项目完成报告

组合数学

——硬币中的组合数

作 者 姓 名： 张喆

学 号： 1754060

指 导 教 师： 冯巾松

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

目 录

[1 分析 1](#_Toc495668153)

[1.1 背景分析 1](#_Toc495668154)

[1.2 功能分析 1](#_Toc495668155)

[2 设计 2](#_Toc495668156)

[2.1 数据结构设计 2](#_Toc495668157)

[2.2 类结构设计 2](#_Toc495668158)

[2.3 成员与操作设计 2](#_Toc495668159)

[2.4 系统设计 3](#_Toc495668160)

[3 实现 4](#_Toc495668161)

[3.1 插入功能的实现 4](#_Toc495668162)

[3.1.1 插入功能流程图 4](#_Toc495668163)

[3.1.2 插入功能核心代码 5](#_Toc495668164)

[3.1.3 插入功能截屏示例 6](#_Toc495668165)

[3.2 删除功能的实现 7](#_Toc495668166)

[3.2.1 删除功能流程图 7](#_Toc495668167)

[3.2.2 删除功能核心代码 7](#_Toc495668168)

[3.2.3 删除功能截屏示例 8](#_Toc495668169)

[3.3 查找功能的实现 9](#_Toc495668170)

[3.3.1 查找功能流程图 9](#_Toc495668171)

[3.3.2 查找功能核心代码 9](#_Toc495668172)

[3.3.3 查找功能截图示例 10](#_Toc495668173)

[3.4 修改功能的实现 11](#_Toc495668174)

[3.4.1 修改功能流程图 11](#_Toc495668175)

[3.4.2 修改功能核心代码 11](#_Toc495668176)

[3.4.3 修改功能截屏示例 12](#_Toc495668177)

[3.5 统计功能的实现 13](#_Toc495668178)

[3.5.1 统计功能流程图 13](#_Toc495668179)

[3.5.2 统计功能核心代码 13](#_Toc495668180)

[3.5.3 统计功能截屏示例 14](#_Toc495668181)

[3.6 总体系统的实现 14](#_Toc495668182)

[3.6.1 总体系统流程图 14](#_Toc495668183)

[3.6.2 总体系统核心代码 14](#_Toc495668184)

[3.6.3 总体系统截屏示例 16](#_Toc495668185)

[4 测试 1](#_Toc495668186)7

[4.1 功能测试 1](#_Toc495668187)7

[4.1.1 插入功能测试 1](#_Toc495668188)7

[4.1.2 删除功能测试 18](#_Toc495668189)

[4.1.3 查找功能测试 19](#_Toc495668190)

[4.1.4 修改功能测试 19](#_Toc495668191)

[4.1.5 统计功能测试 19](#_Toc495668192)

[4.2 边界测试 2](#_Toc495668193)0

[4.2.1 初始化无输入数据 2](#_Toc495668194)0

[4.2.2 删除头结点 2](#_Toc495668195)0

[4.2.3 删除后链表为空 2](#_Toc495668196)1

[4.3 出错测试 2](#_Toc495668197)2

[4.3.1 考生人数错误 2](#_Toc495668198)2

[4.3.2 操作码错误 2](#_Toc495668199)2

[4.3.3 插入位置不存在 2](#_Toc495668200)2

[4.3.4 删除考号不存在 2](#_Toc495668201)3

[4.3.5 查找考号不存在 2](#_Toc495668202)3

[4.3.6 修改考号不存在 2](#_Toc495668203)3

# 1 分析

# 1.1 题目说明

比较三种货币（人民币、美元、欧元）中硬币构成各自货币里10元的组合数。对比说明每种硬币的表现力

人民币有六种面额的硬币，分别是1分、2分、5分、1角、5角、1元。

美元有六种面额的硬币，分别是1美分、5美分、10美分、25美分，50美分、1美元。

欧元有八种面额的硬币，分别是 1欧分、2欧分、5欧分、10欧分、20欧分、50欧分、1欧元和2欧元。

## 1.2 背景分析

硬币中的组合数项目是解决“不同国家发行的不同数量和面额的硬币构成各自货币中的10元的组合数”问题。

不同的硬币种类在组成同样的金额时的表现能力差别很大，这对国家和地区发行货币的种类和金额有很重要的意义。究竟什么样的组合可以让货币的表性能力更好，这是一个值得我们去研究的问题。

## 1.3 解题思路

本项目采用多种方法求解硬币中的组合数，并比较各种方法的优劣，以及组合数学中母函数引入的重要意义。

最简单的是用C语言实现暴力穷举法，把所有的可能情况模拟一遍，可以得到三种货币中硬币构成各自货币中1分~10元的全部组合数。但这样的缺点也很明显：首先，效率很低，由于每种货币的硬币种类都很多，所有穷举起来的计算量很大（事实上金额达到190分时计算时间已经很长了）；其次，不能直观的显示各种货币的表现能力。

第二种方法使用C++，通过动态规划的方法逐步求解指定硬币种类和金额的组合数。优点是效率很高；缺点也是不能通过图形直观的显示各种货币的表现能力。

第三种方法是将各个货币的各个面额的硬币的母函数写出来，然后写出该货币的母函数，通过母函数各项的系数理论上可以得出任意金额的组合数。该项目使用MATLAB的符号运算求解高阶多项式。

# 2 算法设计

## 2.1 穷举法

C语言实现

### 2.1.1 穷举法核心思想

举人民币为例，总共有1分、2分、5分、1角、5角、1元共六种硬币，而每一种硬币对应的取值情况可能是0~amout/金额个。

因此可以使用6个变量，分别表示六种硬币种类，当硬币数量\*硬币金额的累加和等于amount的时候则令计数变量递增一。

如此循环即可计算出六种硬币表示1000分的组合数。

### 2.1.2 穷举法核心代码

int Combination\_Yuan(int amount)

{

int cnt = 0;

for (int a1 = 0; a1 <= amount / 1; ++a1)

{

for (int a2 = 0; a2 <= amount / 2; ++a2)

{

for (int a3 = 0; a3 <= amount / 5; ++a3)

{

for (int a4 = 0; a4 <= amount / 10; ++a4)

{

for (int a5 = 0; a5 <= amount / 50; ++a5)

{

for (int a6 = 0; a6 <= amount / 100; ++a6)

{

if (a1 \* 1 + a2 \* 2 + a3 \* 5 + a4 \* 10 + a5 \* 50 + a6 \* 100 == amount)

{

++cnt;

}

}

}

}

}

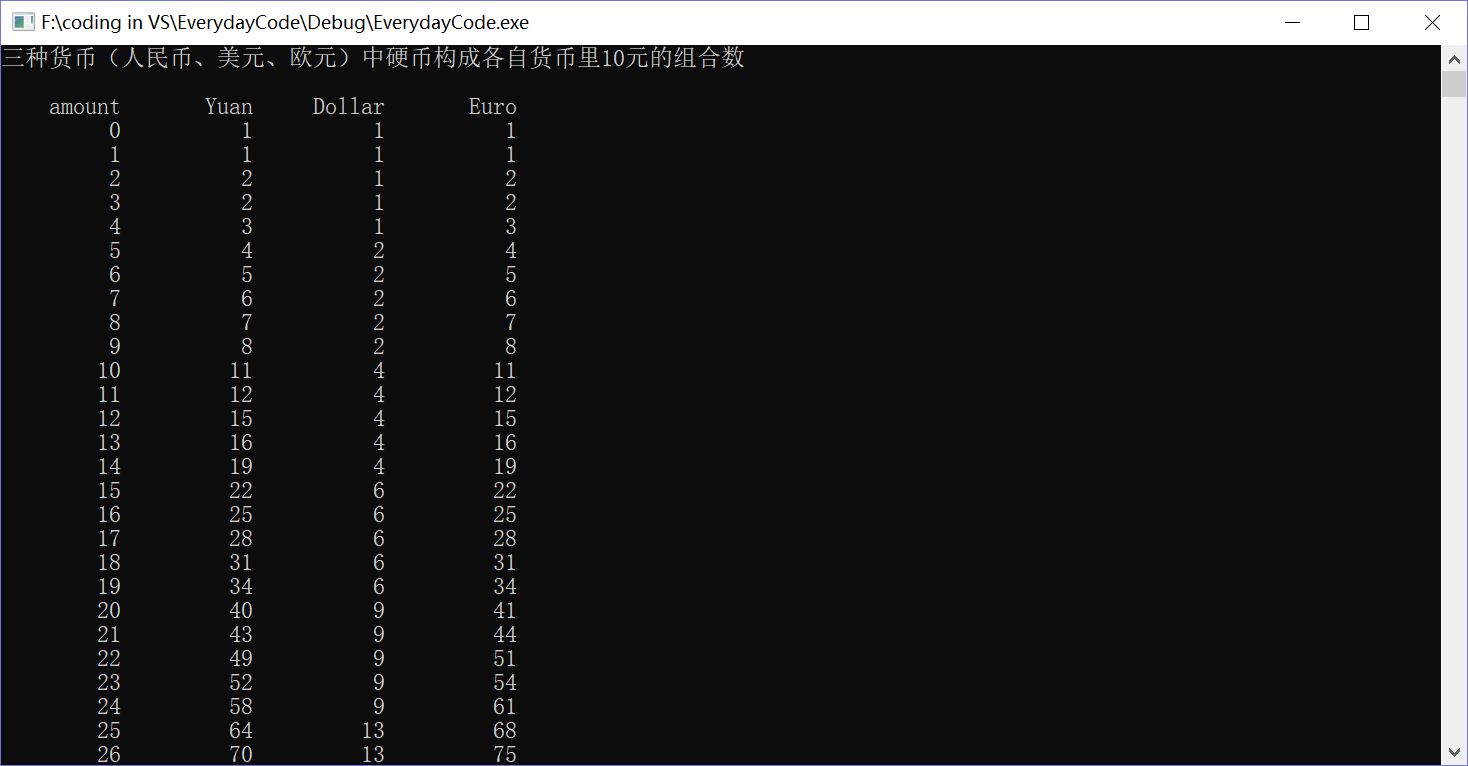
}

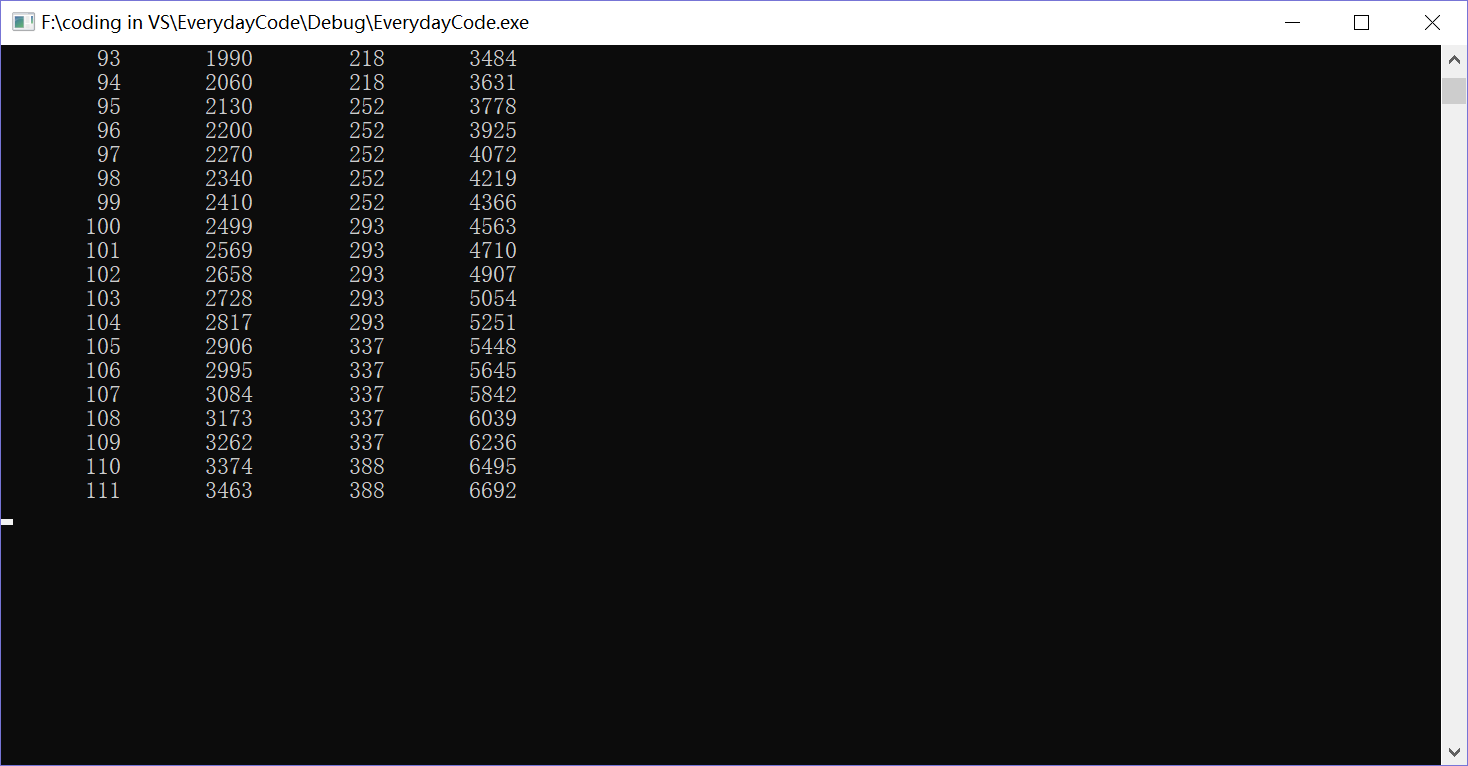
}

return cnt;

}

### 2.1.3 穷举法运行截图





### 2.1.4 穷举法优缺点分析

优点在于思想简单，代码易于编写，因为计算机的计算速度远大于人的手算速度，在金额不大的情况下不失为首选的方法。

然而该算法的时间复杂度达到，当总金额较大的时候对于计算机而言仍是不可解的。本题而言，在我的笔记本上跑到190的时候已经非常非常慢了。

## 2.2 动态规划法

C++实现

### 2.2.1 动态规划法核心思想

我们有m种不同的硬币{v1,v2,…,vm}，要组合成给定的金额sum

即sum=x1\*v1 + x2\*v2 +…+ xm\*vm中x1~xm所有可能的个数

对于第m种硬币，它可以使用1,2,…,k次(k=sum/vm)

则对于第m种硬币的不同取值情况我们可以把sum分解成

sum=x1\*v1 + x2\*v2 +…+ 0\*vm

sum=x1\*v1 + x2\*v2 +…+ 1\*vm

…

sum=x1\*v1 + x2\*v2 +…+ k\*vm

定义一个二维数组result[i][sum]表示用前i中硬币构成sum的所有组合数

对于人民币，问题的规模就是i=6，sum=1000的组合数

### 2.2.2 动态规划法核心代码

vector<int> Yuan = { 1,2,5,10,50,100 };

vector<vector<int> >ComYuan(Yuan.size()+1, vector<int>(amount+1, 0)); //增加0种硬币的情况 和 0元钱的情况

void Combination(const vector<int> &coins, vector<vector<int> > &result, const int amount=1000)

{

//把0种硬币组成任意金额的位置置为0

for (int sum = 0; sum <= amount; ++sum)

{ result[0][sum] = 0; }

//任意硬币数量组成0元有1种可能

for (int i = 0; i <= coins.size(); ++i)

{

result[i][0] = 1;

}

for (int i = 1; i <= coins.size(); ++i)

{

for (int sum = 1; sum <= amount; ++sum)

{

for (int k = 0; k <= sum / coins[i-1]; ++k)

{

result[i][sum] += result[i - 1][sum - k \* coins[i-1]];

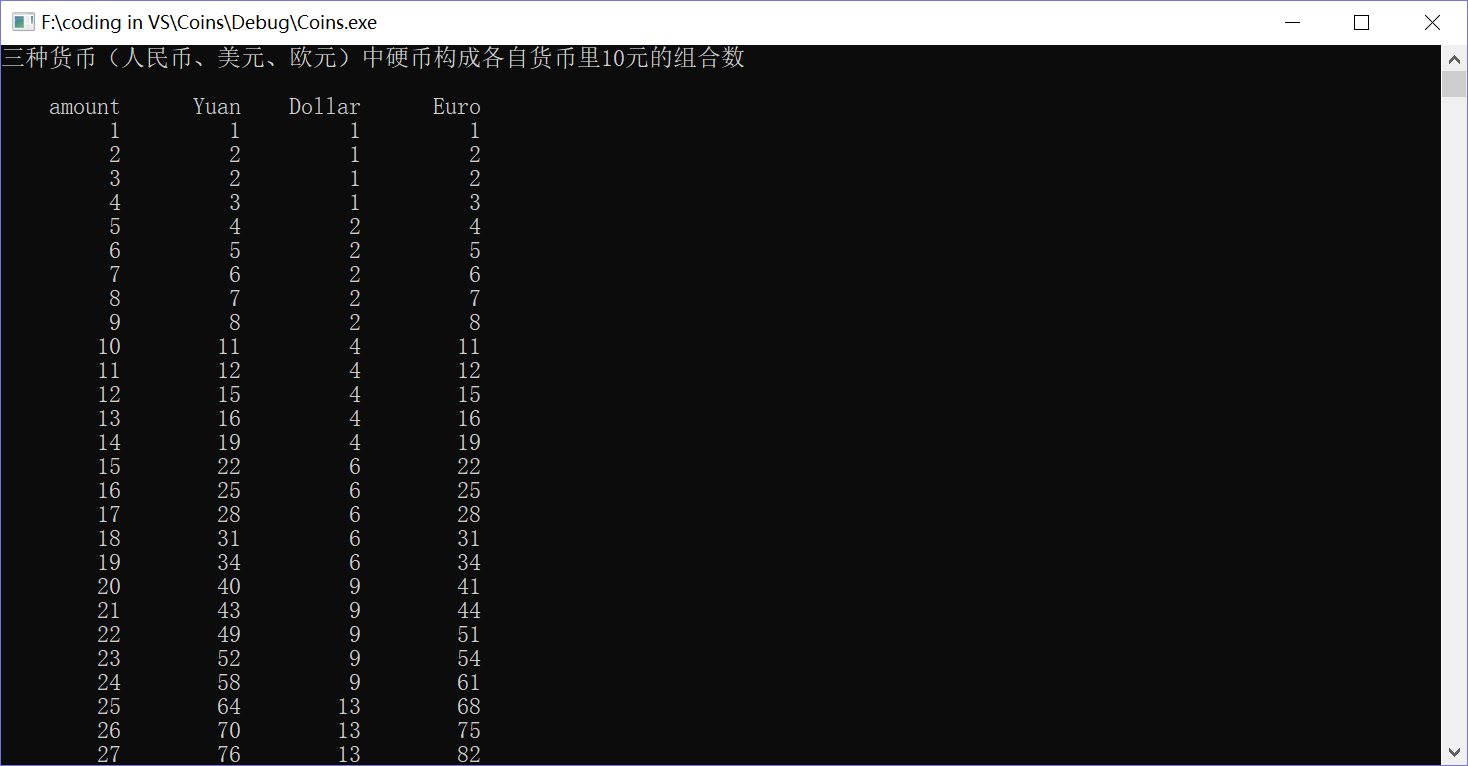
}

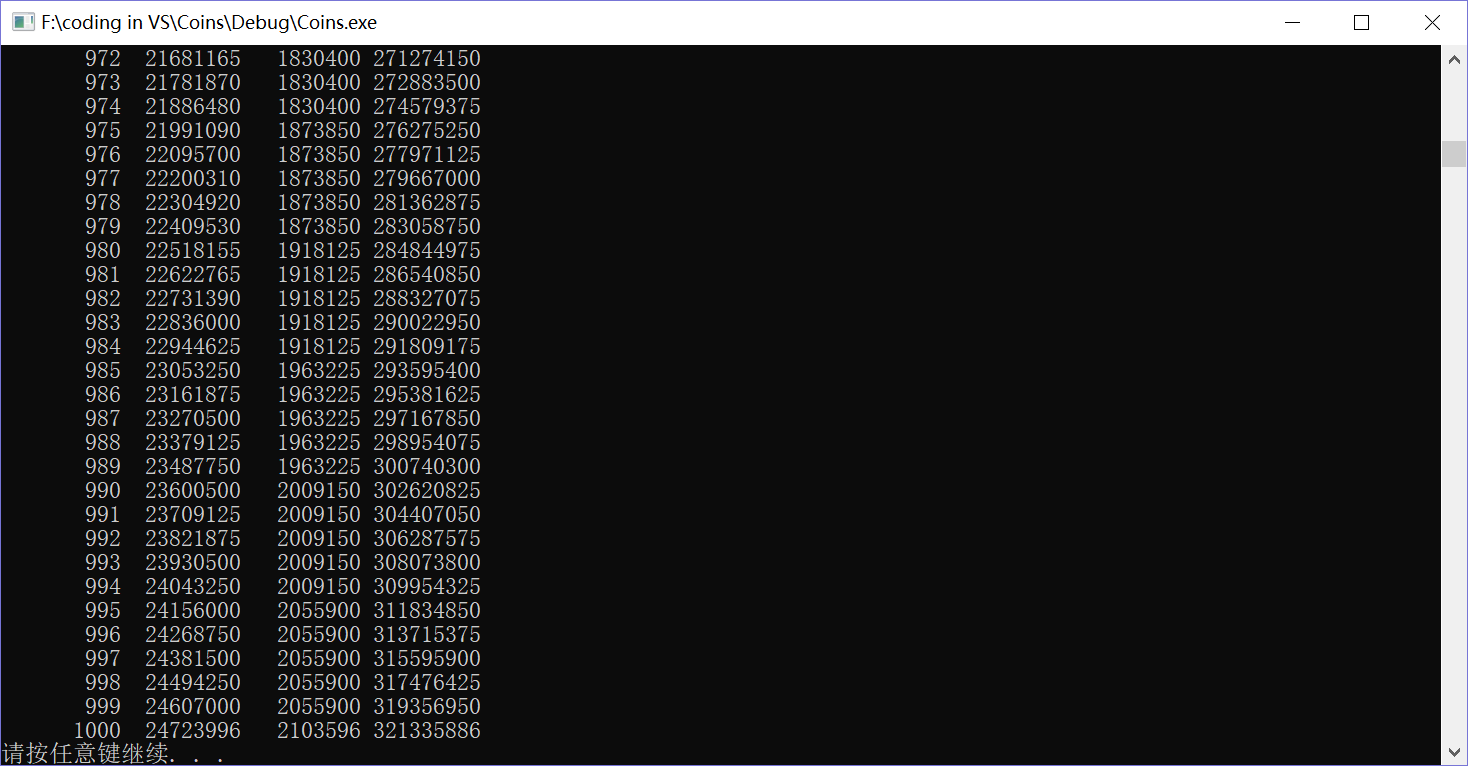
}

}

}

### 2.2.3 动态规划法运行截图





### 2.2.4 动态规划法优缺点分析

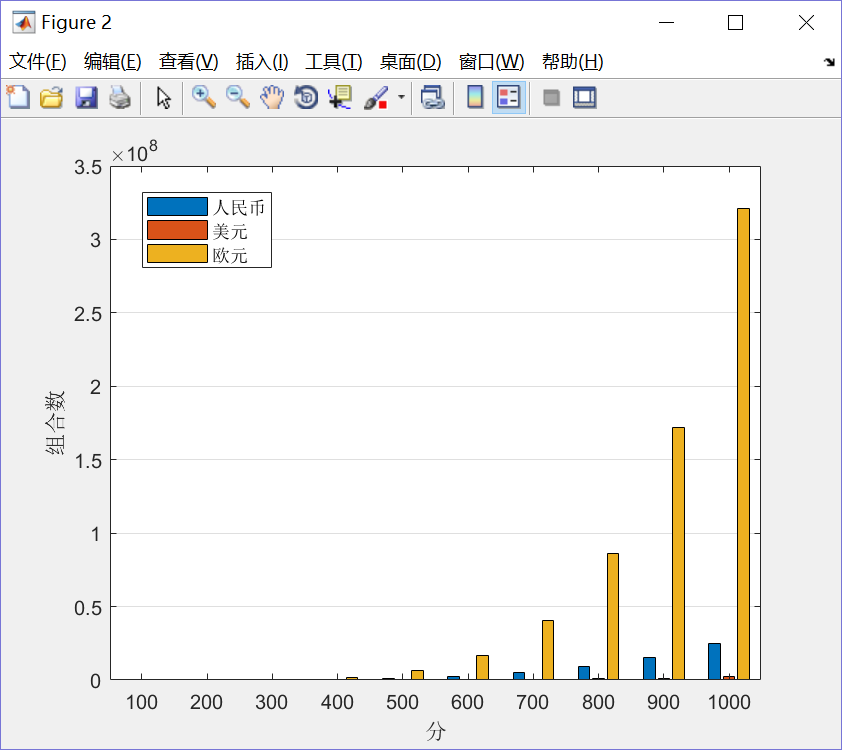
## 2.2 母函数法

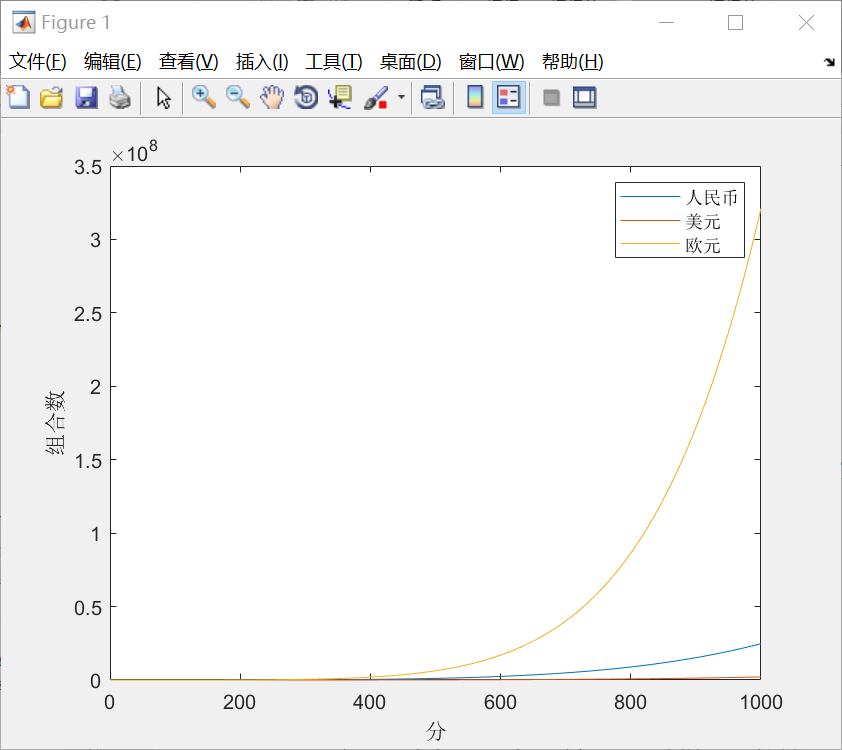
MATLAB实现

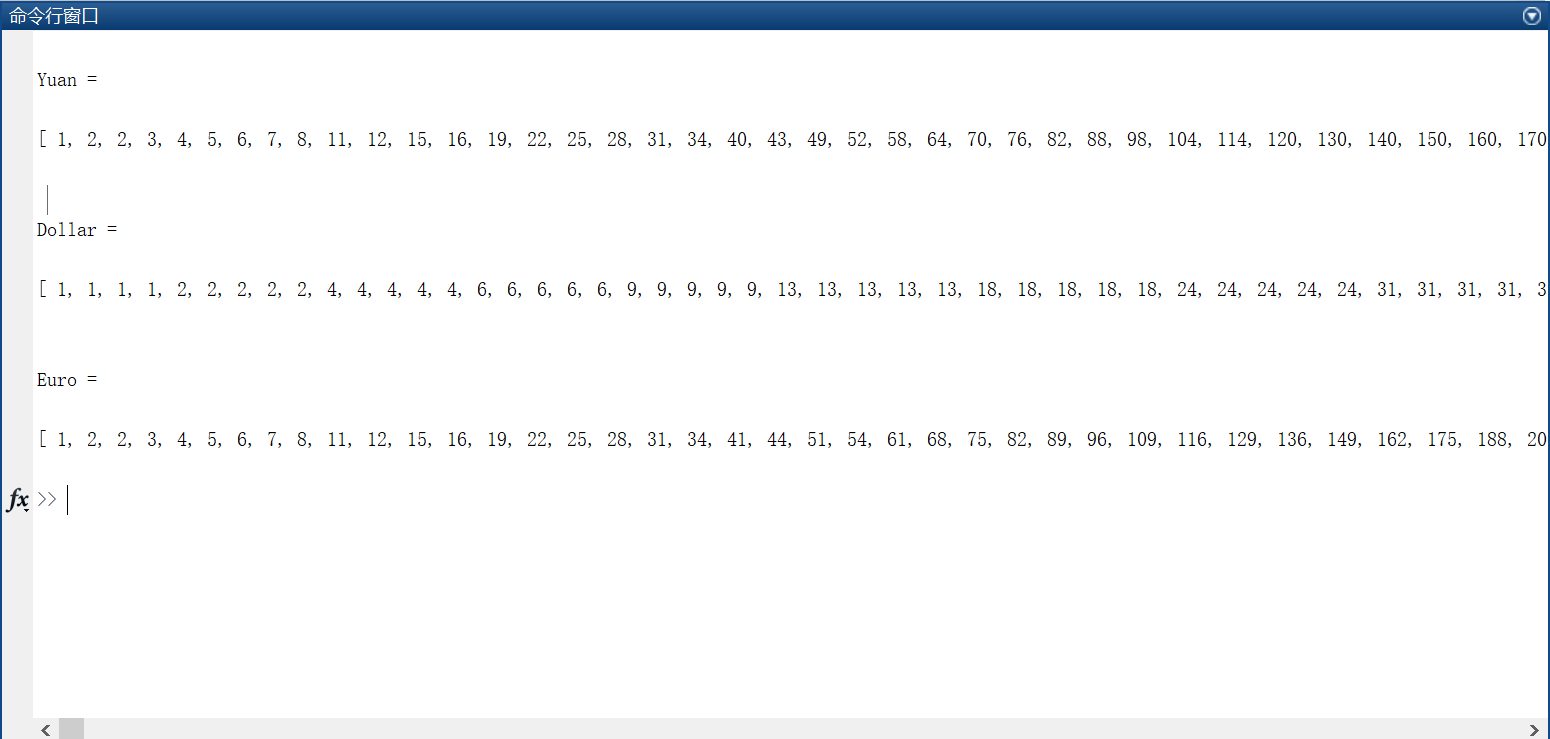
### 3.2.1 母函数法核心思想

### 3.2.2 母函数法核心代码

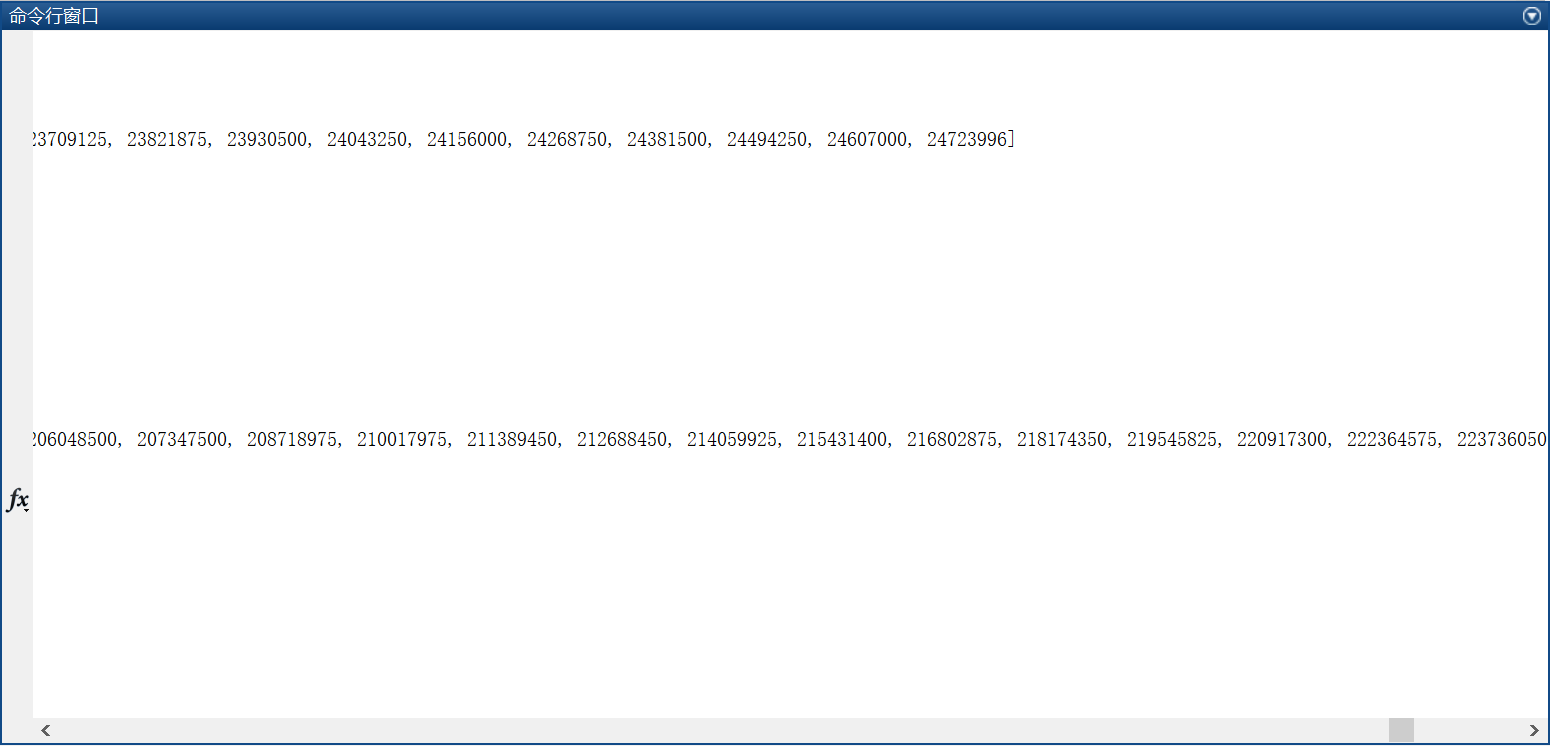
### 3.2.3 母函数法运行截图

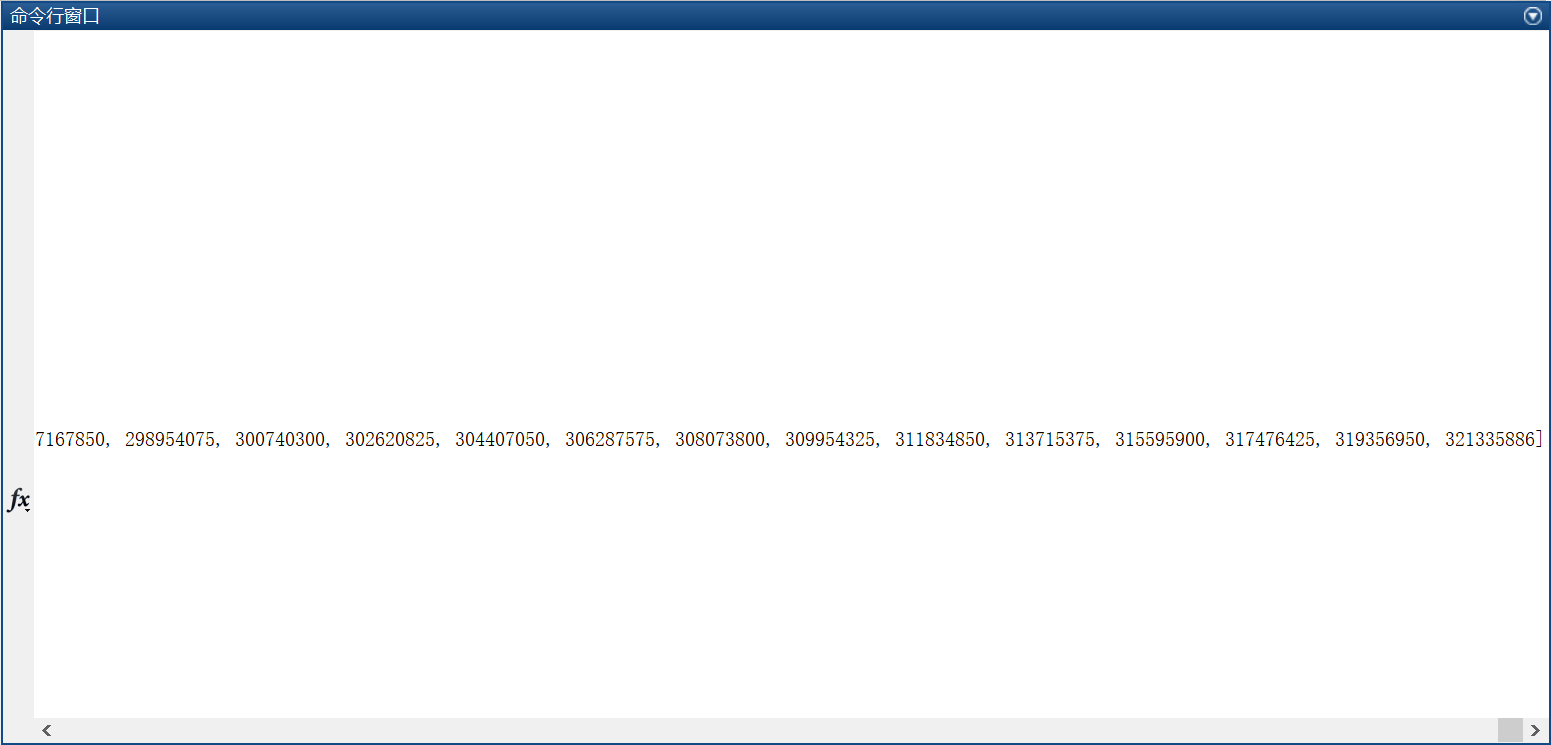












### 3.2.4 母函数法优缺点分析

# 3 结果分析

## 3.1 插入功能的实现

### 3.1.1 插入功能流程图

# 4 结论