Red-black Tree

何钦铭

April 13,2022

1 Chapter 1: Introduction

1.1 Background

红黑树是用来高效存储和查询数据的数据结构，相比普通的树有以下五点属性:

- 每个节点不是红色就是黑色。
- 根是黑色的。
- 所有的叶子都是空节点，颜色都是黑色。
- 每个红色节点必须有两个黑色子节点（可能为空)。
- 从任何节点 x 到后代叶的所有简单路径都有相同数量的黑色节点

1.2 Tasks specification

在本题中需要求出有 N 个内部节点的红黑树有多少种，并将结果对 1000000007 取模后输出。

2 Algorithm Specification

我们使用动态规划算法来求解这道题。

2.1 数据存储

首先我们使用两个二维数组储存 dp 信息，分别代表根为红色的红根红黑树和根为黑色的黑根红黑树。数组第一个下标代表内部节点个数，第二个下标代表 black-height。

```
long long Red[1000][1000];
long long Black[1000][1000];
```

2.2 递归基底

通过观察可知，内部节点为 1，black-height 为 1 的黑根红黑树个数为 1，内部节点为 2，black-height 为 1 的黑根红黑树个数为 2，内部节点为 3，black-height 为 1 的黑根红黑树个数为 1。对数组赋初始值。

```
Black[1][1] = 1;
Black[2][1] = 2;
Black[3][1] = 1;
```

2.3 递推式

需要求出不同项数之间的递推关系。对于黑根树来说，它的两棵子的根颜色可以任意，只要保证它的黑色高度是子树的黑色高度加一。对于红根树来说，它的两棵子树都是黑根树，并且它的黑色高度和子树相同。由此可得递推关系。并在递推式的基础上遍历数组进行计算。

```
Black[i][k] = (Black[i][k] + (Red[j][k - 1] + Black[j][k - 1])
* (Red[i - j - 1][k - 1] + Black[i - j - 1][k - 1]) ) % mod;
```

```
Red[i][k] = (Red[i][k]+ Black[j][k] * Black[i - j - 1][k] ) % mod;
//i为内部节点个数，j为左子树的内部节点个数，k为black-height
```

2.4 得出结果

统计符合条件的有 N 个节点的黑根红黑树的数量，只需要统计可能 black-height 的子树个数即可，最后将结果取模并输出。由于使用 int 可能

造成数字超限，所以使用 long long 类型的整数表示。

```
long long sum = 0;
for (int i = log2(N + 1)/2; i <= log2(N + 1); i++)
{
    sum = (sum + Black[N][i]) % mod;
}
printf("%lld", sum);
```

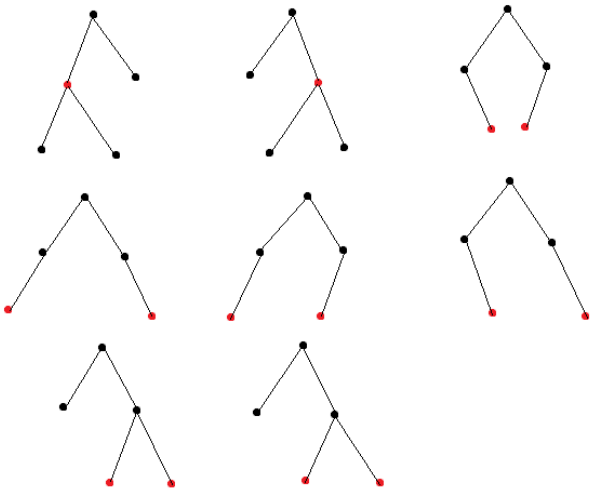
2.5 Main data structures

本题主要数据结构为二维数组，因为 dp 问题储存数据的需要，使用二维数组更能满足解题要求。

3 Chapter 3: Testing results

testing reason	input	output	correct output
最小值	1	1	1
最大值	500	905984258	905984258
一般值 1	5	8	8
一般值 2	3	2	2

经检验可知结果正确



提交时间	状态	分数	题目	编译器	耗时	用户
2022/04/15 09:10:14	答案正确	35	编程题	C++ (g++)	11 ms	

测试点	结果	分数	耗时	内存
0	答案正确	18	4 ms	460 KB
1	答案正确	7	5 ms	592 KB
2	答案正确	7	5 ms	456 KB
3	答案正确	2	11 ms	456 KB
4	答案正确	1	3 ms	328 KB

4 Chapter 4: Complexity Analysis

4.1 空间复杂度:

程序中使用了一个二维数组，第一个坐标是点的个数，上界为 N, 第二个坐标是 black-height 的大小，上界为 $log2(N+1)$，所以空间复杂度为 $Nlog(N)$.

4.2 时间复杂度:

时间复杂度考虑三重循环即可。该循环花费时间为 $O(log2(1)*1+log2(2)*2+...+log2(N)*N)$，具体的求和难度较大，但可以判断在 $O(n^2)$ 到 $O(n^3)$ 之间。

5 代码附录

```
#include <stdio.h>
#include <math.h>
#define mod 1000000007
//第一个下标代表内部节点个数，第二个下标代表black-height
long long Red[1000][1000];//代表根为红色的红根红黑树
long long Black[1000][1000]; //代表根为黑色的黑根红黑树
int main()
{
    int N;//内部节点的个数
    scanf("%d",&N);
    //定义初始情形
    Black[1][1] = 1;
    //内部节点为1，black-height为1的黑根红黑树个数为1
    Black[2][1] = 2;
    //内部节点为2，black-height为1的黑根红黑树个数为2
    Black[3][1] = 1;
    //内部节点为3，black-height为1的黑根红黑树个数为1
    //因为最终的红黑树的根为黑色，所以就不定义红根红黑树的初始情形
```

```
    for (int i = 3; i <= N; i++)
    {
        //j为左子树的内部节点个数
        for (int j = 1; j < i - 1; j++)
        {
            //k为black-height
            for (int k = log2(i+1)/2; k <= log2(i+1); k++)
            {
                //对于黑根树来说，它的两棵子的根颜色可以任意，只
                要保证它的黑色高度是子树的黑色高度加一
                Black[i][k] = (Black[i][k] + (Red[j][k - 1] +
                Black[j][k - 1]) * (Red[i - j - 1]
                [k - 1] + Black[i - j - 1][k - 1]) ) % mod;//
                //对于红根树来说，它的两棵子树都是黑根树，并且它
                的黑色高度和子树相同
                Red[i][k] = (Red[i][k]+ Black[j][k] * Black[i - j
                - 1][k] ) % mod;
            }
        }
    }
    long long sum = 0;
    //统计内部节点为n的子树个数
    for (int i = log2(N + 1)/2; i <= log2(N + 1); i++)
    //只需要统计可能black-height的子树个数即可
    {
        sum = (sum + Black[N][i]) % mod;
    }
    printf("%lld", sum);
    return 0;

}
```