点，而是将对应的点的标记从有效改为无效。同样需要判断重构。
- 由于重构，我们可以证明单个操作的均摊复杂度为 $O(\log n)$

旋转

除了重构，有其他方式来维持树的平衡吗？
将树旋转，让树从曲折变的平坦，从“高但贵”变成“矮却惠”。

旋转



旋转的思路修改子结点和其父结点的关系，进而改变树的形态。

代码

```
void rotate(Node *u) {
  Node *f = u->fa, *ff = f->fa;
  int d = u == f->ch[1];
  push_down(f), push_down(u);
  if ((f->ch[d] = u->ch[d ^ 1]) != nil) f->ch[d]->fa = f;
  if ((u->fa = ff) != nil) ff->ch[f == ff->ch[1]] = u;
  f->fa = u;
  u->ch[d ^ 1] = f;
  maintain(f), maintain(u);
}
```

Treap

- Treap 是一种弱平衡的二叉搜索树。
- Treap = Tree + Heap，是一种由树和堆组合形成的数据结构。
- 在 Treap 中，每个结点都额外存储了一个属性：priority。
- Treap 除了要满足二叉搜索树的性质之外，还需满足父节点的 priority 大于等于两个儿子的 priority。
- 而 priority 是每个结点建立时随机生成的，因此 Treap 是期望平衡的。
- Treap 分为两种：无旋式 Treap 和旋转 Treap。

旋转 Treap 的插入与删除

- 向 Treap 首先按照 BST 的方法插入一个结点，之后随机产生一个 priority 值。如果新结点的 priority 值大于父结点，就向上旋转，直到不满足条件或者到根。
- 删除则与 BST 无异。

局部性原则

局部性原则

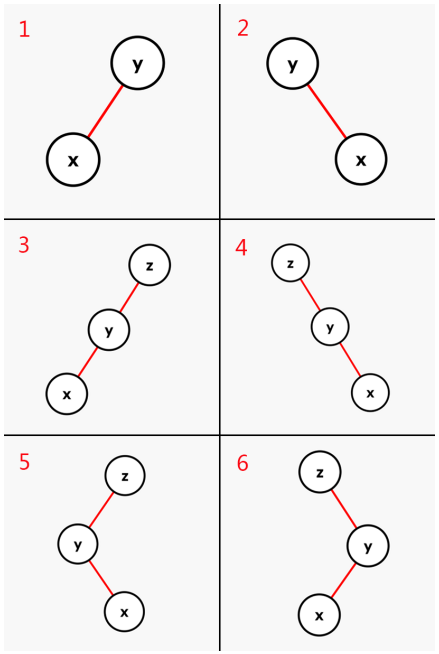
刚刚被访问的元素，下一次有更大的概率再次被访问。在一个短暂时间内，被连续查询的一系列元素是接近的。

基于上述思想，我们可以在每次访问/操作树之后，将访问/操作到的结点旋转到整棵平衡树的根。这就引出了 Splay 树。

Zig Zag

- 对于 Splay 树来说，重点是 Splay 操作，即将结点旋转到树的特定位置。
- 为了保证复杂度，我们需要在每次操作后将最后访问到的那个点旋转到根的位置，将旋转进行更细致的讨论。

Zig Zag



- 如果 x 的父亲是根节点，直接将 x 左旋或右旋（图 $1, 2$）。
- 如果 x 的父亲不是根节点，且 x 和父亲的儿子类型相同，首先将其父亲左旋或右旋，然后将 x 右旋或左旋（图 $3, 4$）。
- 如果 x 的父亲不是根节点，且 x 和父亲的儿子类型不同，将 x 左旋再右旋、或者右旋再左旋（图 $5, 6$）。
- 复杂度证明见Wikipedia。

代码

```
void splay(Node *u, Node *target) {
  for (Node *f; u->fa != target; rot(u))
  if ((f = u->fa)->fa != target) {
    ((u == f->ch[1]) ^ (f == f->fa->ch[1])) ? rotate(u) :
        rotate(f);
  }
  if (target == nil) root = u;
}
```

Sequence I

Sequence I

给定一个下标为 $1,\ldots,n$ 的序列，有如下两种操作：

1. 修改 a_i 为 x。
2. 寻问序列区间 $[l,r]$ 内的最大值。

$n \le 10^5$，操作数 $\le 10^5$，元素值 $\le 10^9$。

题解

- 根据序列的下标建立一棵平衡树。

题解

- 根据序列的下标建立一棵平衡树。
- 每个结点额外维护一个值：子树中的最大值。

题解

- 根据序列的下标建立一棵平衡树。
- 每个结点额外维护一个值：子树中的最大值。
- 修改就将对应的结点旋转到根。

题解

- 根据序列的下标建立一棵平衡树。
- 每个结点额外维护一个值：子树中的最大值。
- 修改就将对应的结点旋转到根。
- 查询则将取出对应的区间：
  - 添加两个“哨兵”结点对应 $-\infty, +\infty$，
  - 将 $l-1$ 对应的结点旋转到根，
  - 将 $r+1$ 对应的结点旋转为 $l-1$ 的子节点，
  - 那么区间 $[l, r]$ 对应的结点为 $r+1$ 的右结点。

Sequence II

Sequence II

给定一个下标为 $1,\ldots,n$ 的序列，有如下两种操作：

1. 将 $a_l,\ldots,a_r$ 增加 x。
2. 寻问序列区间 $[l,r]$ 内的最大值。

$n \le 10^5$，操作数 $\le 10^5$，元素值 $\le 10^9$。

题解

- 此题大部分与上一题相同。

题解

- 此题大部分与上一题相同。
- 对于区间修改，我们可以类似线段树，使用惰性标记。

Sequence III

Sequence III

给定一个下标为 $1,\ldots,n$ 的序列，有如下两种操作：

1. 将区间 $[l,r]$ 的值修改为 x
2. 给区间 $[l,r]$ 的值增加为 x
3. 给区间 $[l,r]$ 的值乘上为 x
4. 寻问序列区间 $[l,r]$ 内的最大值。

$n \le 10^5$，操作数 $\le 10^5$，元素值 $\le 10^9$。

题解

- 此题中重点如何设计是惰性标记。

题解

- 此题中重点如何设计是惰性标记。
- 设 $lazy\_tag = k \times a_i + b$。

题解

- 此题中重点如何设计是惰性标记。
- 设 $lazy\_tag = k \times a_i + b$。
- 对于修改，置 $lazy\_tag = 0 \times a_i + x$。
- 对于增加，置 $lazy\_tag = 1 \times a_i + x$。
- 对于乘法，置 $lazy\_tag = x \times a_i + 0$。

题解

- 此题中重点如何设计是惰性标记。
- 设 $lazy\_tag = k \times a_i + b$。
- 对于修改，置 $lazy\_tag = 0 \times a_i + x$。
- 对于增加，置 $lazy\_tag = 1 \times a_i + x$。
- 对于乘法，置 $lazy\_tag = x \times a_i + 0$。
- 注意惰性标签的合并。

Sequence IV

Sequence IV

有一个变长数组，初始为空，有以下几种操作：

- 删除数组第 i 个元素。
- 修改下标为 i 的值为 x。
- 在数组下标为第 i 个元素后插入一个元素。
- 查询下标区间 $[l, r]$ 内的最大值。

操作数 $\leq 10^5$，元素值 $\leq 10^9$.

题解

- 增加删除和添加操作即可。

Sequence V

Sequence V

有一个变长数组，初始为空，有以下几种操作：

- 在数组下标为第 i 个元素后插入一个元素。
- 删除数组第 i 个元素。
- 修改下标为 i 的值为 x。
- 翻转数组下标区间 $[l, r]$ 的所有元素。
- 查询下标区间 $[l, r]$ 内的最大值。

操作数 $\leq 10^5$，元素值 $\leq 10^9$。

题解

- 对于翻转操作，我们只有交换两个子结点即可。

题解

- 对于翻转操作，我们只有交换两个子结点即可。
- 添加一个惰性标记来标记是否需要翻转。

Shaolin

HDU 4585 Shaolin

初始有一个编号为 1 能力值为 10^9 的人在队列中，给你新人的数目 n 和 n 个新人的编号和能力值。每来一个新的人，他就会在比他先来的所有人中挑一个和他能力值相差最少的人进行比试，然后输出比试的名单。

$n \le 10^5, 0 \le$ 编号和能力值 $\le 5^5$。

题解

- 每次查找元素的前驱和后继。

神秘物质

BJWC 2017 神秘物质

有个序列，一开始有 n 个元素 $\{E_i\}$，你需要维护以下几种操作：

- insert x e：在第 x 个数后面插入一个 e。
- merge x e：删掉第 $x+1$ 个数并将第 x 个数改为 e。
- max x y：询问当前区间 $[x, y]$ 内任意长度大于 1 的子区间中极差的最大值。
- min x y：询问当前区间 $[x, y]$ 内任意长度大于 1 的子区间中极差的最小值。

操作数为 q，$1 \le n, q \le 10^5$，$1 \le e, E_i \le 10^9$。

题解

- 极差的最大值就是区间最大值减去最小值。
- 极差的最小值就是区间必定出现在两个相邻元素之间。