**周报**

本周学习了《零基础学算法》的第八、第九章。第八章学习的是算法的经典问题，第九章学习的是信息学奥赛试题精讲。学习的相关内容简单总结如下：

1. 第八章——算法经典问题
2. 不定方程问题
3. 百钱买白鸡
4. 存钱利息最大化：使用穷举法求解
5. 求阶梯数：使用穷举法
6. 五家共井：可使用枚举法
7. 鸡兔同笼：使用穷举法
8. 推算问题
9. 猴子吃桃：使用递归实现
10. 舍罕王的赏赐
11. 魔术方阵：（阶数必须为4的倍数）将连续的整数按某种特别的顺序排在方阵里，每一行、每一列或者对角线位置的数各自相加的和均相等。
12. 简捷连续填数法：1立首行中，右1上1，受阻下1。
13. 双向反转法：
    1. 将数字由左向右、由上到下填入方阵
    2. 将方阵中间部分半行数的数字左右翻转
    3. 将方阵中间部分半数列的数字上下翻转
14. 井字调整法：在使用该方法时，为了便于识别，在方阵中用井字字形作为标记，因此该方法称为井字调整法（只适用于4K+2阶魔术方阵的生成）

实现的步骤：

* 1. 将数字由左向右，由上到下顺序填入方阵，然后在第K+1、3K+2行及列做井字标记。
  2. 将标记范围内的行做水平翻转，再将列做垂直翻转（井字标记行上的数字忽略）。
  3. 将井字分隔线的行及第K+2行两侧的数字左右对调，两行中央测数据上下对调，左边列的数字除交叉点 外进行垂直翻转。
  4. 将井字分隔线列的纵向中央的数字除第2k+1列外左右对调，井字橫行左方的第一个数字上下对调，上横线中央的数字水平翻转。

1. 智力趣题
2. 汉诺塔
   1. 递归法解汉诺塔：

如果只有一个圆盘，则把该圆盘从A棒移动到C棒上

如果圆盘N>0,则移动圆盘的过程可分为三个步：

第一步：将A棒上的N-1个圆盘移到B棒上；

第二步：将A棒上的一个圆盘移到C棒上；

第三步：将B棒上的N-1个圆盘移到C棒上。

* 1. 非递归法：使用判断N为偶数还是奇数来实现。

1. 背包问题

操作过程：首先创建一个空集合，然后向空集合中增加元素，每个增加元素就先求出该阶段的最优解，直到所有元素都添加到集合中，然后得到的是最优解。

1. 递归法：

实现的过程：

* 1. 定义物品的数据结构：

typedef struct goods{

double \*value; //价值

double \*weight; //重量

char \*selevt; //是否选到方案中

int num; //物品的数量

double limitw; //限制的数量

}GOODS;

* 1. 判断物品i加入到背包中，并判断是否超重，如果没有超重，则标记选中该物品，递归调用该函数继续加入下一个拍物品；如果超重，则把标记记录到一个数组中，并保持当前方案的最大值。
  2. 再判断未排除物品价值是否大于最大值，如果大于最大值，则继续添加物品，如果小于最大值，则直接把当前的最大值输出。

1. 穷举法

对于n个物品使用n层循环，这样就可得到不同各种不同的组合。

每个物品有两种情况，可以使用二进制数来进行模拟。对于n个物品，就可使用n位二进制数来进行模拟。

数据的定义：

Typedef struct goods{

Double \*value; //价值

Double \*weight; //重量

Int num; //物品数量

Int limitw; //限制重量

}

1. 马踏棋盘（非递归、递归、栈）
2. 循环查找：
   1. 将当前步数写入棋盘数组中，接着开始探测下一个应该走的位置，
   2. 马的下一步最多可以有8个个位置可以走，分别测试每个方向是否可以走，将可走的位置作为下一步的位置；
   3. 对下一步可走的位置再进行探测，统计每个位置的下一个可走步数，选出步数最少的位置作为下一步位置。
   4. 重复以上三个步骤，直到将棋盘所有单元格走完。
3. 递归法

思想：从起始点开始向下一个可走的位置走一步，接着以该位置为起始，再向下一个可走的位置走一步，就这样不断的循环，直到走完64个单元格，就找到一个遍历方案。若某个位置的8个方向都没有可走的路，则退回上一步，从一个位置的下一个可走位置继续递归调用。

1. 使用栈实现

思想：跟递归类似，从起点开始，将棋盘中通过的位置放入栈中，然后以该位置为起点，试探从该位置向下走的8个位置，找到一个能走的位置时，又将该位置放入栈中，然后继续向前走，就这样不断重复。若在某个位置试探起8个下一步的位置都不能走，则从栈中弹出前一步，然后试探前一步中8个方向的其他方向。

1. 趣味游戏
2. 取石游戏
3. 生命游戏：在程序中可设置一个二维数组，用来存储细胞的死活状态，通过循环逐个统计网格四周的活细胞数量，根据这个数量决定当前细胞的死活，这时需要使用另一个临时二维数组保存新生成的细胞状态。
4. 洗扑克牌：原理是与随机数排列是相同的。都是将一组数字打乱重新排序。首先逐个生成1~25个不同随机数，将这些数按生成顺序放入数组即可。但是生成的随机数可能有重复，在几段的情况下，可能会出现多次重复生成随机数仍和已有的数据重复，从而导致程序效率大大降低。为了提高程序效率，可以首先申城按顺序排序的52张牌，再生成1~52的一个随机数，按这个随机数从已有的扑克牌中去出一张牌，与当前位置的牌进行交换。这样重复52次即可将所有牌的顺序打乱，达到洗牌的目的。
5. 第九章——信息学奥赛试题精讲（一共有24题）

这一章主要看了决绝问题的思路，对于代码的具体实现过程也大概了解。

**2016-10-23**

**系统控制与信息处理重点实验室**