**北京邮电大学软件学院**

**2020-2021学年第一学期实验报告**

**课程名称：** 算法分析与设计

**项目名称： 分支界限法**

**项目完成人：**

**指导教师：**\_\_\_\_\_\_\_\_\_\_\_李朝晖\_\_\_\_\_\_\_\_\_\_\_

**日 期： 2020年 11月 18日**

1. **实验目的**

* 深刻理解并掌握分支限界法的设计思想，比较与回溯法的不同之处；
* 提高应用分支限界法设计算法的技能；
* 理解这样一个观点：好的限界函数不仅计算简单，还要保证最优解在搜索空间中，更重要的是能在搜索的早期对超出目标函数界的结点进行丢弃，减少搜索空间，从而尽快找到问题的最优解。

1. **实验内容**

* 必做：用分支界限法求解0/1背包问题；√
* 必做：用分支界限法求解TSP问题；√
* 选作：用分支界限法求解八数码问题；√

1. **实验环境**

Pycharm IDE、Win10系统

1. **实验结果**
2. ***0/1背包问题***

**测试数据：**

# 构造货物列表 已给数据(可修改)

# Goods = [Good(2, 6), Good(2, 3), Good(6, 5),

#          Good(5, 4), Good(4, 6), Good(7, 10)]

# Goods = [Good(20, 20), Good(15, 30), Good(10, 25)]

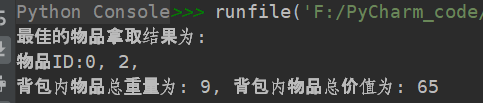
goods = [Good(4, 40), Good(7, 42), Good(5, 25), Good(3, 12)]

# 物品节点数组

good\_nodes = []

backpack\_capacity = 10  # 背包容量, (可修改)

ub\_bound = [0, 0]  # 界限

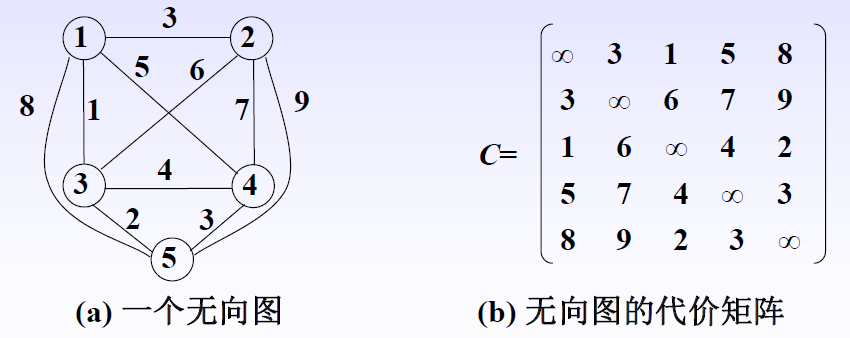
**测试结果：**

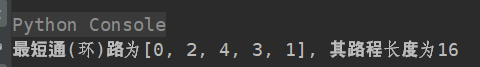
**物品总重量：**5 + 4 = 9 < 10

**物品总价值：**40 + 25 = 65

**时间复杂度：** n为物品数量

1. ***TSP问题***

**测试数据：**

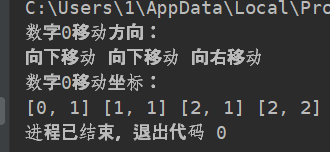
**测试结果：**

**时间复杂度：**其中E为边的个数

1. ***八数码问题***

**测试数据：**

使用数字0代替空白

**测试结果：**

**时间复杂度：**O（9！）

1. **附录**
2. ***0/1背包问题***

* 问题分析

在0-1背包的分支界限算法中，存在如何确定合适的限界函数、如何组织待处理结点表以及如何确定最优解中的各个分量的问题。首先应该解决的是如何组织节点的问题，因为节点中需要存储的信息有此时背包的重量、价值和此时的上界值，所以可以设计一个类来存放节点信息。然后是明确上下界问题，设计一个求解上界的函数，通然后再过计算和比较上界的值来确定在解空间树中下一个作为根节点搜索的节点。

* 初始化界限

首先可以应用贪心法求得近似解，获得的价值可以作为0-1背包问题的下界。之后再考虑最好情况，即背包中装入的全部是价值最大的物品且可以将背包装满，由此得到评估函数的界。需要注意的是，上界的计算实际上是一种十分理想的情况，即假设物体可切割，并且可全部放置这个物品，因此是根据当前背包容量和未被选择的物品中单位质量价值最大的物品来填充的。界限函数公式：ub=v+(W-w)\*(vi+1/wi+1)

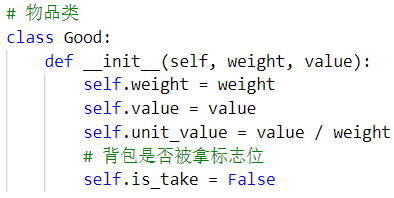
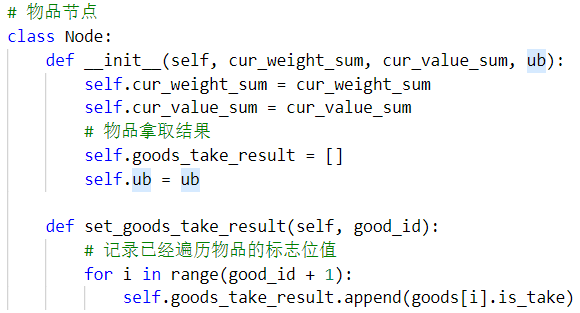
在本测试数据中：



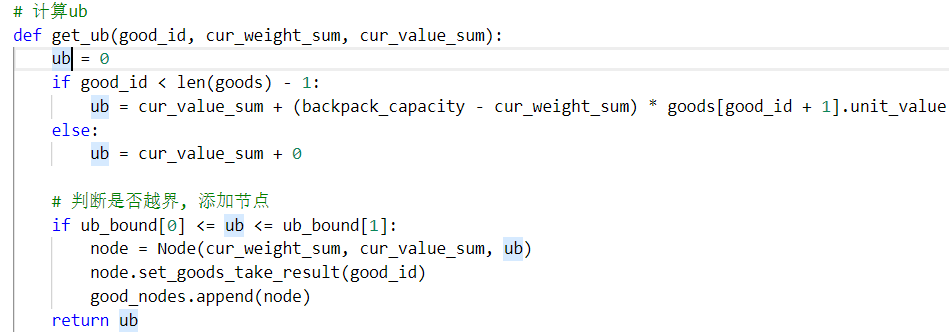
上界，由贪心算法得(1,0,0,0)，价值为40。下界，最好情况：背包中装入的全部是第1个物品且可以将背包装满：ub=W×(v1/w1)=10×10=100。得到评估函数的界[40, 100]。

* 算法讲解

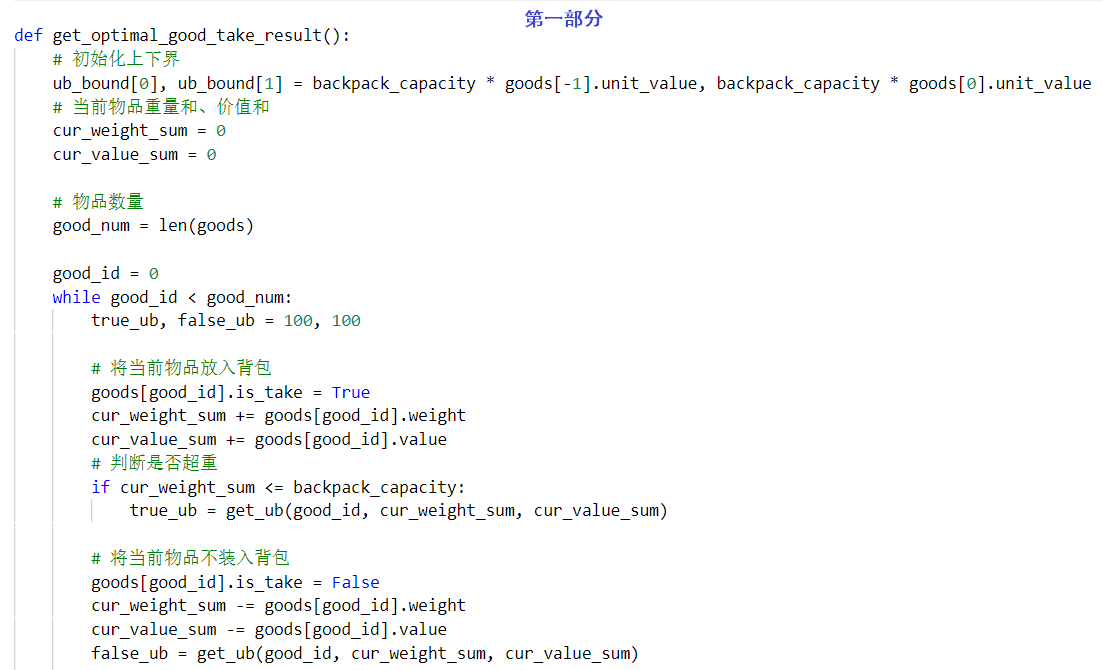
首先构造物品类-记录每个物品的信息，物品节点类-记录当前背包内的物品总重量、总价值、物品拿取情况。

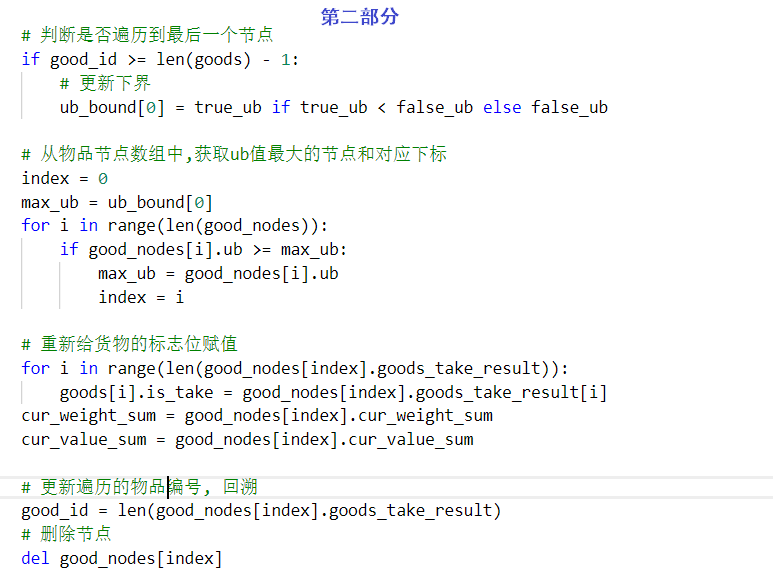


之后构造get\_ub函数计算ub，每当遍历到一个新的物品节点时记录计算当前的ub值。之后将新的节点存储起来，继续遍历下一个。



最后构造get\_optimal\_good\_take\_result()函数，计算价值最大的物品拿取结果，此函数主要分为两部分，在循环内，首先计算出物品拿或不拿的结果，在满足限制条件的情况下，将结果存入到Node中，再将Node存入到good\_nodes中，之后判断当前拿取的物品是否为最后一个，如果是，则更新下界，然后从good\_nodes中获取ub值最大且遍历深度最深的节点，取出该节点对应的物品信息，在该节点物品信息的基础之上继续循环向下遍历，直到遍历的物品编号大于物品数量时，跳出循环输出结果。





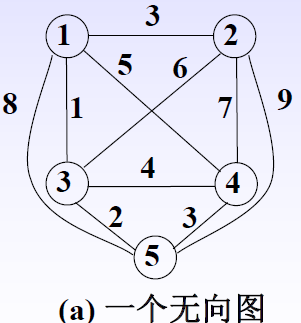
1. ***TSP问题***

* 问题分析

TSP问题是指旅行家要旅行n个城市，要求各个城市经历且仅经历一次然后回到出发城市，并要求所走的路程最短，若要采用分支界限的方法求解，因为在每条路径上，每个城市都有两条邻接边，一条是进入这个城市的，另一条是离开这个城市的，所以可以找出与每个城市连接的两条最小的路径来代表进入和离开的路程，在每次选择下一个城市后，将其路程值重新计算，当的下一个将要经过的城市编号大于城市数量时，程序结束输出结果。

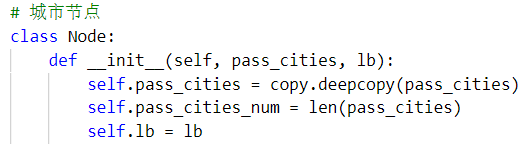
* 初始化界限

采用贪心法求得近似解为1→3→5→4→2→1，其路径长度为1+2+3+7+3=16，这可以作为TSP问题的上界。在每条路径上，每个城市都有两条邻接边，一条是进入这个城市的，另一条是离开这个城市的，那么，如果把矩阵中每一行最小的两个元素相加再除以2，如果图中所有的代价都是整数，再对这个结果向上取整，就得到了一个合理的下界。lb=((1+3)+(3+6)+(1+2)+(3+4)+(2+3))/2=14。于是，得到了目标函数的界[14, 16]

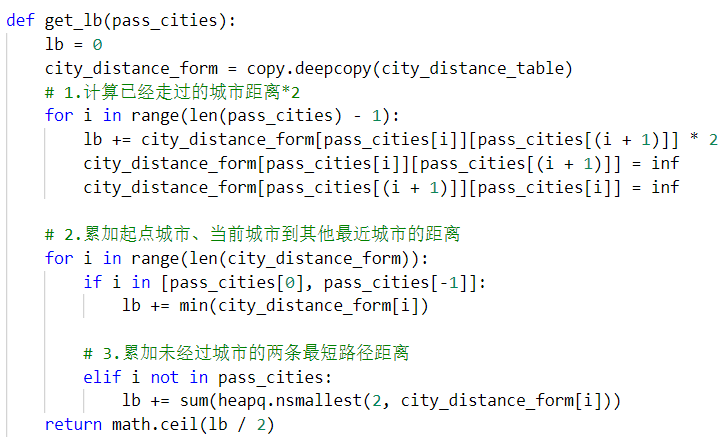


* 算法讲解

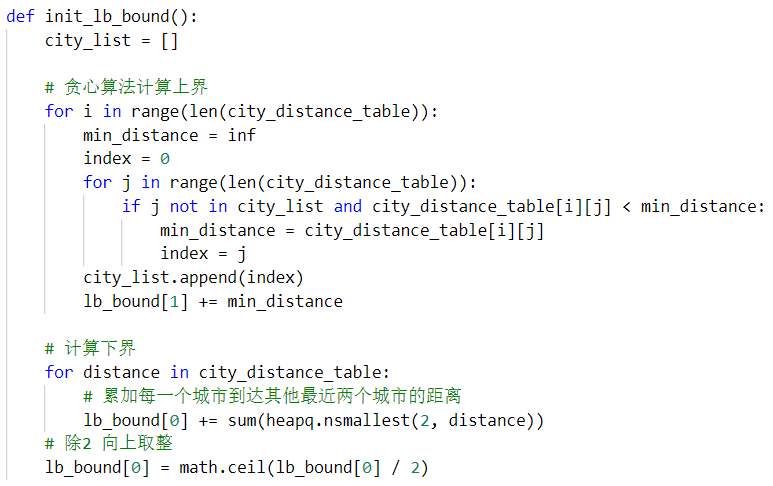
首先设计出城市节点类Node，记录当前经过的城市顺序，城市数量，lb值。



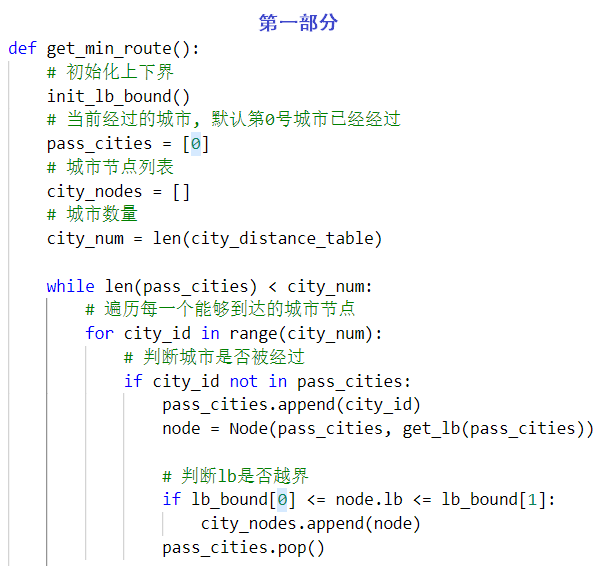
其次构造出计算lb的函数get\_lb(), 函数分为三部分：①计算出已经走过的路程\*2；②计算出起点城市、当前到达城市所能到达其他最近未经过的最近的城市距离之和；③计算出未经过城市的两条最路径径之和，最后将三个计算结果相加除以2得到lb。

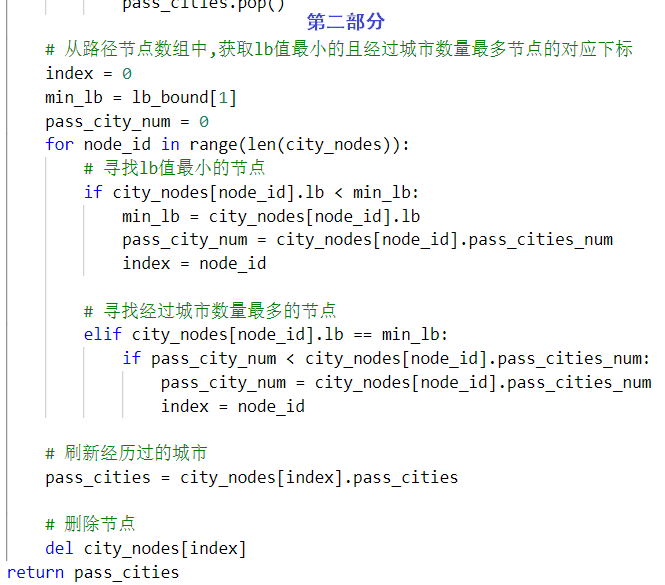


之后实现出lb界限初始化函数，init\_lb\_bound()



最后构造出计算最短路径函数，get\_min\_route()，此函数总共分为两部分，第一部分，数据初始化，从第0号城市出发，使用循环遍历其所能到达的城市，将到达城市后的信息用Node节点存储起来，之后再将节点存放到city\_nodes中。第二部分，从city\_nodes中找出lb最小且经过城市最多的节点，从当前节点状态开始继续向下遍历，当经过的城市数量等于总的城市数量时，跳出循环，输出所获得的最短城市路径。





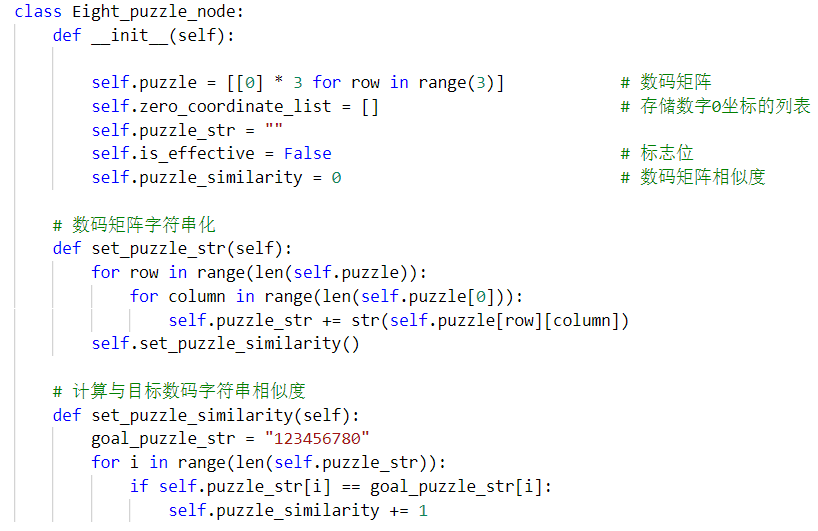
1. ***八数码问题***

* 问题重述   
   八数码问题也称为九宫问题。在3×3的棋盘，摆有八个棋子，每个棋子上标有1至8的某一数字，不同棋子上标的数字不相同。棋盘上还有一个空格，与空格相邻的棋子可以移到空格中。要求解决的问题是：给出一个初始状态和一个目标状态，找出一种从初始转变成目标状态的移动棋子步数最少的移动步骤。所谓问题的一个状态就是棋子在棋盘上的一种摆法。棋子移动后，状态就会发生改变。解八数码问题实际上就是找出从初始状态到达目标状态所经过的一系列中间过渡状态。
* 实现算法思想

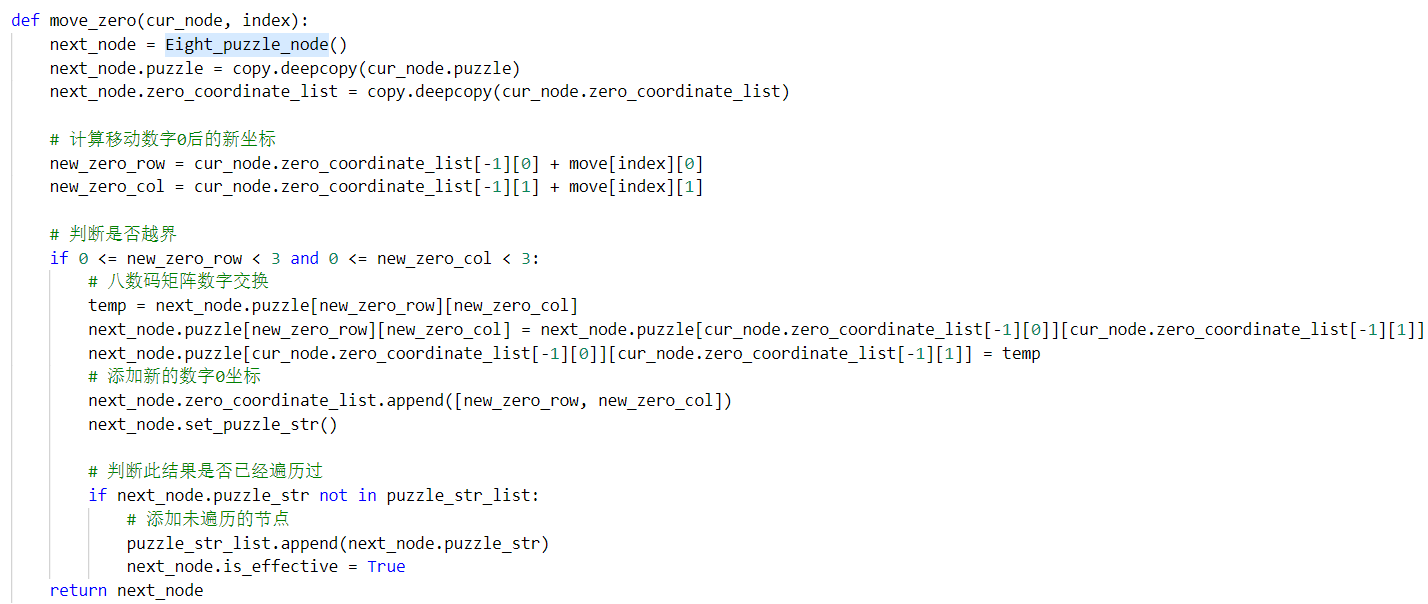
这个问题可以使用bfs加上目标状态相似度的条件限制来计算出解，首先因为它有九个格子，那么便可以用puzzle数组记录他的的九个格子的数值，然后将九个数值转化成一个整数字符串表示其状态，比如我们到达的状态是“123456780”其中空格为0，之后每经过一个状态就将其记录下来，防止重复遍历，之后找出和目标状态相似度最高的节点继续向下遍历。

* 算法讲解

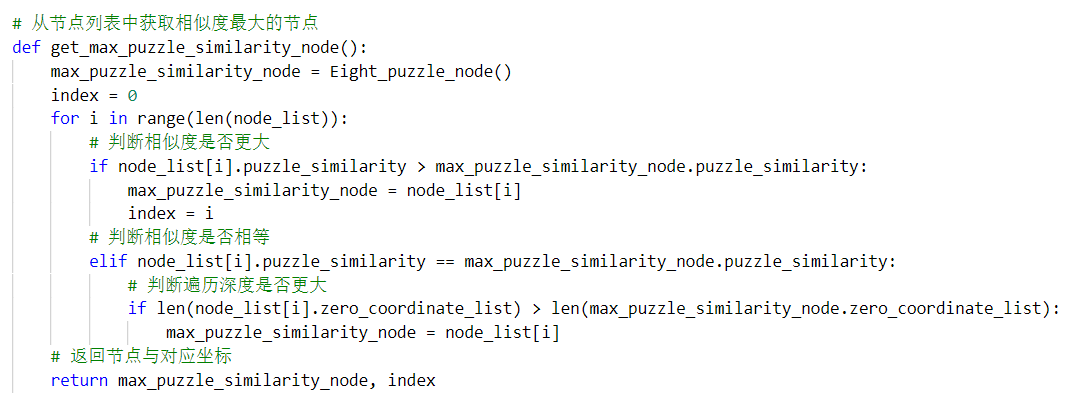
首先构造出八数码节点Eight\_puzzle\_node，用于记录九个格子内的对应数值、数字0在格子内的位置，当前状态与目标状态相似度大小等信息。set\_puzzle\_similarity计算当前状态与目标状态相似度函数，通过将字符串化后的九个宫格数据状态与目标状态对比，每有一个字符相对应，相似度加一。



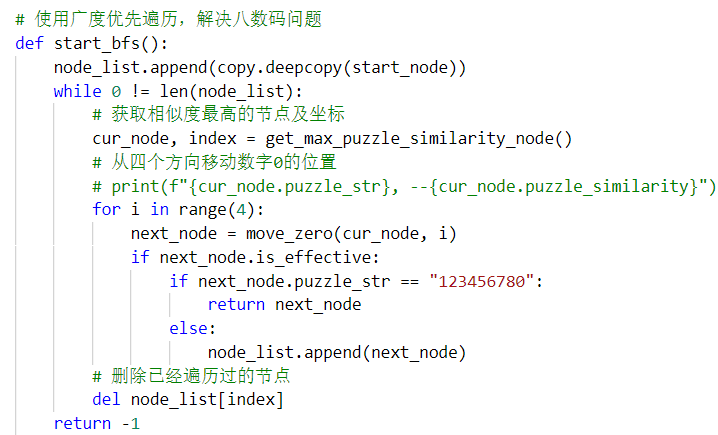
其次构造移动0的函数，move\_zero()此函数返回一个八数码节点，此节点记录了数字0移动之后的九宫格状态，此函数主要分为如下几步，计算数字0移动后的新坐标，判断坐标是否越界，之后判断此状态是否已经遍历过，如果都为否，则将记录到一个node中，返回node。



之后，构造出获取相似度最高节点函数get\_max\_puzzle\_similarity\_node()，从node\_list中依次遍历，找到相似度最高且遍历程度最深的的节点，将其返回。



最后构造出深度优先遍历函数，从初始状态开始，遍历数字0在上下左右移动后的状态，通过move\_zero()获取状态节点，将节点加入到node\_list中，四个方向遍历完后，从node\_list中删除遍历过的节点，调用get\_max\_puzzle\_similarity\_node()函数获取相似度最高的节点，继续循环遍历。当遍历的状态与目标状态相同时，结束遍历，输出从初始状态到目标状态，数字0在九宫格中移动的方向。



1. **调试心得**

通过本次实验，我对0-1背包问题，和TSP问题有了更深的了解，并且对分支界限的思想有了进一步的认识，对循环以及广度优先遍历的使用也更加熟练。