| 学号 | 2022212080 | 算法思路(30%) | 编码实现与算法分析(50%) | 实验报告(20%) | 得分 |
|---|---|---|---|---|---|
| 姓名 | 刘纪彤 | | | | |
| 评语 | | | | | |

# 《算法设计与分析》实验报告

## 实验 5 回溯法实验

### 一、*实验目的*

1. 加深对回溯法的理解，了解其基本思想和算法流程。
2. 学习如何在具体问题中应用回溯法设计算法。
3. 分析回溯法与分支限界法的区别和联系，以及它们在不同类型问题中的适用性。
4. 提高算法设计能力和编程实践能力。

### 二、实验内容(题目)

1、假设某国家发行了 n 种不同面值的邮票，并且规定每张信封上最多只允许贴 m 张邮票。连续邮资问题要求对于给定的 n 和 m，给出邮票面值的最佳设计，在 1 张信封上贴出从邮资 1 开始，增量为 1 的最大连续邮资区间。 例如当 n=5，m=4 时，面值为 1，3，11，15，32 的 5 种邮票可以贴出邮资的最大连续区间是 1 到 70。

### 三、算法设计思路

主要包括两个函数：traceback 和 main。

traceback 函数是一个递归函数，用于回溯搜索所有可能的邮票面值组合。它接收四个参数：temp 是一个向量，用于存储当前的邮票面值组合；cur 是当前正在考虑的邮票的索引；maxt 是当前邮票面值组合可以达到的最大邮资；m 是每张信封最多允许贴的邮票数量。

traceback 函数的设计思路如下：

如果 cur 等于 n，表示已经考虑了所有的邮票，此时如果 maxt 大于 maxRe，则更新 maxRe 和 result。

否则，对于 cur 位置的邮票，尝试所有可能的面值（从 temp[cur] + 1 到 maxt + 1），并对每种可能的面值，计算下一个面值取该值时能达到的最大邮资 maxl。如果 maxl 大于 maxt，则表示该面值是可取的，继续递归确定接下来的面值。
main 函数是程序的入口点，它首先读入邮票的种类数 n 和每张信封上允许贴的

最多邮票数 m，然后调用 traceback 函数进行回溯搜索。最后，输出最优的邮票面值组合和最大邮资区间。

main 函数的设计思路如下：

首先，读入邮票的种类数 n 和每张信封上允许贴的最多邮票数 m。
然后，初始化 temp 和 result 向量，设置 temp[1]为 1。
调用 traceback 函数进行回溯搜索，寻找最优的邮票面值组合。
最后，输出最优的邮票面值组合和最大邮资区间。
这段代码的算法步骤如下：

读入邮票的种类数 n 和每张信封上允许贴的最多邮票数 m。
初始化 temp 和 result 向量，设置 temp[1]为 1。
调用 traceback 函数进行回溯搜索，寻找最优的邮票面值组合。
输出最优的邮票面值组合和最大邮资区间。

## 四、各功能模块设计

```
#include<bits/stdc++.h>
using namespace std;

int n; // n 种面值
int m; // 最多允许贴m 张
vector<int> result; // 最终结果
vector<int> temp; // 记录已经确定下来的面值
int maxRe = 0; //最大邮资

//判断由已选的前 cur 个面值,在张数不超过 m 的情况下能否贴出该邮资 re
//回溯法判断 re:当前需要贴出的邮资
/*
 temp 包含邮票面值的向量。
 cur 当前考虑的邮票面值的索引。
 re 剩余需要组成的总和。
 m 可以使用的最大邮票数量。
 num 当前使用的邮票数量。
 return 如果可以使用给定条件组成和为're'的总和，则返回true，否则返回fals
e。
 */
bool checkAnswer(vector<int>& temp, int cur, int re, int m, int num)
{
    // 递归出口
    if (re == 0) return true;
    if (re < 0) return false;
    for (int i = cur; i >= 1; i--) {
        if (num < m) {
```

```
            num++;
            if (checkAnswer(temp, cur, re - temp[i], m, num)) // 使用了
第 i 个面值的邮票
                return true;
            else
                num--; // 恢复现场
        }
    }
    return false;
}
/*
temp:包含邮票面值的数组。
cur:当前考虑的邮票面值的索引。
m:可以使用的最大邮票数量。
maxt:当前已经贴出的最大邮资。
*/
//找出由已选的前 cur 个面值中选,在张数不超过 m 的情况下能贴出的最大邮资
int maxl(vector<int>& temp, int cur, int m, int maxt) {
    int k;
    for (k = maxt + 1; k < m * temp[cur]; k++) {//从 maxt+1 开始尝试
        if (!checkAnswer(temp, cur, k, m, 0))//如果不能贴出 k 邮资
            break;
    }
    return k - 1;//返回 k-1
}
//递归找出最大邮资
/*
temp:包含邮票面值的数组。
cur:当前考虑的邮票面值的索引。
m:可以使用的最大邮票数量。
maxt:当前已经贴出的最大邮资。
*/
void traceback(vector<int>& temp, int cur, int maxt, int m) {
    if (cur == n) { //遍历到叶子
        if (maxRe < maxt) {
            maxRe = maxt;
            result = temp;
        }
        return;
    }
    for (int k = temp[cur] + 1; k <= maxt + 1; k++) {
        temp[cur + 1] = k;
        int max1 = maxl(temp, cur + 1, m, maxt);//下一个面值取 k 时能达到
的最大邮资
```

```
        if (max1 > maxt) {//k 可取
            traceback(temp, cur + 1, max1, m);//继续递归确定接下来的面值

        }
    }
}

int main() {
    cout << "请输入面值数量(n): " << endl;
    cin >> n;
    cout << "请输入每张信封最多允许贴的邮票数量(m): " << endl;
    cin >> m;
    temp.resize(n + 1);
    result.resize(n + 1);
    temp[1] = 1;
    traceback(temp, 1, m, m);
    cout << endl << "邮票面值分别为: ";
    for (int i = 1; i <= n; i++)
        cout << result[i] << " ";
    cout << endl;
    cout << endl << "最大邮资区间为: 1-" << maxRe << endl;
    return 0;
}
```

## 五、运行结果与分析

```
PS F:\Study-Program\源代码存储\算法设计\实验\实验5> & .\'1.exe'
请输入面值数量(n):
5 4
请输入每张信封最多允许贴的邮票数量(m):

邮票面值分别为: 1 3 11 15 32

最大邮资区间为: 1-70
```

5-1

traceback 函数，它是一个递归函数，用于回溯搜索所有可能的邮票面值组合。

在最坏的情况下，traceback 函数的时间复杂度是 O(n * m * maxt)，其中 n 是邮票的种类数，m 是每张信封上允许贴的最多邮票数，maxt 是当前邮票面值组合可以达到的最大邮资。这是因为对于每一种邮票，我们都需要遍历所有可能的面值（从 temp[cur] + 1 到 maxt + 1），并对每种可能的面值，递归确定接下来的面值。

空间复杂度是 O(n)，这是因为我们需要一个长度为 n 的向量 temp 来存储当前的邮票面值组合，以及一个长度为 n 的向量 result 来存储最优的邮票面值组合。

需要注意的是，这个复杂度分析是基于最坏情况的，实际上在运行过程中，由于有剪枝操

作（即如果 max1 不大于 maxt，则不会继续递归），实际的运行时间可能会小于最坏情况的时间复杂度。

## 六、实验总结

连续邮资问题是一个典型的组合优化问题，通过回溯法可以有效地找到给定邮票数量和邮票数量限制下的最佳邮票面值设计。通过此次实验，我理解了如何利用回溯法，承担一部分剪枝策略