王欢 学号：**220181499 github:** https://github.com/njustwh2014/data_structure_example

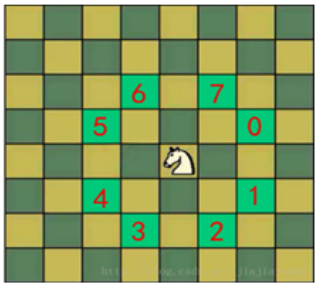
骑士周游问题

Knight_traveled_around.py

Knight_dst.py 解决思考题

骑士周游问题

将马随机放在n × n棋盘的某个方格中，马按走棋规则进行移动。要求每个方格只进入一次。要求走遍棋盘上全部n × n个方格。例如下图中的马下一步可走的位置分布标注为0—7。



进一步思考：你设计的方法能否实现求出任意两个格子之间最少需要跳几步的问题？如果不行，有没有其他的办法？

```
63 58 11 46 25  4  9  6
12 47 64 57 10  7 26  3
59 62 51 48 45 24  5  8
50 13 56 61 52 27  2 23
55 60 49 44  1 34 19 28
14 43 38 53 40 31 22 33
37 54 41 16 35 20 29 18
42 15 36 39 30 17 32 21
```

这是从（4,4）开始的一个可行的周游结果

```
[1, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0]
[0, 2, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 3, 0, 0, 0, 5, 0]
[0, 0, 0, 0, 4, 0, 0, 0]
[0, 0, 0, 0, 0, 6, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 7]
```

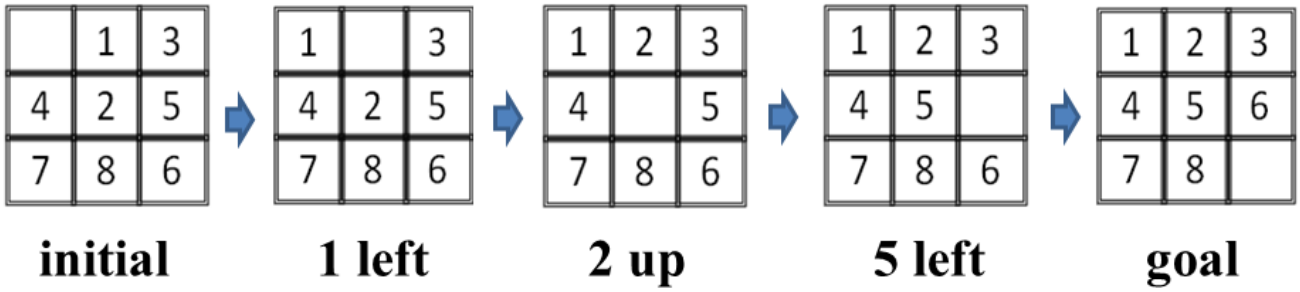
这是从（0,0）到（7,7）的跳马过程，步数最少为6步。

八数码问题

eightnumques.py

八数码问题

8数码问题（8 puzzle）也称九宫难题，是18世纪70年代由Noyes Palmer提出并逐渐流行起来的。它由8个方块和一个空格组成3×3的网格。8个方块随机放置1-8个数字，目标是移动方块使得8个方块满足一个确定的布局要求。例如：

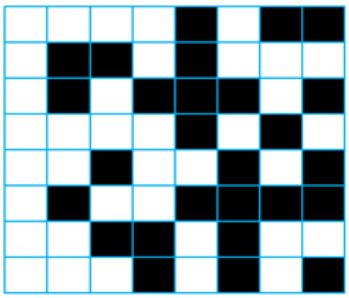


统计黑格子问题

countblack.py

统计黑格子

在一个矩形网格中每一个格子的颜色或者为白色或者为黑色。任意或上、或下、或左、或右相邻同为黑色的格子组成一个家族。家族中所有格子的数量反映家族的大小。要求找出最大家族的家族大小（组成最大家族的格子的数量）。例如下图中最大家族的格子数量为**8**。



进一步思考：你设计的方法能否同时给出哪些黑格子是属于这个最大家族的？

计算24点问题

howequal24.py

计算24点

对于给定1-13的任意四个整数（可重复），判定是否可以通过加、减、乘、除（可重复）运算使得计算结果为24。如果可以则给出其计算表达式或者计算步骤。

例如：若四个数为3, 4, 5, 6，则表达式

(3+5-4)*6

的计算结果为24；计算步骤可以表示为：

//3+5//8-4//4*6

而2, 5, 5, 6这四个数则是无法通过加、减、乘、除运算使得其计算结果为24的。

进一步思考：你设计的方法能否方便地扩展到n(n>4)的情形？

链表重排问题

linklisttest.py

给定一个单链表 L：L0→L1→…→Ln-1→Ln ， 将其重新排列后变为： L0→Ln→L1→Ln-1→L2→Ln-2→…

你不能只是单纯的改变节点内部的值，而是需要实际的进行节点交换。

示例 1:

给定链表 1->2->3->4, 重新排列为 1->4->2->3. 示例 2:

给定链表 1->2->3->4->5, 重新排列为 1->5->2->4->3.

Stack操作

StackTest.py

- 试设计算法，将栈S中的元素排序。要求不用辅助数据结构，仅通过对S自身的操作完成S中元素的排序。PS:我还是利用了两个栈，不符合题意。
- 现有栈S，试设计算法，将S中的元素逆置。要求不用辅助数据结构，仅通过对S自身的操作完成S中元素的逆置。

树、二叉树、满二叉树、完全二叉树

binaryTree.py

自由树

自由树是一个连通的、无回路的无向图。下面表述是等价的。

- G是自由树
- G中任意两个顶点由唯一一条简单路径得到。
- G是连通的，但从E中去掉任何边后得到的图都是非连通的。
- G是无回路的，且|E|=|V|-1。
- G是连通的，且|E|=|V|-1。
- G是无回路的，但添加任何边到E中得到的图包含回路。

二叉树

在计算机科学中，二叉树是每个节点最多有两个子树的树结构。通常子树被称作“左子树”（left subtree）和“右子树”（right subtree）。二叉树的每个结点至多只有二棵子树(不存在度大于2的结点)，二叉树的子树有左右之分，次序不能颠倒。

二叉树的第i层至多有$2^{(i-1)}$个结点；

深度为k的二叉树至多有2^k-1个结点；（等比数列$1+2+4+\cdots+2^{(k-1)} = 2^k-1$）。

对任何一棵二叉树T，如果其终端结点数为n_0，度为2的结点数为n_2，则$n_0 = n_2 + 1$。

树和二叉树的三个主要差别：

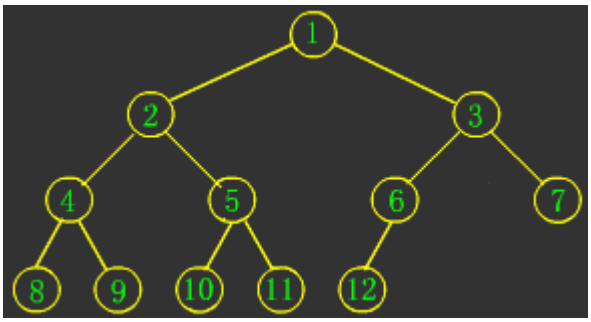
1. 树的结点个数至少为1，而二叉树的结点个数可以为0；
2. 树中结点的最大度数没有限制，而二叉树结点的最大度数为2；
3. 树的结点无左、右之分，而二叉树的结点有左、右之分。

完全二叉树

- 完全二叉树是由满二叉树而引出来的。对于深度为K的，有n个结点的二叉树，当且仅当其每一个结点都与深度为K的满二叉树中编号从1至n的结点一一对应时称之为完全二叉树。
- 若设二叉树的深度为h，除第h层外，其它各层(1～h-1)的结点数都达到最大个数，第h层所有的结点都连续集中在最左边，这就是完全二叉树。

满二叉树一定是完全二叉树，完全二叉树不一定是满二叉树。

下面是完全二叉树的基本形态：



完全二叉树的性质：

1. 深度为k的完全二叉树，至少有$2^{(k-1)}$个节点，至多有2^k-1个节点。
2. 树高$h=log2n + 1$。

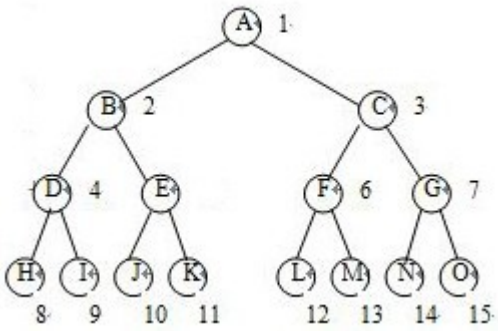
满二叉树

一棵深度为k，且有2^k-1个节点的树是满二叉树。

另一种定义：除了叶结点外每一个结点都有左右子叶且叶子结点都处在最底层的二叉树。

这两种定义是等价的。

从树的外形来看，满二叉树是严格三角形的，大家记住下面的图，它就是满二叉树的标准形态：

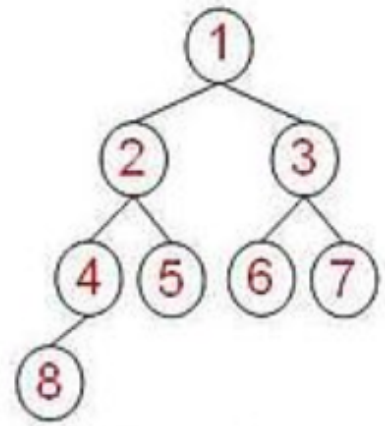


所有内部节点都有两个子节点，最底一层是叶子节点。

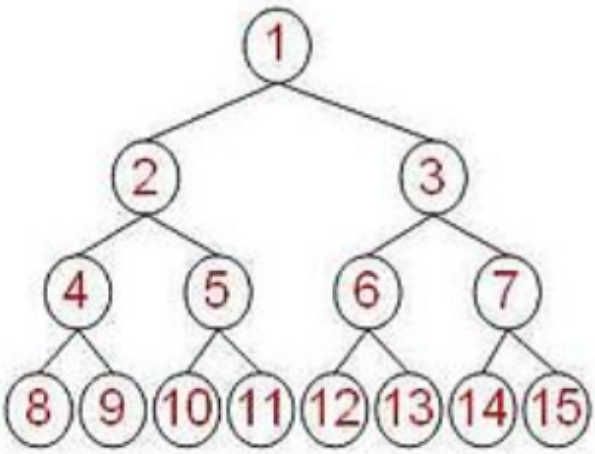
性质：

1. 如果一颗树深度为h，最大层数为k，且深度与最大层数相同，即k=h;
2. 它的叶子数是:$2^{(h-1)}$
3. 第k层的结点数是:$2^{(k-1)}$
4. 总结点数是:2^k-1
5. 总节点数一定是奇数。
6. 树高:$h=log2(n+1)$。

完全二叉树与满二叉树树高计算



(a).Complete Tree



(b). Full Tree

	Complete Binary Tree	Full Binary Tree
总节点k	$2^{h-1} < k < 2^h - 1$	$k = 2^h - 1$
树高h	$h = log_2 k + 1$	$h = log_2(k+1)$

无向图最小生成树的克鲁斯卡尔算法实现

MST.py

参考(https://www.cnblogs.com/yoke/p/6697013.html)