# 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：**10195501436　**上机实践日期：**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。

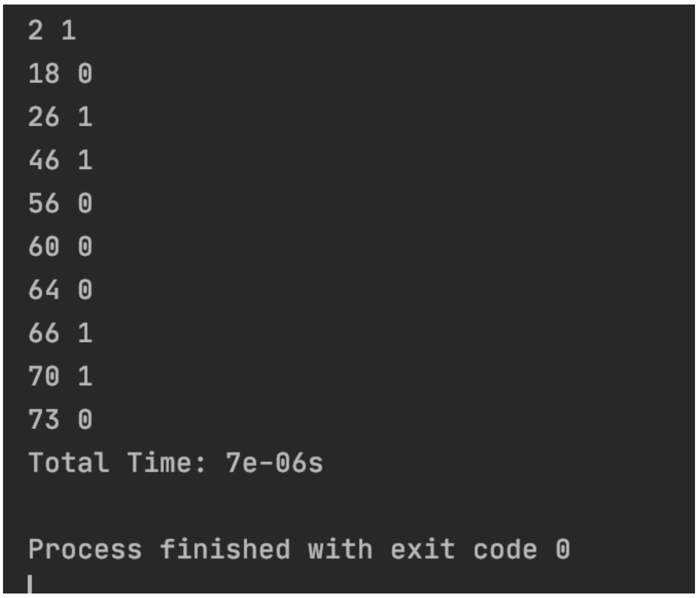
## 四、实验过程

### *1.　写出红黑树构建算法的源代码*

```
#include <iostream>
#include <fstream>
#include <cstdlib>
using namespace std;
struct node{
    int data;
    int color; //0 is black, 1 is red
    struct node *parent;
    struct node *lchild;
    struct node *rchild;
};
void lrotate(struct node *n){
    struct node *nr = n->rchild;
    n->rchild = nr->lchild;
    if (nr->lchild) nr->lchild->parent = n;
    nr->parent = n->parent;
    if(n->parent) {
        if (n == n->parent->lchild) n->parent->lchild = nr;
        else n->parent->rchild = nr;
    }
    nr->lchild = n;
    n->parent = nr;
}
void rrotate(struct node *n){
    struct node *nl = n->lchild;
    n->lchild = nl->rchild;
    if (nl->rchild) nl->rchild->parent = n;
    nl->parent = n->parent;
    if(n->parent) {
        if (n == n->parent->lchild) n->parent->lchild = nl;
        else n->parent->rchild = nl;
    }
    nl->rchild = n;
    n->parent = nl;
}
struct node *fixup(struct node *n){
    struct node *t;
    while(n->parent && n->parent->color == 1){
        if(n->parent == n->parent->parent->lchild){
            t = n->parent->parent->rchild;
            if (t && t->color == 1){
                n->parent->color = 0;
                t->color = 0;
                n->parent->parent->color = 1;
                n = n->parent->parent;
                if(!n->parent) n->color = 0;
            }
            else {
                if (n == n->parent->rchild) {
                    n = n->parent;
                    lrotate(n);
                }
                n->parent->color = 0;
                n->parent->parent->color = 1;
                rrotate(n->parent->parent);
            }
        }
        else{
            t = n->parent->parent->lchild;
            if (t && t->color == 1){
                n->parent->color = 0;
                t->color = 0;
                n->parent->parent->color = 1;
                n = n->parent->parent;
                if(!n->parent) n->color = 0;
            }
            else {
                if (n == n->parent->lchild) {
                    n = n->parent;
                    rrotate(n);
                }
                n->parent->color = 0;
                n->parent->parent->color = 1;
                lrotate(n->parent->parent);
            }
        }
    }
    t = n;
    while (t->parent){
        t = t->parent;
    }
    t->color = 0;
    return t;
}
struct node *insert(struct node *root, struct node *n){
    struct node *x, *y, *nroot;
    x = root;
    while(x){
        y = x;
        if(n->data < x->data){
            x = x->lchild;
        }
        else{
            x = x->rchild;
        }
    }
    n->parent = y;
    if(n->data < y->data){
        y->lchild = n;
        }
        else{
            y->rchild = n;
        }
        n->color = 1;
        nroot = fixup(n);
        return nroot;
    }
    void traverse(struct node *n){
        if(!n) return;
        traverse(n->lchild);
        cout<<n->data<<" "<<n->color<<endl;
        traverse(n->rchild);
    }
    int main(){
        ifstream fin("data.txt");
        int a[1000005], n = 0;
        while (!fin.eof()){
            fin>>a[n];
            n++;
        }
        n--;
        node *r = new node{a[0], 0, 0x0, 0x0};
        for(int i = 1; i < n; i++){
            node *n = new node{a[i], 1, 0x0, 0x0};
            r = insert(r, n);
        }
        traverse(r);
        fin.close();
        return 0;
    }
```

2. *截取各个实验的实验结果*

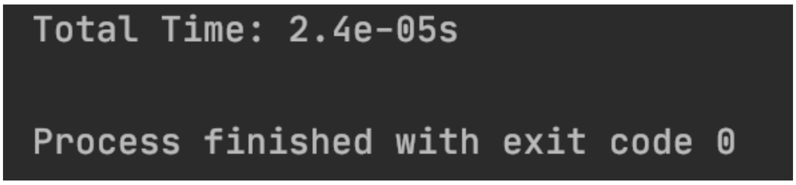
构建 10 个结点的红黑树的中序遍历结果



```
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
```

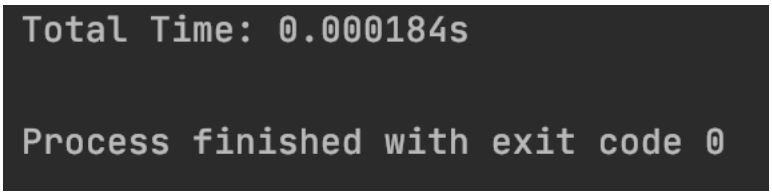
构建 100 个结点的红黑树的运行时间



```
Total Time: 2.4e-05s

Process finished with exit code 0
```

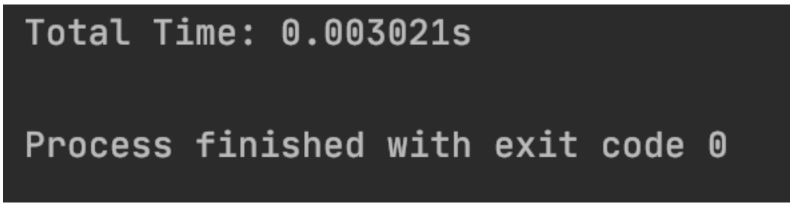
构建 1000 个结点的红黑树的运行时间



```
Total Time: 0.000184s

Process finished with exit code 0
```

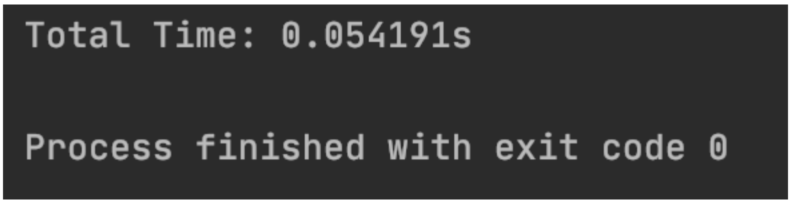
构建 10000 个结点的红黑树的运行时间



```
Total Time: 0.003021s

Process finished with exit code 0
```

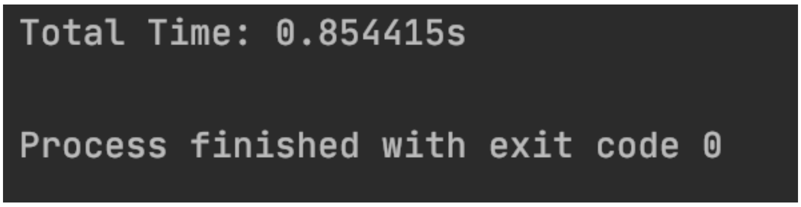
构建 100000 个结点的红黑树的运行时间



```
Total Time: 0.054191s

Process finished with exit code 0
```

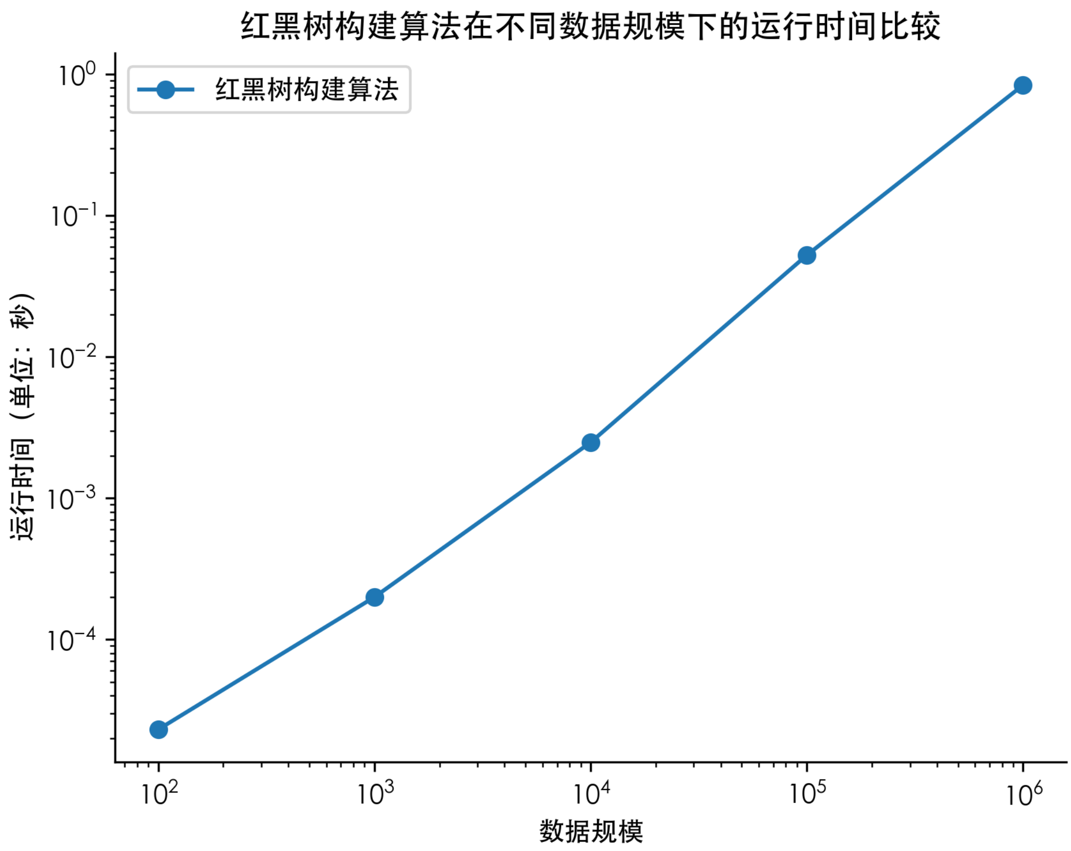
构建 1000000 个结点的红黑树的运行时间



```
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$O(lgn)$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$O(nlgn)$。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。