算法设计与分析第三次作业

2019202243 郭炜

题目描述：

在n枚硬币里有一枚重量不合格的硬币（过重或过轻），如果只有一个天平可以用来称量且称重的硬币数没有限制，设计一个算法找出这枚不合格的硬币，使得称重次数最少？给出算法的伪码描述，如果每称一次作为一次基本运算，算法复杂度是多少？算法输入为n，输出为最少称重次数，提交代码和运行结果截图。

Persuade Code:

1. 统计硬币总个数；
2. 如果，则两两比较即可找到不合格硬币；
3. 如果，则将硬币三等分；若无法三等分，则计算,前两部分为的硬币个数，第三部分为剩余硬币；
4. 接下来比较前两部分的硬币重量是否相等，相等则原问题转变为第三部分硬币的子问题，不相等则原问题转变为前两部分硬币的子问题；
5. 每次从原问题到子问题的转变过程中，要先判断子问题中硬币个数，如果前两部分的硬币重量相等，则需要判断该问题硬币个数是否为4，如果为4，则说明是112的情况，则不需要继续分治，直接4枚硬币两两判断即可确定不合格硬币的位置；如果前两部分的硬币重量不等，仍需判断该问题硬币个数是否为4，如果为4，则说明是211或者121的情况，依旧不需要继续分支，直接4枚硬币两两判断即可确定不合格硬币的位置。

算法复杂度：

最坏情况下，设最长路径长度为，则，求出，因此递归树的层数为，每层节点的数值和为，所以：

Code：

#include <iostream>

#include <vector>

#include <ctime>

using namespace std;

*// 统计天平称重次数*

int getCount = 0;

*// 统计子硬币总重量*

int getSum(const vector<int>& *v*, int *left*, int *right*)

{

    int sum = 0;

    for (size\_t i = left; i <= right; ++i)

    {

        sum += v.at(i);

    }

    return sum;

}

*// 天平 判断两组硬币是否相等*

bool isEqual(const vector<int>& *v*, int *l1*, int *r1*, int *l2*, int *r2*)

{

    ::getCount += 1;

    int sum1 = getSum(v, l1, r1);

    int sum2 = getSum(v, l2, r2);

    return sum1 == sum2 ? true : false;

}

*// 分治法查找*

int search(vector<int>& *co*, int *left*, int *right*)

{

    int count = right - left + 1;

*// 三枚硬币的情况下两两判断即可*

    if (count == 3)

    {

        if (isEqual(co, left, left, left + 1, left + 1))

        {

            return left + 2;

        }

        else if (isEqual(co, left, left, left + 2, left + 2))

        {

            return left + 1;

        }

        else

        {

            return left;

        }

    }

    int add = count / 3;

    if (count > 3)

    {

*// 三等分，若无法三等分，则前两部分相等，剩下为第三部分*

*// 相等说明不合格硬币在第三部分，不相等说明不合格硬币在前两部分*

        if (isEqual(co, left, left + add - 1, left + add, left + add \* 2 - 1))

        {

*// 1 1 2*

            if (right - left - add \* 2 + 1 <= 2)

            {

                if (isEqual(co, left, left, left + add \* 2, left + add \* 2))

                {

                    return left + add \* 2 + 1;

                }

                else

                {

                    return left + add \* 2;

                }

            }

            else

            {

                return search(co, left + add \* 2, right);

            }

        }

        else

        {

*// 1 2 1或者 2 1 1*

            if (add == 1)

            {

                if (isEqual(co, left + add \* 2, left + add \* 2, left, left))

                {

                    return left + add;

                }

                else

                {

                    return left;

                }

            }

            else

            {

                return search(co, left, left + add \* 2 - 1);

            }

        }

    }

}

int main()

{

srand((unsigned)time(NULL));

*// 保证至少有3枚硬币*

    int cur = rand() % 50 + 3;

    cout << "Size is " << cur << endl;

vector<int> coins(cur, 1);

*// vector<int> coins(cur, 2);*

    int cur2 = rand() % cur;

    cout << "The unqualified coins set in " << cur2 << endl;

coins.at(cur2) = 2;

*// coins.at(cur2) = 1;*

    int left = 0;

    int right = coins.size() - 1;

    for (int v : coins)

    {

        cout << v << " ";

    }

    cout << endl;

    cout << "The unqualified coins set in " << search(coins, left, right) << endl;

    cout << "Weighing times is :" << ::getCount << endl;

}

Result:

若硬币个数少于3个，由于不合格硬币过重或过轻不确定，则无法判断，因此实验假设硬币个数大于等于3个。并且代码中保证了至少3枚硬币。

假设不合格硬币过重情况下：

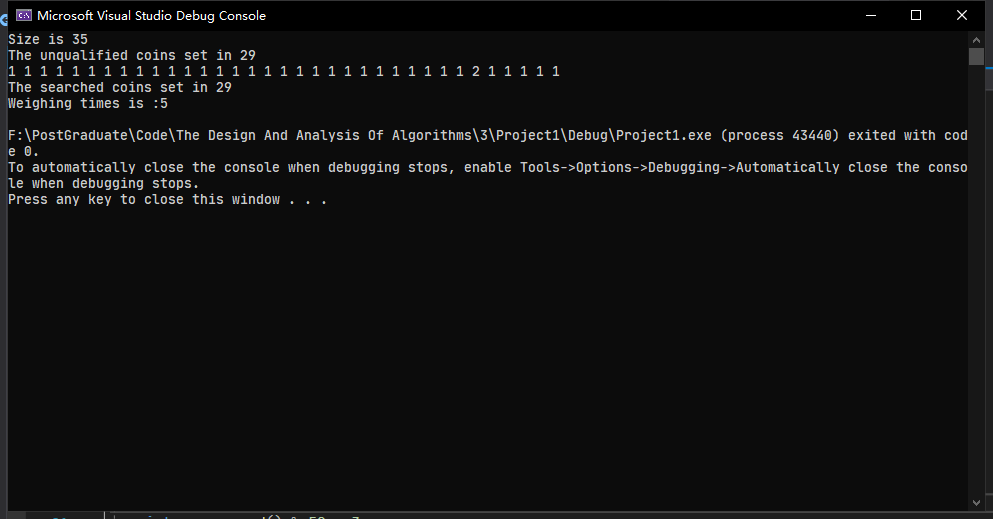
第一行为输入硬币总数

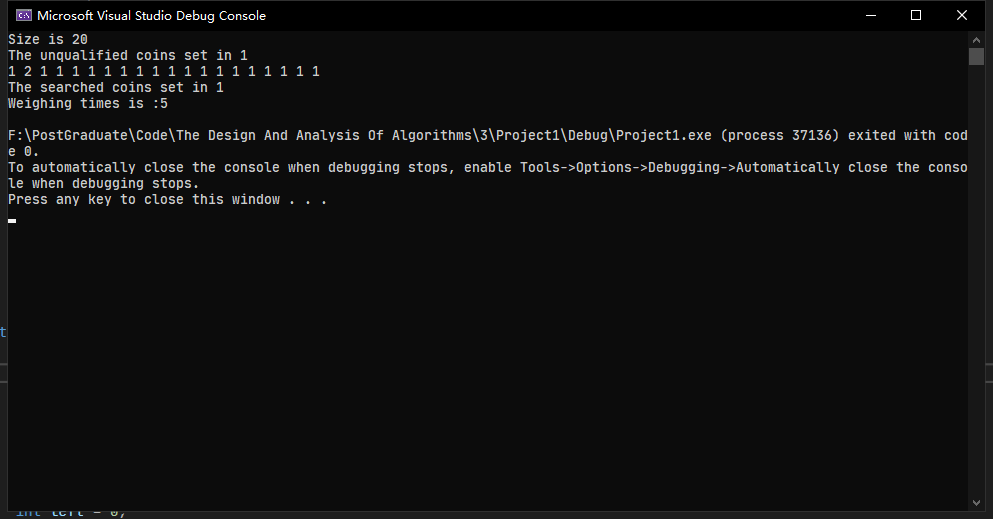
第二行为设定的不合格硬币所在位置

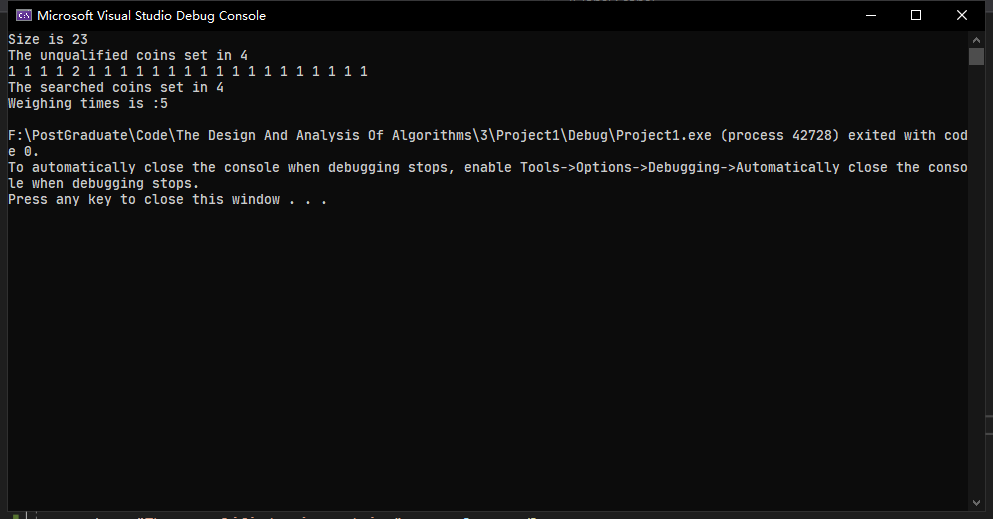
第三行为硬币重量

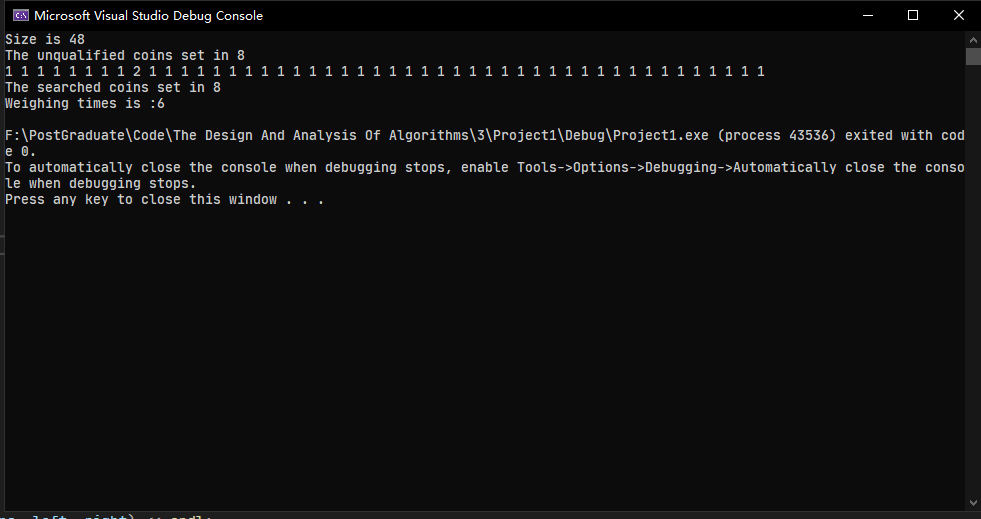
第四行为搜索到的不合格硬币所在位置

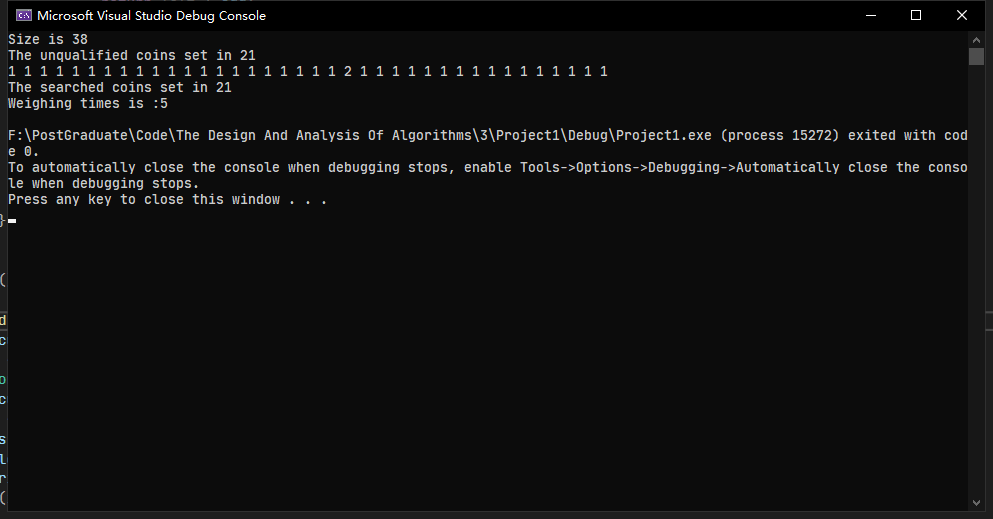
第五行为天平称重次数











假设不合格硬币过轻情况下：

第一行为输入硬币总数

第二行为设定的不合格硬币所在位置

第三行为硬币重量

第四行为搜索到的不合格硬币所在位置

第五行为天平称重次数

