Red-Black Tree

Yunce Zhang,

Haoxiang Huang,

Yuwei Cao

Date: 2022-04-22

Chapter 1: Introduction

In the project, it’s required to calculate how many distinct Red-Black Trees consists of exactly pre-given N internal nodes.

Chapter 2: Data Structure / Algorithm Specification

In the chapter, we will briefly discuss the Red-Black Tree.

2.1 Red-Black Tree

Red-Black Tree (RBT) is a kind of self-balancing binary search tree which satisfies the following requirements:

1) Every node is either red or black.
2) The root is black.
3) All the leaves are NULL nodes and are colored black.
4) Each red node must have 2 black descends (may be NULL).
5) All simple paths from any node x to a descendant leaf have the same number of black nodes.

In 1972, Rudolf Bayer invented a data structure that was a special order-4 case of a B-tree. These trees maintained all paths from root to leaf with the same number of nodes, creating perfectly balanced trees. However, they were not binary search trees. Bayer called them a "symmetric binary B-tree" in his paper and later they became popular as 2–3–4 trees or just 2–4 trees.

In a 1978 paper, "A Dichromatic Framework for Balanced Trees", Leonidas J. Guibas and Robert Sedgewick derived the red–black tree from the symmetric binary B-tree. The color "red" was chosen because it was the best-looking color produced by the color laser printer available to the authors while working at Xerox PARC. Another response from Guibas states that it was because of the red and black pens available to them to draw the trees.

In 1993, Arne Andersson introduced the idea of a right leaning tree to simplify insert and delete operations.

In 1999, Chris Okasaki showed how to make the insert operation purely functional. Its balance function needed to take care of only 4 unbalanced cases and one default balanced case.

The original algorithm used 8 unbalanced cases, but Cormen et al. (2001) reduced that to 6 unbalanced cases. Sedgewick showed that the insert operation can be implemented in just 46 lines of Java code.

In 2008, Sedgewick proposed the left-leaning red–black tree, leveraging Andersson's idea that simplified the insert and delete operations. Sedgewick originally allowed nodes whose two children are red, making his trees more like 2–3–4 trees, but later this restriction was added, making new trees more like 2–3 trees. Sedgewick implemented the insert algorithm in just 33 lines, significantly shortening his original

46 lines of code.

2.2 Algorithm Specification

We use a naïve algorithm to calculate the answer. We define two arrays redNode[i][j] and blackNode[i][j] which represents the number of possible trees when we choose red node or black node as the root node while i is the black height and j is the number of internal nodes.

We can know the all possible situations using recursion:

1) When the root node has null child-node,

   redNode[0][1] = 1, blacNode[1][1]=1, blackNode[1][2]=2.

2) A red node has exactly two black child.

   redNode[i][j+1] += blackNode[i][j-k]*blackNode[i][k];

3) A black node's child is arbitrary.

   blackNode[i+1][j+1] += (redNode[i][j-k]+blackNode[i][j-k])*

   (redNode[i][k]+blackNode[i][k])

Pseudo-code

```
    blackNode[1][1] = 1;
    blackNode[1][2] = 2;
    redNode[0][1] = 1;
    for(i = 0; i <= 10; i++)// The problem defines a mod num 1000000007
    {
        for(j = 2; j < n; j++)
        {
            for(k = 1; k < j; k++)
            {
                redNode[i][j+1] +=
blackNode[i][j-k]*blackNode[i][k];
                blackNode[i+1][j+1] +=
(redNode[i][j-k]+blackNode[i][j-k])*(redNode[i][k]+blackNode[i][k
])%modNum;
                redNode[i][j+1] %= modNum;
                blackNode[i+1][j+1] %= modNum;
            }
        }
    }
    for(i = 0; i <= n; ++i)
    {
        resNum = (resNum+blackNode[i][n])%modNum;
    }
    cout << resNum;
```

Chapter 3: Testing Results

3.1 Test Environment

Our test environment is just as following:

OS: Windows 11 Pro for Workstations, Build 22598.200

Compiler: gcc 8.1.0, x86_64-posix-seh-rev0

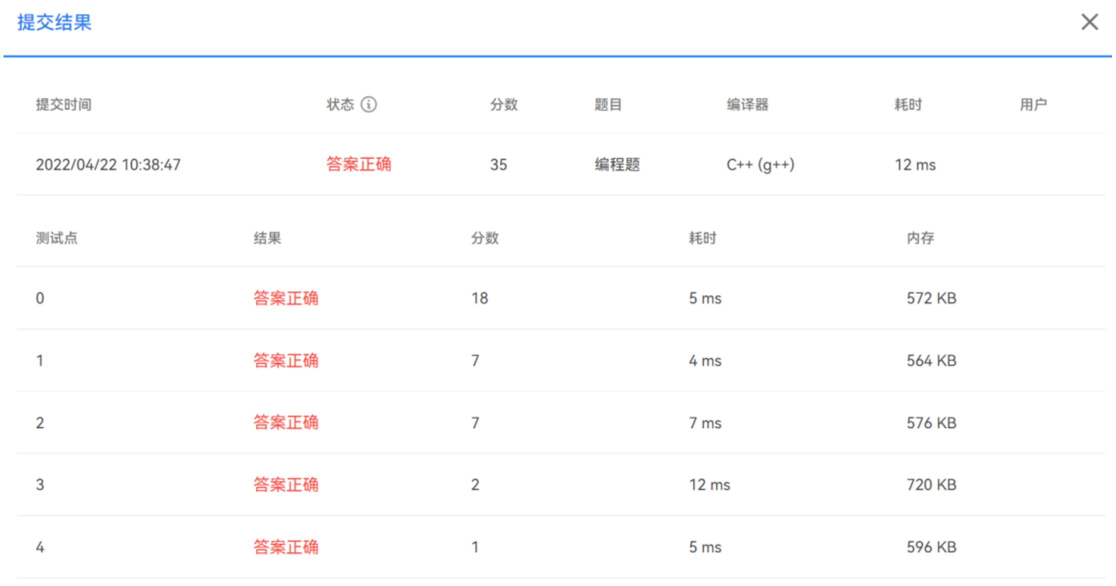
Compile command: g++ -Wall -std=c++17 source.cpp -o source -O2

CPU: AMD EPYC-7542

RAM: 256GiB

3.2 Test Results and Analysis

The project problem is the same as Pintia TOP 1007 Red-Black Tree. We pass all the test points.



提交结果

提交时间	状态	分数	题目	编译器	耗时	用户
2022/04/22 10:38:47	答案正确	35	编程题	C++ (g++)	12 ms	

测试点	结果	分数	耗时	内存
0	答案正确	18	5 ms	572 KB
1	答案正确	7	4 ms	564 KB
2	答案正确	7	7 ms	576 KB
3	答案正确	2	12 ms	720 KB
4	答案正确	1	5 ms	596 KB

We test three cases. The 1st, 2nd and 3rd cases.

Format description of input data:

We input the given number N which defines the number of internal nodes.

1st input: 3

Output: 2

2nd input: 5

Output: 8

3rd input: 8

Output: 56

Those three results all agree with our expectations. Hence the algorithm is correctly implemented.

We want to use these three cases to evaluate the algorithm complexity.

N	1	2	3	4
Time/s	2.3159e-08	2.4629e-08	2.28823e-08	6.22272e-07
N	5	6	8	10
Times/s	1.258444e-06	2.072292e-06	4.336219e-06	7.368353e-06
N	15	20	40	80
Time/s	1.86423e-05	3.5167e-05	1.53202e-04	6.50253e-04
N	100	200	400	500
Time/s	1.02412e-03	4.19709e-03	1.70507e-02	2.68083e-02
N	1000			
Time/s	1.21369e-01			

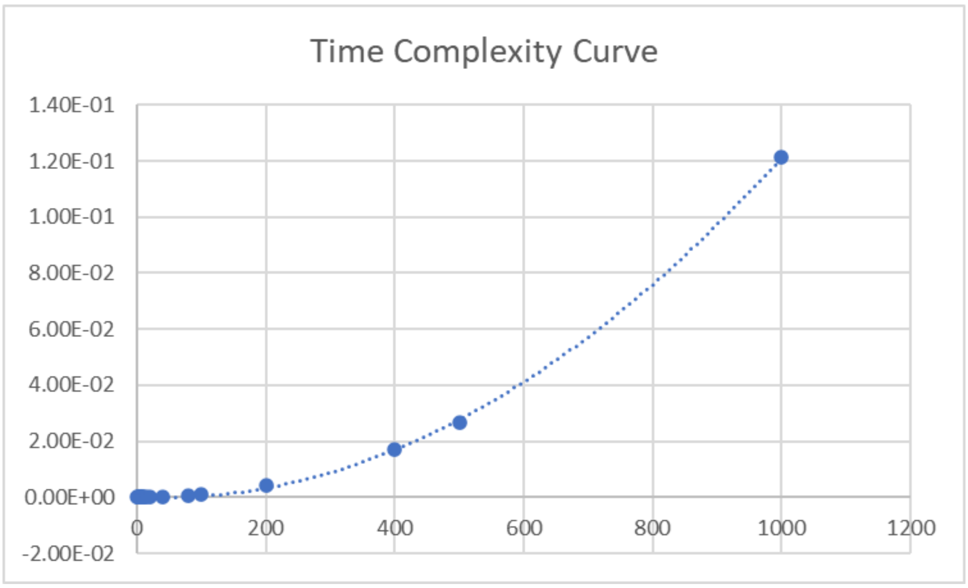


Table: Time-Complexity Curve

The result curve equation is y=1E-07x^2+0.0002 with R^2=0.9998.

Chapter 4: Analysis and Comments

4.1 Time Complexity Analysis

Clearly, the time-complexity is O(N^2).

4.2 Space Complexity Analysis

Clearly, the space-complexity is O(V).

Appendix: Source Code

The source code folder consists of one file: the source.cpp.

The source.cpp:

```
#include <iostream>
using namespace std;
int main() {
    long long modNum = 1000000007;
    long long resNum = 0;
    int maxN = 700;
    long long redNode[maxN][maxN], blackNode[maxN][maxN];
    int n, i, j, k, l;
    cin >> n;
    blackNode[1][1] = 1;
    blackNode[1][2] = 2;
    redNode[0][1] = 1;
    for(i = 0; i <= 10; i++)
    {
        for(j = 2; j < n; j++)
        {
            for(k = 1; k < j; k++)
            {
                redNode[i][j+1] +=
blackNode[i][j-k]*blackNode[i][k];
                blackNode[i+1][j+1] +=
(redNode[i][j-k]+blackNode[i][j-k])*(redNode[i][k]+blackNode[i][k
])%modNum;
                redNode[i][j+1] %= modNum;
                blackNode[i+1][j+1] %= modNum;
            }
        }
    }
    for(i = 0; i <= n; ++i)
    {
        resNum = (resNum+blackNode[i][n])%modNum;
    }
    cout << resNum;
    return 0;
}
```

References

Author List

Programmer: Yunce Zhang

Tester: Mengyu Wu

Document Writer: Haoxiang Huang

Declaration

We hereby declare that all the work done in this project titled " Red-Black Tree" is of our independent effort as a group.

Signatures