**组合数学编程作业**

----同济大学 软件学院 15级2班 王一同 1552661

**目录**

[一、项目使用说明 3](#_Toc25725)

1、项目简介.............................................................................................................................3

2、项目功能要求.....................................................................................................................3

[3、操作手册 3](#_Toc16089)

[1)输入总人数 3](#_Toc11819)

[2)显示结果.......................................................................................................................3](#_Toc11819)

[3)容错操作.......................................................................................................................3](#_Toc11819)

[4)退出 4](#_Toc11847)

[4、 注意事项 4](#_Toc31016)

[二、 概述 4](#_Toc16663)

[1、 基本思路 4](#_Toc29633)

[2、 文件目录 4](#_Toc28624)

[三、 具体实现 5](#_Toc7356)

[1、购票总人数和对应排队方案数表 5](#_Toc32501)

2、 关于n和用时统计的对应表..............................................................................................5

1. 算法1..................................................................................................................................6
2. 算法2..................................................................................................................................6
3. 算法3..................................................................................................................................7
4. 算法4..................................................................................................................................8
5. 算法5..................................................................................................................................8

四、总结...........................................................................................................................................9

**一、使用说明**

1、项目简介：

农夫John和他的朋友们一同去参加展览会。展览会的门票为$50。John发现一个奇怪的现象:在排队购票的2n个人中，总有n个人拿的是面值为$100的钞票，而另外的n个人拿的是面值为$50的钞票。农夫John想知道的是在这种情况下这2n个人共有多少种排队方式，使售票处不至出现找不开钱的局面(假设售票处原来没有零钱的情况)?

2、项目功能要求：

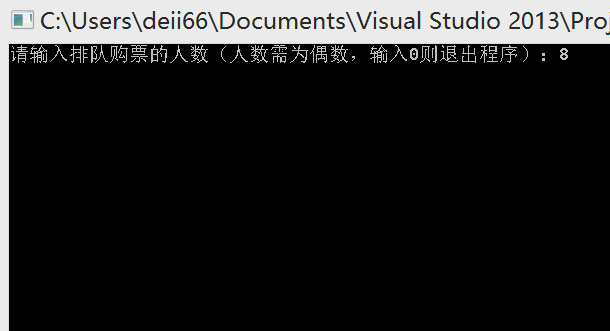
利用5种不同的算法实现该问题的求解，算法1搜索策略，算法2栈模型，算法3递归算法，算法4递推算法，算法5组合算法。

每一个算法都设计一个计数器，记录程序执行时间。最终形成一张关于n和用时统计的对应表。

3、操作手册：

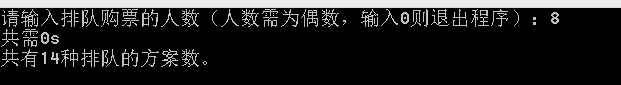
### 输入总人数

首先要输入排队购票的总人数，即题目中的2n：



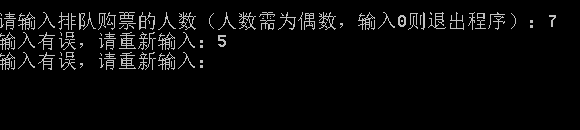
1. **显示结果**

在输入总人数后，程序自动显示该算法所需时间以及当前排队的方案数。



1. **容错操作**

由题可知，人数必须为偶数，若输入奇数，则程序会要求重新输入直至输入正确：



### 4)退出

输入0退出程序。

4、注意事项：

\*由题可知，用户输入的排队购票人数必须为偶数。

\*用户必须输入数字，不能输入其他字符。

\*由于算法时间较慢，前三个算法建议输入总人数小于30。

1. **概述**
2. 基本思路：

算法1搜索策略使用回溯和递归的方式来实现。算法2栈模型模拟了栈的出入来得到算法的解。算法3利用公式进行递归。算法4利用公式进行递推。算法5利用现有的公式直接求解。

1. 文件目录：

1552661\_王一同\_1.cpp (算法1文件)

1552661\_王一同\_2.cpp (算法2文件)

1552661\_王一同\_3.cpp (算法3文件)

1552661\_王一同\_4.cpp (算法4文件)

1552661\_王一同\_5.cpp (算法5文件)

1552661\_王一同\_1.exe (算法1程序可执行文件)

1552661\_王一同\_2.exe (算法2程序可执行文件)

1552661\_王一同\_3.exe (算法3程序可执行文件)

1552661\_王一同\_4.exe (算法4程序可执行文件)

1552661\_王一同\_5.exe (算法5程序可执行文件)

1552661\_王一同.doc (项目文档)

1. **具体实现**

1、购票总人数和对应排队方案数表：

|  |  |
| --- | --- |
| 购票总人数 | 排队方案数 |
| 2 | 1 |
| 4 | 2 |
| 8 | 14 |
| 10 | 42 |
| 16 | 1430 |
| 20 | 16796 |
| 26 | 742900 |
| 30 | 9694845 |

1. 关于n和用时统计的对应表（单位为s）：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 算法  n | T1（n） | T2（n） | T3（n） | T4（n） | T5（n） |
| 4 | <0.0001 | 0.001 | <0.0001 | <0.0001 | <0.0001 |
| 5 | <0.0001 | 0.059 | <0.0001 | <0.0001 | <0.0001 |
| 7 | <0.0001 | 4.469 | <0.0001 | <0.0001 | <0.0001 |
| 9 | 0.003 | >60 | <0.0001 | <0.0001 | <0.0001 |
| 11 | 0.034 | >60 | 0.003 | <0.0001 | <0.0001 |
| 13 | 0.493 | >60 | 0.046 | <0.0001 | <0.0001 |
| 14 | 1.927 | >60 | 0.169 | <0.0001 | <0.0001 |
| 15 | 7.465 | >60 | 0.634 | <0.0001 | <0.0001 |

1. 算法1：

（1）基本思路：

我们用回溯法来直观地枚举所有情况。算法中指定一变量k记录售票处有$50钞票的张数，初始时令k=0，若某人手持$100钞票且k=0时则回溯，否则继续递归。若第2n个人购完票即递归进行到第2n层时计数器累加1。递归结束后，计数器中的值便为排队方案总数。

（2）具体实现：

利用函数void search(int count, int &k, int &ans, int farmer)进行递归，其中，count指的是算法执行到该步骤时当前的排队人数，k为思路中的k，即$50钞票的张数，ans为排队方案数，farmer为总人数。起始时前三个数据均为0。

每次递归先判断当前人数是否达到总人数，如果当前人数与总人数相同，且此时k为0，由于$50和$100的人数相同，证明$50均被用完，该种排队方式可行，ans+1，否则回溯到上一层。

如果当前人数未达到总人数，则对k进判断，k不为0，则证明售票处有$50，可以使用$100，所以k-1，之后对当前人数加一继续递归。如果k为0，则$50已用完，k需要增加1，并继续向下递归。每次递归返回上一层后k-1。

（3）分析：

该算法的实质是模拟，建立了一棵n层的解答树，除最后一层外每层都向下扩展了2个子节点，该算法时间复杂度较高，为指数级，限制了n的大小。

1. 算法2：
2. 基本思路：

算法2使用栈模型的算法，算法先产共n个数的全排列，对于每种排列，若符合出栈规则，那么这个排列便是一个可能的出栈序列，计数器加1，当n个元素的全排列列举结束时，计数器的值便是问题的解。

1. 进出栈规则：

若是手持$50的人来购票，则将数字1入栈，若是手持$100的人来购票，则将栈顶数据出栈。则出入栈的过程中不存在栈下溢的情况。

1. 具体实现：

首先要生成n个数的全排列，通过函数void print(int l, int r, int n)来实现，l为当前人数，r和n为思路中的n，该函数生成了n个数的全排列。

之后通过函数bool test\_stack(int n)来模拟出入栈的顺序。如果test不等于a[i]中的值，意味着$50，则入栈，如果栈不为空，且栈顶等于a[i]，则是$100，出栈。

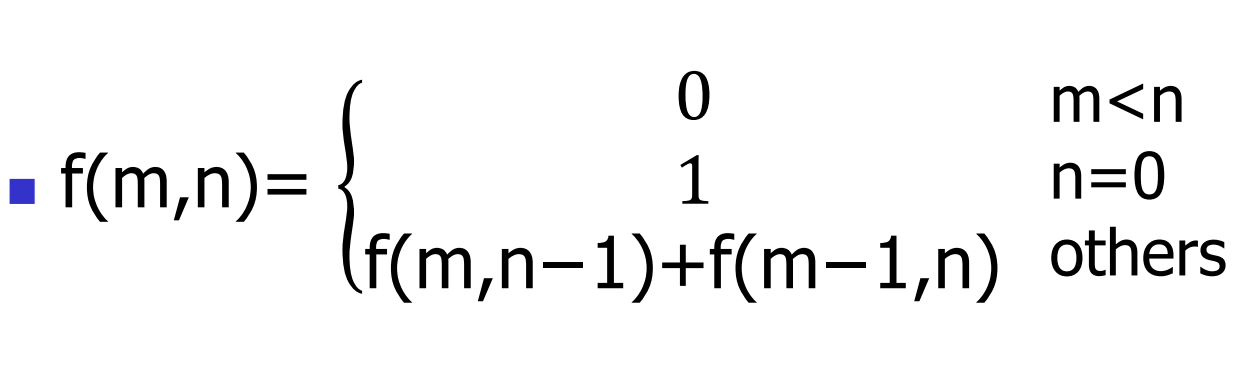
1. 分析：

该算法与第一个算法相比，递归深度更低，栈的使用空间不大，然而在扩展解答树的时候，该方法的节点大量增加，许多节点没有意义，使该算法的时间大大增加。实际上可以设计一种算法一边生成全排列一边对输出序列进行判断，不满足条件的即使删除该节点。

5、算法3：

（1）基本思路：

前两个算法都涉及到解答树的节点过多问题，这样的问题会影响到算法的规模大小。算法3着重在在数学方面上进行求解，通过下面的公式进行递归求解。



令f(m,n)中的m人手持$50的钞票，n个人手持$100的钞票时共有的方案总数。我们分情况来讨论这个问题。

(1) 当n=0时，n=0意味着排队购票的所有人手中拿的都是$50的钱币，那么这m个人的排队方案总数为1。

(2)当m<n时

若排队购票的(m+n)个人中有m个人手持$50的钞票，n个人手持$100的钞票，当m<n时，即使把m张$50的钞票都找出去，仍会出现找不开钱的局面，所以这时排队总数为0。

 (3)其他情况

 我们思考(m+n)个人排队购票的情景，第(m+n)个人站在第(m+n-1)个人的后面，则第(m+n)个人的排队方式可由以下两种情况获得:

① (m+n)个人手持$100的钞票，则在他之前的(m+n-1)个人中有m个人手持$50的钞票，有(n-1)个人手持$100的钞票，此种情况共有f(m,n-1)。

 ②第(m+n}个人手持$50的钞票，则在他之前的(m+n-1)个人中有(m-1)个人手持$50的钞票，有n个人手持$100的钞票，此种情况共有f(m-1,n)。

（2）具体实现：

递归算法是由终止条件向初始条件推导。利用函数void recursion\_search(int m, int n)进行递归。其实参数均为n（此处的n为题目中排队总人数的一半），函数参数m和n为上述思路中的m和n所代表含义。在递归前进行判断，若m<n，为所有的$50找出去也存在无法找开的情况，直接返回上一层。若n==0，则该方案可行，方案数加一。其他情况，分别进行判断和递归。

1. 分析：

该算法是从数学角度进行求解，相较于前2个算法，时间有所缩短，问题的规模也得到了扩大。

1. 算法4：
2. 基本思路：

递归算法是由终止条件向初始条件推导，而递推算法是由初始条件向终止条件推导。可以说，它们本质上是相同的。递推算法的时间复杂度为O（n²），它与递归算法比较最大的优点在于充分利用了已经得到的信息，从而使算法的时间复杂度大大降低，算法本身能够接受的规模也大大增加。

1. 具体实现：

利用函数void recursion\_search()实现递推，首先建立一个二维数组作为f(m,n)的容器，行数小于列数的时候，对应的f(m,n)=0，第一列为0时，对应着f(m,n)=m，其他情况则是进行递推可得到。

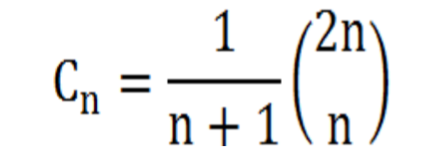
1. 分析：

在得到算法3中的公式后，同样可以将其用于递推，只是与算法3递归算法的方向不同，递推是由初始条件向终止条件推到，这样避免了大量的数据冗余，有利于扩大算法规模。

1. 算法5：
2. 基本思路：

下而用一种崭新的模型——二叉树来反映本题。我们依据下面的原则将具有n个结点的二叉树的顶点标号。若结点i是结点j的子结点，则i>j 若结点i是结点k的左儿子，结点j是结点k的右儿子，则i<j。

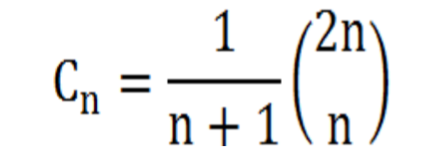
此时，这棵树最左边的叶结点一定是中序遍历次序的第一个顶点，最右边的叶结点一定是中序遍历次序的最后一个顶点。所以，对于任意一棵具有n个结点的二叉树，它的前序遍历顺序便为1~n，即n个元素的入栈顺序，它的中序遍历顺序便是这n个元素的出栈顺序。即2n个人的排队方式总数即为具有n个结点的二叉树的个数，又因为具有n个结点的二叉树个数为



所以该公式即为本题的解。

1. 具体实现：

由基本思路可以得到以下公式：



只需通过公式即可得到答案。

1. 分析：

该算法是所有算法中最抽象的，很难理解，但这个算法抓到了问题的本质，所以效率很高。

1. **总结**

通过对该问题的5种算法求解，我首先懂得了对应题目求解，利用抽象数学方式更加有效，不会严格限制问题的规模，并且相较于递归，递推的效率更高。5个算法中最简单最高效的是最后一个，可以直接靠公式得出，但公式的推到很难理解。对于我本人来说，可能会通常采用第一种方法解题，但通过这次作业，我会考虑在程序中加入递归，尤其是递推的方式，对问题的求解更加高效。关于算法2模拟栈的出入，效率并不是很高，但是对于我来说也是学到了一种换方式思考问题的思考方式。