平衡树

12 天前 笔记 1513 Words 被浏览 26 次

- 我自己的复习就开始从平衡树开始吧qwq
- 这是我最早咕的一个知识点，现在来看，我那时菜得竟然连$Treap$都看不懂，$emming$

$Treap$

- 特点：
  - 优点：
    1. 最常用最好写的平衡树，在考场上如果可以实现，$Treap$打起来是最快的。
    2. 不是很慢，甚至可以说挺快的。
  - 缺点：
    1. 通用性和扩展性不如$Splay$
- 算法：简而言之就是一个二叉搜索树(BST)+一个堆。~~众所周知~~，堆是一个完全二叉树，所以我们只要在一个BST中的每一个结点创造出一个附属值（在堆中的优先级），然后再用堆化为一个接近完全二叉树的BST（还要满足BST的性质下）。

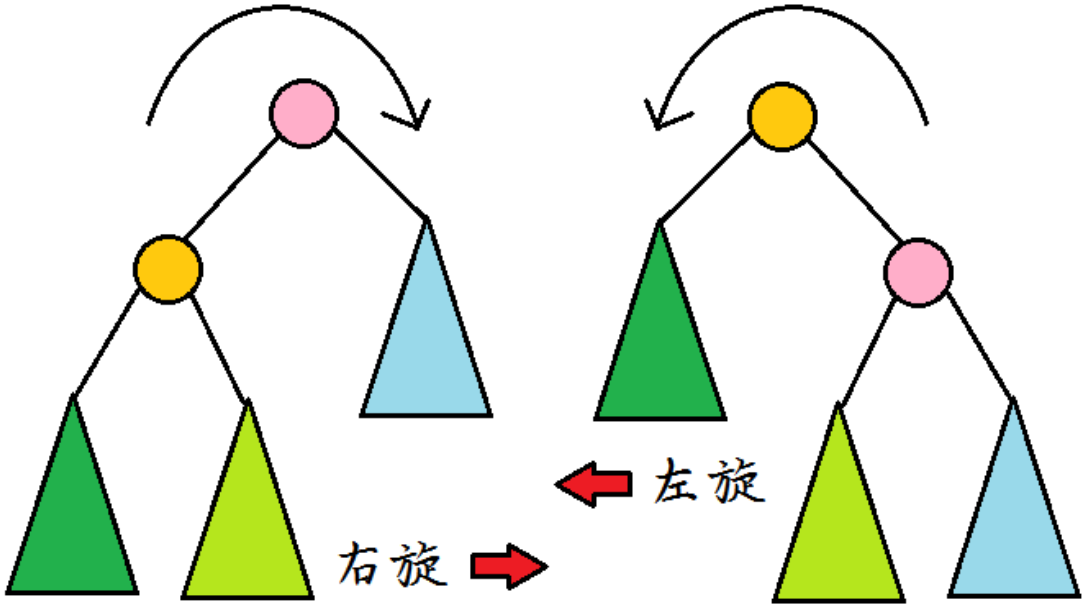
目录

食之无味 弃之可惜
The Simple World

文章	分类	标签
68	8	44

关注我

1. 如果要插入一个数，用 `rand` 进行优先级赋值。然后加入到树中去（BST的加入方法，不是堆的加入方法，否则不满足BST）。
2. 然后就是一个堆的上浮过程了，根据实际情况进行相连的两个节点的左旋和右旋。



旋转

如果看不懂，就再看看下面的代码帮助理解。其实就是在旋转中交换儿子，左旋和右旋的交换都是固定的。不需要考虑多种情况。

3. 再然后就没有什么了，更多的功能按照自己需要的加。

    1. 每个树节点几个属性（或者说你要开几个不同的数组），`l[]` 左二子，`r[]` 右儿子，`sum[]` 这个数有几个（因为你可能会插入多个一样的数），`rank[]` 优先值，`val[]` 这个节点的值，

链接

Todo List

摘抄本

Whisper 细语

分享站

图床

控制台

`size` 以这个节点为根的子树有多少个结点。

- 例题：普通平衡树
  - 模板题，直接打即可，插入和删除都和线段树比较类似。

```
#include <iostream>
#include <cstdio>
#include <cstdlib>

using namespace std;
const int MAXN=100005;
int n;
struct treap{
    int l[MAXN],r[MAXN],val[MAXN],rank[MAXN],size[MAXN],sum[MAXN];
    int sz,ans,rt;
    inline void pushup(int x){
        size[x]=size[l[x]]+size[r[x]]+sum[x];
    }
    inline void lrotate(int &k){
        int t=r[k];
        r[k]=l[t];
        l[t]=k;
        size[t]=size[k];
        pushup(k);
        k=t;
    }
    inline void rrotate(int &k){
        int t=l[k];
        l[k]=r[t];
        r[t]=k;
        size[t]=size[k];
        pushup(k);
        k=t;
```

```
29      }
30      void insert (int &k,int x){
31          if (!k){
32              sz++;
33              k=sz;
34              size[k]=1;
35              sum[k]=1;
36              val[k]=x;
37              rank[k]=rand();
38              return;
39          }
40          size[k]++;
41          if (val[k]==x){
42              sum[k]++;
43          }
44          else if (val[k]<x){
45              insert (r[k],x);
46              if (rank[r[k]]<rank[k])
47                  lrotate(k);
48          }
49          else {
50              insert (l[k],x);
51              if (rank[l[k]]<rank[k])
52                  rrotate(k);
53          }
54      }
55      void del(int &k,int x){
56          if (!k) return;
57          if (val[k]==x){
58              if (sum[k]>1){
59                  sum[k]--;
60                  size[k]--;
61                  return;
62              }
```

```
63              if (l[k]==0||r[k]==0){
64                  k=l[k]+r[k];
65              }
66              else if (rank[l[k]]<rank[r[k]]){
67                  rrotate(k);
68                  del(k,x);
69              }
70              else{
71                  lrotate(k);
72                  del(k,x);
73              }
74          }
75          else if (val[k]<x){
76              size[k]--;
77              del(r[k],x);
78          }
79          else {
80              size[k]--;
81              del(l[k],x);
82          }
83      }
84      int queryrank(int k,int x){
85          if (!k) return 0;
86          if (val[k]==x){
87              return size[l[k]]+1;
88          }
89          else if (x>val[k]){
90              return size[l[k]]+sum[k]+queryrank(r[k],x);
91          }
92          else return queryrank(l[k],x);
93      }
94      int querynum(int k,int x){
95          if(!k) return 0;
96          if(x<=size[l[k]]) return querynum(l[k],x);
```

```
97          else if(x>size[l[k]]+sum[k])
98              return querynum(r[k],x-size[l[k]]-sum[k]);
99          else return val[k];
100     }
101     void querypre(int k,int x){
102         if(!k) return ;
103         if(val[k]<x) ans=k,querypre(r[k],x);
104         else querypre(l[k],x);
105     }
106     void querysub(int k,int x){
107         if(!k) return;
108         if(val[k]>x) ans=k,querysub(l[k],x);
109         else querysub(r[k],x);
110     }
111 }T;
112 int main(){
113     scanf("%d",&n);
114     int opt,x;
115     for(int i=1;i<=n;i++){
116         scanf("%d%d",&opt,&x);
117         if(opt==1)T.insert(T.rt,x);
118         else if(opt==2)T.del(T.rt,x);
119         else if(opt==3){
120             printf("%d\n",T.queryrank(T.rt,x));
121         }
122         else if(opt==4){
123             printf("%d\n",T.querynum(T.rt,x));
124         }
125         else if(opt==5){
126             T.ans=0;
127             T.querypre(T.rt,x);
128             printf("%d\n",T.val[T.ans]);
129         }
130         else if(opt==6){
```

```
131                 T.ans=0;
132                 T.querysub(T.rt,x);
133                 printf("%d\n",T.val[T.ans]);
134             }
135         }
136         return 0;
137     }
```

- 优化： `<cstdlib>` 的 `rand()` 有点慢，所以可以直接手写一个：

```
1   inline int rand ( )  {
2       static int seed = 233;//seed可以随便取
3       return seed = ( int ) seed * 482711LL % 2147483647;
4   }
```

非旋$Treap$

- 这个一般来说有点慢，但是可以让树可持久化。
- 可持久化就是：

> **可持久化数据结构(Persistent data structure)**是一种在发生改变时，会保存之前的版本的数据结构。这是一种不可变的$(immutable)$数据结构，对数据进行操作时，不会在原数据上进行更新改变，而是会生成另一个新的发生改变了的新数据。

所以这不应该很耗空间吗qwq？？

- 不常用，暂时不学。

$Splay$

- 这个 $Splay$ 相比 $Treap$ 来说更全能一点，除了可能代码会~~又臭又长~~（好吧，其实并没有）。
- $Splay$ 感觉更加优雅一些，因为用其他冗余的东西来平衡，只是单纯的左旋和右旋。
- 优点：和 $Treap$ 差不多，比 $Treap$ 扩展性更高
- 缺点：比 $Treap$ 打起来慢一点，但是一般平衡树的题直接用它就可以了，万一 $Treap$ 写到一半发现只能用 $Splay$ 就比较尴尬。

基本操作

- 更新子树结点数量 `size` ：

```
inline void UDsize(int x){
    size[x]=size[son[x][0]]+size[son[x][1]]+cnt[x];
    return;
}
```

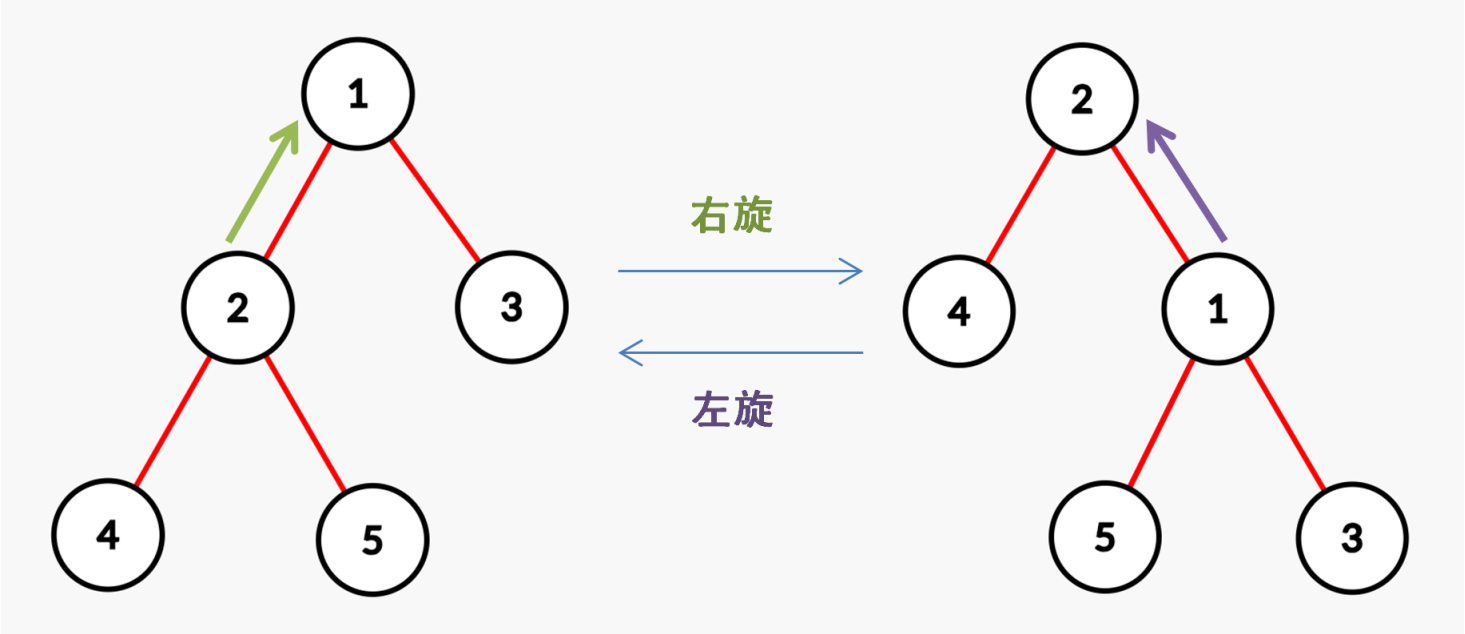
- 左儿子 or 右儿子：

```
inline bool lrson(int x){
    return x==son[fa[x]][1];
}
```

- 删除结点：

```
inline void clear(int x)[
    size[x]=son[x][0]=son[x][1]=cnt[x]=val[x]=fa[x]=0;
    return;
]
```

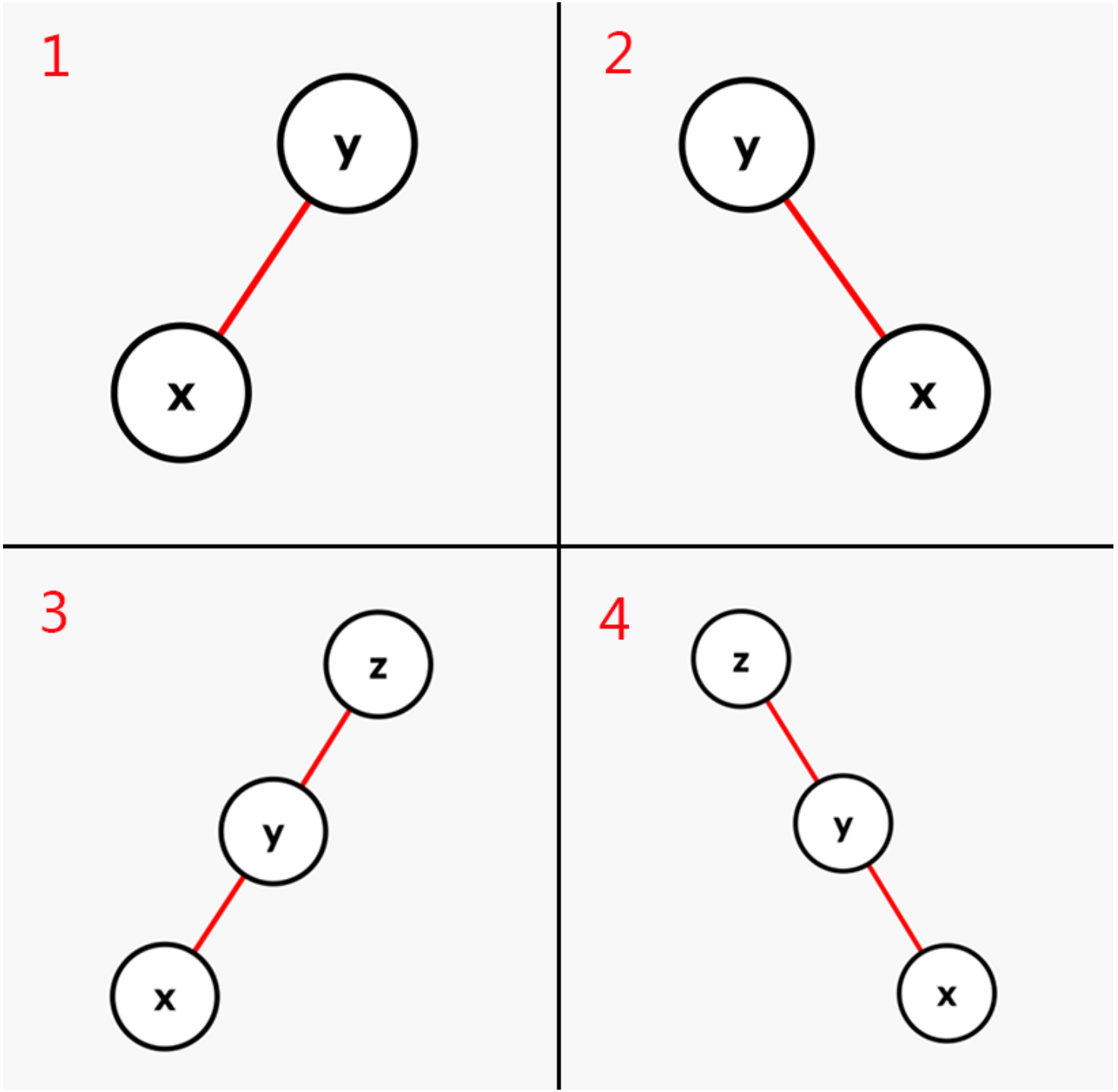
- 旋转~~（竟然不用分左右）~~：用手模拟一遍就可以知道怎么旋转了

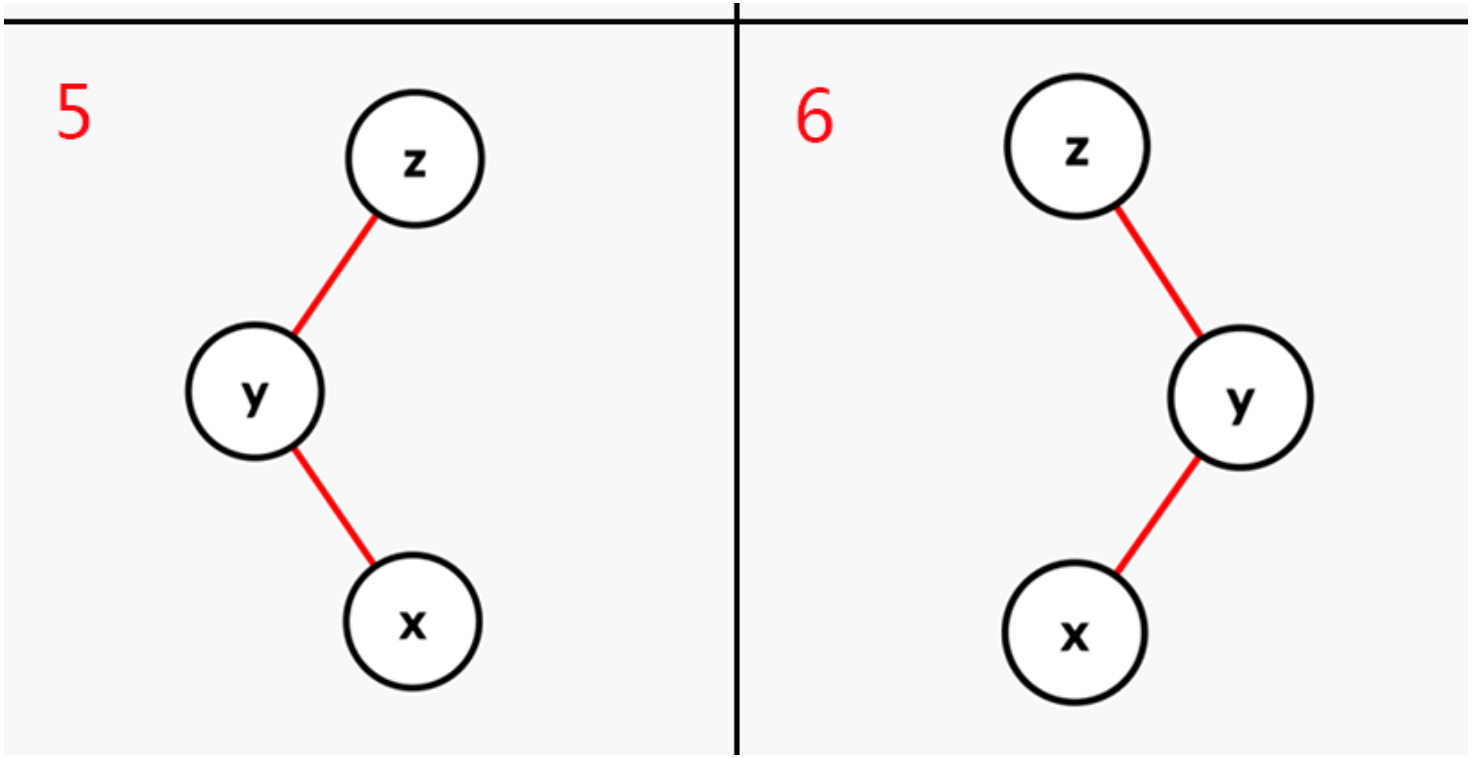


Splay旋转

```
inline void rotate(int x) {
    int y = fa[x], z = fa[y], sonk = lrson(x);
    son[y][sonk] = son[x][sonk ^ 1];
    fa[ch[x][sonk ^ 1]] = y;
    son[x][sonk ^ 1] = y;
    fa[y] = x;
    fa[x] = z;
    if (z) son[z][y == son[z][1]] = x;
    UPsize(y);
    UPsize(x);
    return;
}
```

- *Splay* 伸展：直接引用 OI-Wiki 的例子
  - 一共有6中情况：


1
y
x
2
y
x
3
z
y
x
4
z
y
x



Splay伸展

> 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 ）

```
1   inline void splay(int x) {
2       for (int f = fa[x]; f = fa[x], f; rotate(x))
3           if (fa[f]) rotate(lrson(x) == lrson(f) ? f : x);
4       rt = x;
```

```
5       return;
6   }
```

#平衡树

加密 Hexo 博文 >

昵称 邮箱(用于显示 Gravater 头像及邮箱提醒) 网址(http://)

✧⁺□(●˙▾˙●)□□□来撩我啊~~

表情 | 预览

回复

快来做第一个评论的人吧~

43.8k Words | 7336 Views | 2298 Visitors

Powered by Hexo & Icarus

第 339 条一言：「日子过的象流水一般。它静静的从我们身边缓缓流过，不带半分声响。那些我们当年执着的人，执着的事，执着之后，却变成一种负担。」