算法

Pratice 4

14130130231

吴天成

1. ***Bellman-Ford algorithm***

Bellman-Ford算法是处理存在权重为负的单源最短路径的算法

给定图G(V, E)（其中V、E分别为图G的顶点集与边集），源点s，node节点的length值记录从源点s到顶点i的路径长度，初始化源点的length为0，其他节点Length为无穷大

以下操作循环执行至多n-1次，n为顶点数：

对于每一条边e(u, v)，如果u.length + w(u, v) < v.length，则另v.length = u.length+w(u, v)。w(u, v)为边e(u,v)的权值；

若上述操作没有对length进行更新，说明最短路径已经查找完毕，或者部分点不可达，跳出循环。否则执行下次循环；

为了检测图中是否存在负环路，即权值之和小于0的环路。对于每一条边e(u, v)，如果存在 u.length + w(u, v) < v.length的边，则图中存在负环路，即是说改图无法求出单源最短路径。否则数组 n.length中记录的就是源点s到各顶点的最短路径长度。

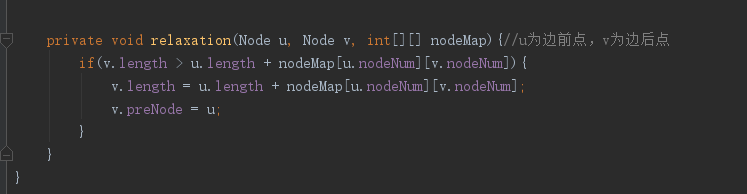
可知，Bellman-Ford算法寻找单源最短路径的时间复杂度为O(V\*E).

代码如下

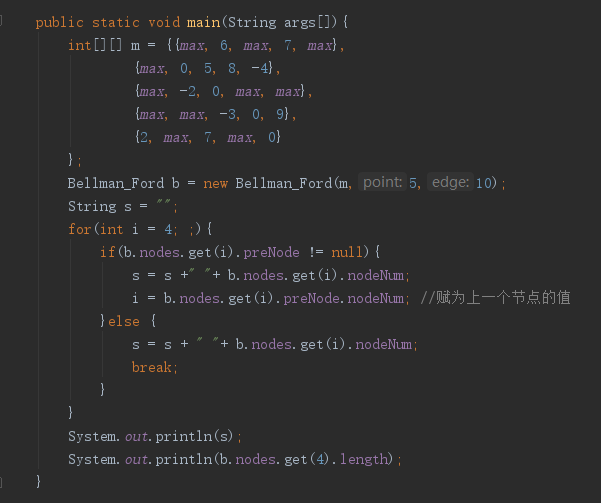


其中max值是Integer的最大值，relaxation方法用于比较u.length + w(u, v) < v.length

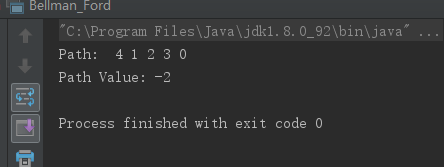
relaxation代码如下



Main函数代码如下



运行结果如图



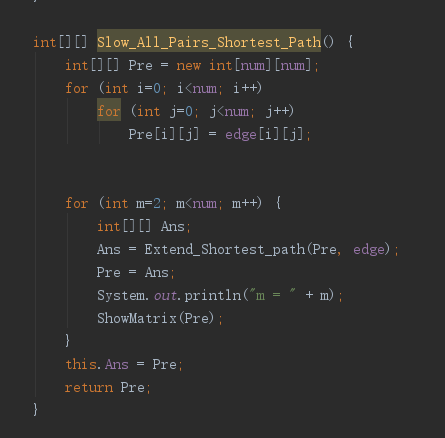
1. ***All-pairs shortest path (choose one from the three algorithms)***

使用书中第一种算法

All-pairs shortest path 为多源最短路径算法，本题采用矩阵乘法解决

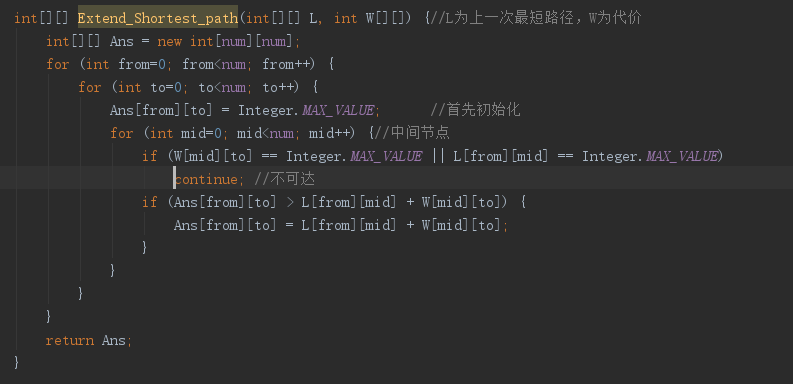
节点数为num，二位数组edge保存的为每条边的代价

主方法代码如下



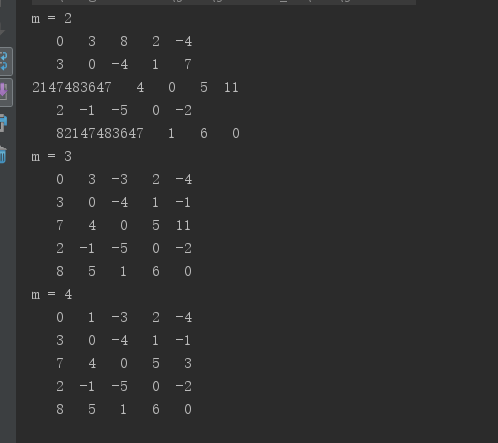
Slow\_All\_Pairs\_Shortest\_Path方法产生 num – 1个Pre数组，第n个Pre[i][j]组表示经过小于等于n跳从节点i到节点j 的最短路径长度。通过方法Extend\_Shortest\_path产生Pre数组

代码如下



该方法比较对于两个节点，如果他们中存在中间节点，则比较他们两个节点原来的最短路径值和经过此中间节点路径值，若加上中间节点路径值更小则更新两节点的最短路径值。

运行程序结果如下

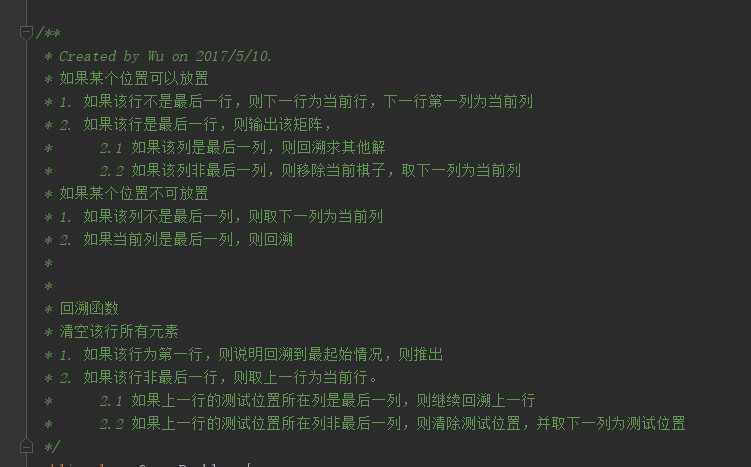


1. ***8-queen problem (back backing)***

八皇后问题

采用回溯法解决

判定条件如下



通过一个二维矩阵来保存棋盘上哪些点放置了皇后

算法的主要方法如下



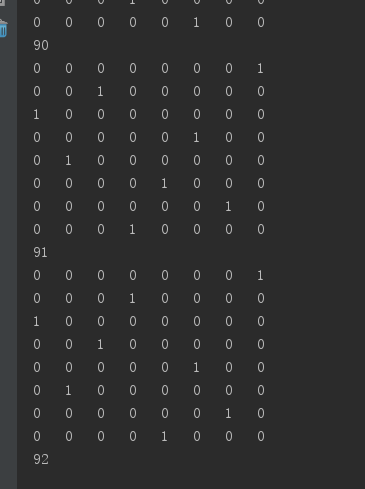
其中back方法是回溯上一行的点并把皇后放到下一列

Remove方法为将此节点的皇后取下

Placechess方法为在此节点放置一个皇后

Canbeplace方法检查此点的行、列、左上、左下、右上、右下对角线是否有其他皇后，有则返回false，否则返回true

运行结果如下



八皇后一共有92种解

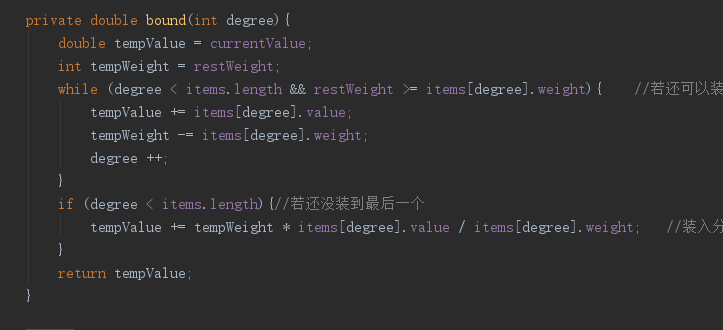
1. ***0-1 knapsack problem (back tracking)***

用回溯法解决01背包问题

对于一个货物数组，首先我们不停的取货物，直到背包装满，然后我们并不知道有些物品该不该拿，但是装货物不需要进行其他的判定，当我们不拿某件货物时，此时我们需要使用bound函数来进行判定，如果不拿某件物品，那之后的价值能否比当前的最大价值高，如果可以，那就选择不拿，如果不可以，那就把树枝砍断，遍历所有叶子结点后，可以得出最终的最优结果。



Bound方法



最终结果

