北京科技大学实验报告

学院： 计算机与通信工程学院 专业： 计算机科学与技术 班级： 计 184

姓名： 王丹琳 学号： 41824179 实验日期： 2020 年 12 月 9 日

1. 实验名称：

上机实验-1 投骰子

2. 实验目的：

小花在玩一款桌游，规则为玩家根据投出的骰子点数来决定走的步数，即骰子点数为 1 时可以走一步，点数为 2 时可以走两步，点数为 n 时可以走 n 步，桌游所用骰子为 20 面骰。桌游中的每个格子都有或多或少的奖励和惩罚，小花想知道要走到第 n 步 (n ≤ 骰子最大点数且投骰子的方法唯一) 奖励格总共有多少种投骰子的方法。

3. 实验环境：

编程语言：C++，环境：计蒜客

4. 实验原理：

在本题中，要想知道走到第 n 步的投骰子方法个数就必须得依靠走到第 n-1 步的投骰子方法个数，第 n-2 步的投骰子方法个数……来计算得出。这与递归思想十分契合。只需要找出递归关系中规模最小的解，其他的解都迎刃而解。所以，本题主要涉及的知识点为递归，关键为寻找递归，找出递归方程式与找出递归中止条件。

5. 实验内容与步骤：

5.1 问题分析

根据题意可以发现，这与老师课上所讲的整数分划问题相似，课上的例题是求一个正整数写成正整数之和的表达式个数。

故进行类比，对于本题，可将本题抽象为输入一个数，假设为 n ，计算可以从 0，1，2…，n 中选（可重复），加起来得到 n 的方法个数。

解法：假设走到第 n 步共 ff(n)种方法：

ff(1)=1//投骰子直接投出 1

ff(2)=ff(1)+1//先投出 1，再投出 1 或者直接投出 2

ff(3)=ff(1)+ff(2)+1//先投出 1，再投 2 或者先投出 2（上面所求得的方法数）再投出 1 或者直接投出 3

由此可以看出递归方程式为 ff(n)=ff(n-1)+ff(n-2)+ff(n-3)+…..+ff(1)+1；边界为 ff(1)=1

归纳推理可得：

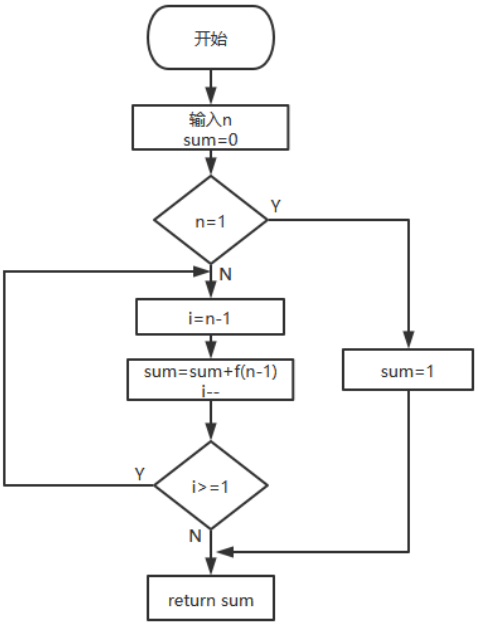
$$ff(n)=\begin{cases}1 & ,n=1\\ 1+\sum_{i=1}^{n-1}ff(i) & ,n>1\end{cases}$$

其中 n>1 的情况下， 1 表示直接投出 n(因为题目中提到 n 小于骰子最大点数)

5.2 算法设计

```
if (n == 1) num = 1; //递归退出条件
else if (n >= 1) {
    sum = 0;
    for (int i = n - 1; i >= 1; i--) {
        sum += ff(i);
    }
    num = sum + 1;//直接抛出来的
```

5.3 算法流程图



5.4 算法实现

```
int ff(int n) {
    if (n == 1) //递归退出条件
        num = 1;

    else if (n >= 1) {
        int sum = 0;
        for (int i = n - 1; i >= 1; i--) {
            sum += ff(i);
        }
        num = sum + 1;//直接抛出来的
    }
    return num;
}
```

5.5 实验测试方案

实验测试方案：

输入 6，即走到第 6 步总共有多少种投骰子的方法，经过计算可以得出是 32

输入 1，即走到第 1 步总共有多少种投骰子的方法，经过计算可以得出是 1

输入 2，即走到第 2 步总共有多少种投骰子的方法，经过计算可以得出是 2

输入 3，即走到第 3 步总共有多少种投骰子的方法，经过计算可以得出是 4

实验测试数据：

输入 6，预期输出为 32

输入 1，预期输出为 1

输入 2，预期输出为 2

输入 3，预期输出为 4

5.6 实验数据：

样例输入

```
6
```

样例输出

```
32
```

5.7 实验数据处理：

输入 1，即走到第 1 步总共有 1 种投骰子的方法（即直接 1）

输入 2，即走到第 2 步总共有 2 种投骰子的方法（即直接 2 或者两次 1）

输入 3，即走到第 3 步总共有 4 种投骰子的方法（即直接 3 或者先 1 后 2 或者先 2 后 1 或者三次 1）

6. 实验结果与分析

6.1 测评结果

计蒜客测评结果

线下模拟测试结果

6.2 实验结果和算法分析

对于测试结果的分析：测试结果符合预期

时间复杂度和空间空间复杂度计算：

空间复杂度也为 $O(n^2)$

时间复杂度为 $O(n^2)$：$ff(n)=\sum_{i=0}^{n-1} ff\ (i)$

$ff(n+1)=\sum_{i=0}^{n} ff\ (i)$

$ff(n+1)=ff(n)+\sum_{i=0}^{n-1} ff\ (i)=2ff(n)$

所以时间复杂度为 O(n2)

7. 实验总结：

通过这次实验，我深刻理解了递归算法的思想与设计要点，递归只需要少量的步骤就可以描述出解题过程中所需要的多次重复计算，减少了算法的代码量，并且可读性比较高。

同时，此题与课堂例题类似，将课上所学灵活运用，举一反三，让我对递归有更深入的认识。 在做题的过程中，我也熟悉了用递归算法解题的三个步骤：分析问题、找出大规模问题与小规模问题的关系，得出让问题规模逐渐变小的递归方程式；设置边界、控制递归，找出算法可解的最小规模问题；设计函数、确定参数。

北京科技大学实验报告

学院： 计算机与通信工程学院 专业： 计算机科学与技术 班级： 计 184

姓名： 王丹琳 学号： 41824179 实验日期： 2020 年 12 月 9 日

1. 实验名称：

上机实验-1 带子的裁剪问题

2. 实验目的：

有一条长度为 N 的带子，每单位长度上都标有一个正整数，现要将其剪成 M(M≤N) 段，要求不能从数字中间剪开，即剪完后的每段长度必须为整数，且每段连续。请问如何剪能使得每段中数字之和的最大值最小？

关于最大值最小：

例如一条带子 4,2,4,5,1 要剪成 3 段，将其如下分段：[4,2][4,5][1]

第一段和为 6，第 2 段和为 9，第 3 段和为 1，和最大值为 9。

将其如下分段：[4][2,4][5,1]

第一段和为 4，第 2 段和为 6，第 3 段和为 6，和最大值为 6。

并且无论如何分段，最大值不会小于 6。

所以可以得到：将带子 4,2,4,5,1 剪成 3 段，每段和的最大值最小为 6。

3. 实验环境：

编程语言：C++，环境：计蒜客

4. 实验原理：

可将本题抽象为求最小 M 段子和问题。在本题中，要想如何剪能使得每段中数字之和的最大值最小，可以把问题转换为找出最小的数，使得按此数来划分段使段数正好为题目要求的 M 段。这与按值二分的思想颇为相似。只需要在 0 到 N 个数之和之间进行二分，就可以找到 M 段和最小的解。所以，本题主要涉及的知识点为二分，本题关键为能将思路进行转换。

5. 实验内容与步骤：

5.1 问题分析

在看到这道题目的时候，我首先根据题意可以发现转换思路：就是数据按顺序装入桶中，M（段数）即是给定的桶数，问题就是求解桶的容量至少应该为多少才能恰好把这些数装入 m 个桶中。

问题转换为求桶的最小容量；

考虑最简单的情况，假设有 1 个数，只有一个桶，即 M=1：至少需要容量为该数的值；

如果 n 个数，只有一个桶，即 M=1：至少需要容量为 n 个数之和；

如果 2 个数，两个桶，即 M=2：至少需要容量为两数之间最大值

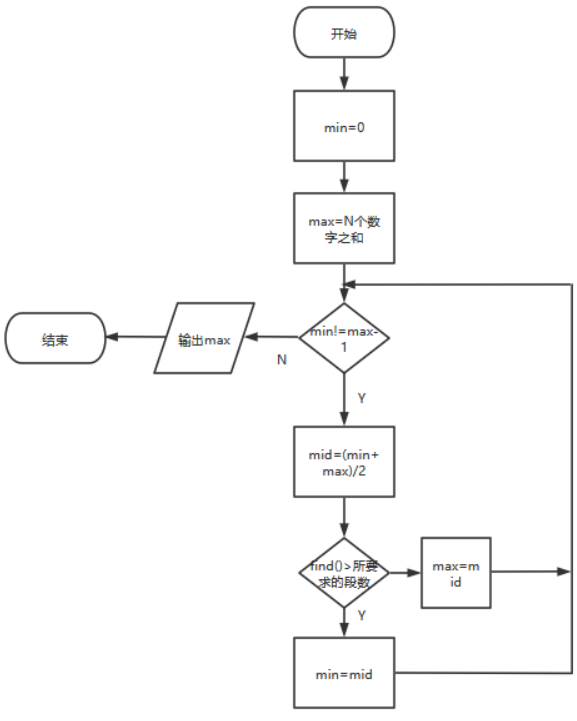
如果 n 个数，M 个桶,，则按值二分，这就与老师课上所讲的那道按值二分相似，课上的例题是找出 1-n 之间唯一的重复数。思想相似。

故进行类比，对于本题，若假设的桶大小拿去装数导致所需桶数>M，说明桶太小了，则在右边进行二分；若假设的桶大小拿去装数导致所需桶数<M，说明桶太大了，则在左边进行二分。

5.2 算法设计

```
//二分过程
min = 0;
max = sum;
while (min != max - 1)//值的二分
{
    int mid = (min + max) / 2;
    if (find(a, n, mid) > m)//大于所要求的M，得放大
        min = mid;
    else            //find(a,n,mid) <= m 得缩小
        max = mid;
}
return max;
```

5.3 算法流程图



5.4 算法实现

```
//遍历得出按照所设桶大小所需要的桶个数
int count = 1;
int sum = 0;
for (int i = 0; i < n; i++)
{
    if (a[i] > lim)//太小了，直接返回
        return INT_MAX;
    else
    {
        sum += a[i];
        if (sum > lim)//重新开一个
        {
            sum = a[i];
            count++;
        }
    }
}
```

5.5 实验测试方案

实验测试方案：

输入 5 3，即有 5 个数（分别为 4 2 4 5 1）分三段，经过实验可以得出[4][2,4][5,1]最小（第一段和为 4，第 2 段和为 6，第 3 段和为 6，和最大值为 6。并且无

论如何分段，最大值不会小于 6。）

输出 6

实验测试数据：

输入 5 3

4 2 4 5 1

预期输出为 6

5.6 实验数据：

样例输入复制

```
5 3
4 2 4 5 1
```

样例输出复制

```
6
```

5.7 实验数据处理：

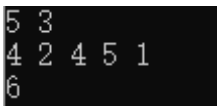
输入 5 3，即有 5 个数（4 2 4 5 1）分三段，经过试验可以得出[4][2,4][5,1]最小（第一段和为 4，第 2 段和为 6，第 3 段和为 6，和最大值为 6。并且无论如何分段，最大值不会小于 6。）

6. 实验结果与分析

6.1 测评结果

计蒜客测评结果

线下模拟测试结果



6.2 实验结果和算法分析

对于测试结果的分析：测试结果符合预期

时间复杂度和空间空间复杂度计算：$O(Nlog(\sum n))$，其中 $\sum n$为给定数组的元素和。

由于没有额外开数组，所以空间复杂度：$O(\mathrm{n})$

7. 实验总结：

通过这次实验，我深刻理解了二分算法的思想，二分法的使用并不一定局限于以下标进行二分，还可以按值二分。本题就是要寻找一个分界点，这恰恰也就是二分法的一个应用，此题帮助我更加深刻的理解了二分法的本质。

同时，此题与课堂例题类似，将课上所学灵活运用，举一反三，让我对课堂例题和按值二分算法有了更深入的认识。 在做题的过程中，我也熟悉了用二分法解题的基本逻辑，即左右边界的确定，最终得出临界点。