作业二：BST 排序算法实现

李沁霞
统计学 3210300363

2022 年 10 月 19 日

1 创建要求

编写 BSTSorting 函数实现数组排序，而必须使用儿茶搜素树排序算法。编写两种不同的格式，一种是乱序后排序，另一种是不乱序排序。使用两种方法测试函数运行时间。

2 设计思路

1. 编写 BSTSorting 函数的数组排序。
2. 编写 main 函数的为头文件实现测试。
3. 进行脚本文件编写。

3 添加函数

```
1.  void BSTSorting(vector<Comparable> &_arr, int _mode = 0)
    {
        BinarySearchTree<Comparable> tree;
        clock_t start,end;
        double time = 0;
        if(mode == 0)
        {
            start = clock;
            for(int i = 0; i < _arr.size(); ++i)
            {
                tree.insert(_arr[i]);
```

```
            }
            end = clock;
            time = double(end-start)/CLOCKS_PER_SEC;
        }
        if(mode == 1)
        {
            for(int j = 0; j < 100; ++j)
            {
                start = clock;
                tree.makeEmpty();
                for(int i< _arr.size()-1; i >= 1; --i)
                {
                    int k = random() % i;
                    Comparable temp = _arr[k];
                    _arr[k] = _arr[i];
                    _arr[i] = temp;
                }
                for(int i = 0; i < _arr.size(); ++i)
                {
                    tree.insert(_arr[i]);
                }
                end = clock;
                time += double(end-start)/CLOCKS_PER_SEC;
            }
        }
        if(_arr.size() <= 10000)
        {
            cout << "After sorting: " << endl;
            tree.printTree();
        }
        cout << "Time for executing: " << time << "s" << endl;
    }
```

2.
```
int main(){
        int mode;
        vector <int> _arr;
```

```
    cout << "Enter sorting mode: ";
    cin >> mode;

    while(cin.fail() || (mode != 0 && mode != 1))
    {
        cin.clear();
        cout << "Please insert 0 or 1." << "Enter sorting mode: " << endl
        cin >> mode;
    }
    cout << "Enter your array: " << endl;
    for(int temp = 0; cin >> temp;)
    {
        _arr.push_back(temp);
        if(cin.get() == '\n') break;
    }
    if(_arr.size() == 1)
    {
        int length = _arr.back();
        _arr.pop_back();
        for(int temp = length; temp >= 1; --temp)
        {
            _arr.push_back(temp);
        }
    }
    BSTSorting(_arr,mode);
    return 0;
}
```

4 Makefile 与 run 脚本

1. make:

```
    g++ -o test main.cpp

report:
    xelatex report.tex
```

```
rm report.aux
rm report.log
```

使用make:输入g++ -o test main.cpp测试程序的运行流程，
使用report:生成report.pdf的文件与remove某些report的文件。

2. #! /bin/bash

```
make test
make report
./test
```

使用make:执行test文件，使用./test命令测试main.cpp的程序

5 测试说明

对每一个数组都测试 2 种的时间复杂性模式.

Enter sorting mode: 0
Enter your array:
10000
Time for executing: 4.49529s

Enter sorting mode: 1
Enter your array:
10000
Time for executing: 0.238379s

Enter sorting mode: 0
Enter your array:
50000
Time for executing: 9.54646s

Enter sorting mode: 1
Enter your array:
50000
Time for executing: 1.73558s

上述测试结果显示乱序程序后排序 (mode = 1) 测试程序的时间比不乱序程序 (mode = 0) 更快.

使用 Valgrind 检查数组的泄露.

```
==5243==
==5243== HEAP SUMMARY:
==5243== in use at exit: 133,311 bytes in 770 blocks
==5243== total heap usage: 1,063 allocs, 293 frees, 158,123 bytes allocated
==5243==
==5243== LEAK SUMMARY:
==5243== definitely lost: 12 bytes in 1 blocks
==5243== indirectly lost: 0 bytes in 0 blocks
==5243== possibly lost: 0 bytes in 0 blocks
==5243== still reachable: 133,299 bytes in 769 blocks
==5243== suppressed: 0 bytes in 0 blocks
==5243== Rerun with –leak-check=full to see details of leaked memory
==5243==
==5243== For lists of detected and suppressed errors, rerun with: -s
==5243== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```