

## 华东师范大学数据科学与工程学院上机实践报告

课程名称：算法设计与分析

年级：19 级

上机实践成绩：

指导教师：金澈清

姓名：龚敬洋

上机实践名称：红黑树

学号：

上机实践日期：

10195501436

2020/11/13

上机实践编号：No.6

组号：1-436

### 一、目的

1. 熟悉算法设计的基本思想
2. 掌握构建红黑树的方法

### 二、内容与设计思想

1. 编写随机整数生成算法，生成 S 到 T 范围内的 N 个随机整数并输出；
2. 编写红黑树构建算法，中序遍历各节点，输出颜色和值；
3. 随机生成  $1e2$ 、 $1e3$ 、 $1e4$ 、 $1e5$ 、 $1e6$  个不同的数，使用红黑树构建算法，并画图描述不同情况下的运行时间差异；

### 三、使用环境

推荐使用 C/C++集成编译环境。

### 四、实验过程

#### 1. 写出红黑树构建算法的源代码

```
1. #include <iostream>
2. #include <fstream>
3. #include <cstdlib>
4. using namespace std;
5. struct node{
6.     int data;
7.     int color; //0 is black, 1 is red
8.     struct node *parent;
9.     struct node *lchild;
10.    struct node *rchild;
11. };
12. void lrotate(struct node *n){
13.     struct node *nr = n->rchild;
14.     n->rchild = nr->lchild;
15.     if (nr->lchild) nr->lchild->parent = n;
16.     nr->parent = n->parent;
17.     if(n->parent) {
18.         if (n == n->parent->lchild) n->parent->lchild = nr;
19.         else n->parent->rchild = nr;
20.     }
21.     nr->lchild = n;
22.     n->parent = nr;
23. }
24. void rrotate(struct node *n){
25.     struct node *nl = n->lchild;
```

```
26.     n->lchild = nl->rchild;
27.     if (nl->rchild) nl->rchild->parent = n;
28.     nl->parent = n->parent;
29.     if(n->parent) {
30.         if (n == n->parent->lchild) n->parent->lchild = nl;
31.         else n->parent->rchild = nl;
32.     }
33.     nl->rchild = n;
34.     n->parent = nl;
35. }
36. struct node *fixup(struct node *n){
37.     struct node *t;
38.     while(n->parent && n->parent->color == 1){
39.         if(n->parent == n->parent->parent->lchild){
40.             t = n->parent->parent->rchild;
41.             if (t && t->color == 1){
42.                 n->parent->color = 0;
43.                 t->color = 0;
44.                 n->parent->parent->color = 1;
45.                 n = n->parent->parent;
46.                 if(!n->parent) n->color = 0;
47.             }
48.             else {
49.                 if (n == n->parent->rchild) {
50.                     n = n->parent;
51.                     lrotate(n);
52.                 }
53.                 n->parent->color = 0;
54.                 n->parent->parent->color = 1;
55.                 rrotate(n->parent->parent);
56.             }
57.         }
58.         else{
59.             t = n->parent->parent->lchild;
60.             if (t && t->color == 1){
61.                 n->parent->color = 0;
62.                 t->color = 0;
63.                 n->parent->parent->color = 1;
64.                 n = n->parent->parent;
65.                 if(!n->parent) n->color = 0;
66.             }
67.             else {
68.                 if (n == n->parent->lchild) {
69.                     n = n->parent;
70.                     rrotate(n);
71.                 }
72.                 n->parent->color = 0;
73.                 n->parent->parent->color = 1;
74.                 lrotate(n->parent->parent);
75.             }
76.         }
77.     }
78.     t = n;
79.     while (t->parent){
80.         t = t->parent;
81.     }
82.     t->color = 0;
83.     return t;
84. }
85. struct node *insert(struct node *root, struct node *n){
86.     struct node *x, *y, *nroot;
87.     x = root;
88.     while(x){
89.         y = x;
90.         if(n->data < x->data){
91.             x = x->lchild;
```

```
92.     }
93.     else{
94.         x = x->rchild;
95.     }
96. }
97. n->parent = y;
98. if(n->data < y->data){
99.     y->lchild = n;
100. }
101. else{
102.     y->rchild = n;
103. }
104. n->color = 1;
105. nroot = fixup(n);
106. return nroot;
107. }
108. void traverse(struct node *n){
109.     if(!n) return;
110.     traverse(n->lchild);
111.     cout<<n->data<<" "<<n->color<<endl;
112.     traverse(n->rchild);
113. }
114. int main(){
115.     ifstream fin("data.txt");
116.     int a[100005], n = 0;
117.     while (!fin.eof()){
118.         fin>>a[n];
119.         n++;
120.     }
121.     n--;
122.     node *r = new node{a[0], 0, 0x0, 0x0};
123.     for(int i = 1; i < n; i++){
124.         node *n = new node{a[i], 1, 0x0, 0x0};
125.         r = insert(r, n);
126.     }
127.     traverse(r);
128.     fin.close();
129.     return 0;
130. }
```

## 2. 截取各个实验的实验结果

### 构建 10 个结点的红黑树的中序遍历结果

```
2 1
18 0
26 1
46 1
56 0
60 0
64 0
66 1
70 1
73 0
Total Time: 7e-06s

Process finished with exit code 0
|
```

构建 100 个结点的红黑树的运行时间

```
Total Time: 2.4e-05s
```

```
Process finished with exit code 0
```

构建 1000 个结点的红黑树的运行时间

```
Total Time: 0.000184s
```

```
Process finished with exit code 0
```

构建 10000 个结点的红黑树的运行时间

```
Total Time: 0.003021s
```

```
Process finished with exit code 0
```

构建 100000 个结点的红黑树的运行时间

```
Total Time: 0.054191s
```

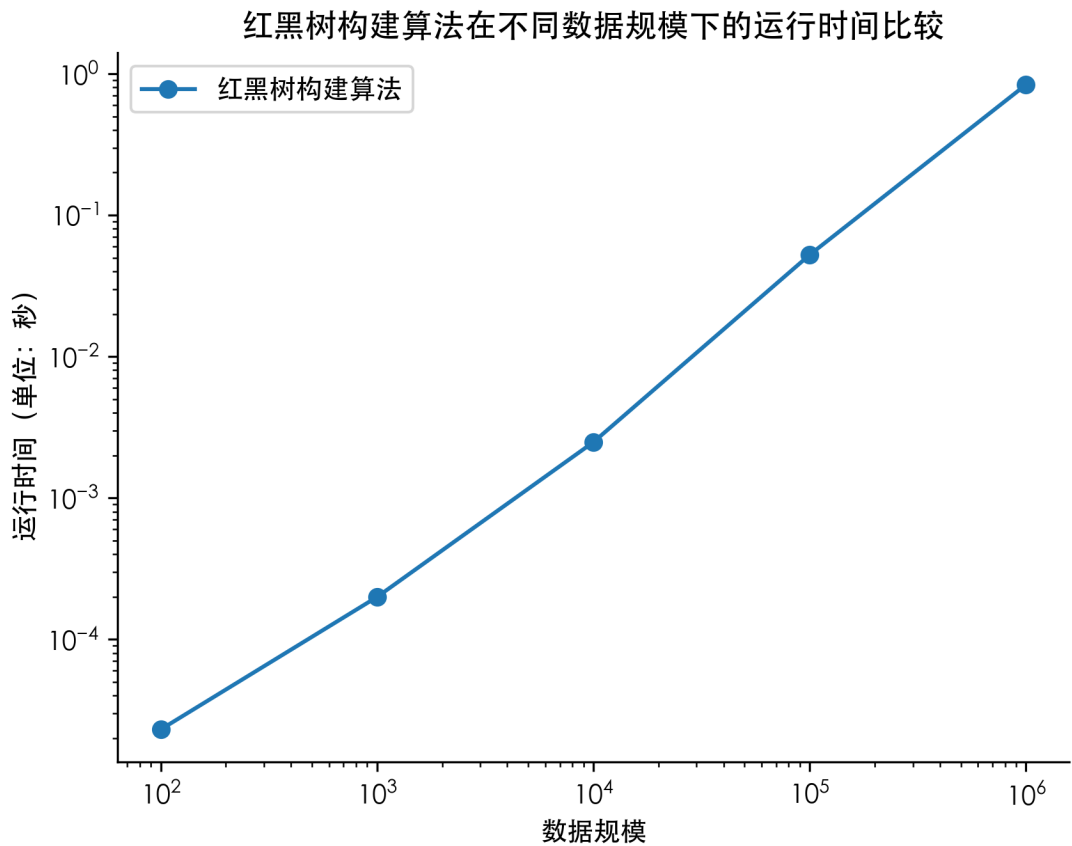
```
Process finished with exit code 0
```

构建 1000000 个结点的红黑树的运行时间

```
Total Time: 0.854415s
```

```
Process finished with exit code 0
```

3. 分别画出各个实验结果的折线图



五、总结

对上机实践结果进行分析，问题回答，上机的心得体会及改进意见。

对于一颗有 $n$ 个结点的红黑树，可以用 $O(\lg n)$ 的时间向其中插入一个新结点，故构建一棵有 $n$ 个结点红黑树的总运行时间为 $O(n \lg n)$ 。从图表中可以看出，在对数坐标下，红黑树构建算法随数据规模的增大呈线性增长，与理论基本吻合。