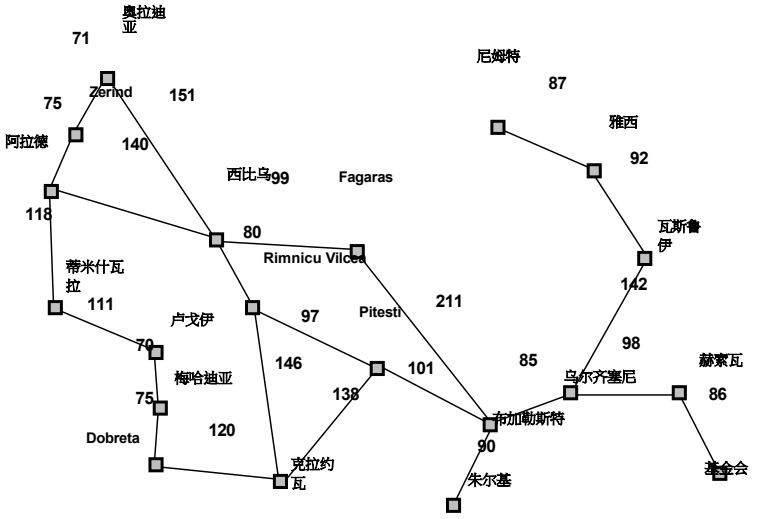
COMP9414: 人工智能教程2：搜索

1. 这个练习是关于寻路问题，以Russell & Norvig (*Artificial Intelligence: A Modern Approach*)中的罗马尼亚地图为例。



到布加勒斯特的直线距离

阿拉德	366
布加勒斯特	0
特-克拉约	160
瓦-多布雷	242
塔-埃弗里	161
-法加拉斯	178
-久尔久	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
尼姆特-奥拉	234
迪亚-皮特什	380
蒂	98
Rimnicu Vilcea	193
Sibiu	253
蒂米什瓦拉	329
乌尔齐塞	80
尼-瓦斯雷	199
-泽林德	374

将寻路问题（从阿拉德到布加勒斯特）定义为一个状态空间搜索问题（给出状态空间的简短描述等，用英语）。当搜索阿拉德和布加勒斯特之间的（最短）路径时，以下每种算法在搜索图中的节点是按什么顺序展开的（给出相关状态）？假设一个状态的继承者是按字母顺序返回的。请确保你理解下面列出的不同算法的关键属性。

澄清一下，对于广度优先搜索，当产生一个具有目标状态的节点时停止搜索，并使用一个检查来确保与先前扩展的节点具有相同状态的节点不被添加到边界。对于其他搜索算法，当目标状态的节点被扩展时停止搜索；对于统一成本搜索，包括检查与先前扩展的节点具有相同状态的节点不被添加到边界（如广度优先搜索）和测试，以便只有一个给定状态的节点被存储在边界上（具有到该状态的最短路径），对于深度优先搜索及其变体，使用沿路径的周期检查以避免可能导致无限分支的重复状态。

(i) 深度优先搜索（有效利用空间，但可能不会终止）。
(ii) 广度优先搜索（空间效率低，保证能找到一个解决方案）
(iii) 统一成本搜索（类似于广度优先，但按成本排列节点）。
(iv) 迭代深化的深度优先搜索（空间效率高，但重复工作）。
(v) 贪婪的最佳优先搜索（高效，不保证最优解）。
(vi) A* 搜索与直线距离启发式（效率低，保证最优解）哪种算法在实践中适合解决这样的寻路问题？

2. 这个版本的地图（来自该书第一版）与该书第二版的地图不同，法加拉斯的启发式数值是178，而不是176，皮特什蒂的启发式数值是98，而不是100（另外，德罗贝特被误写为多布雷塔）。这有什么区别呢？

3. 编程。考虑通用搜索算法的Python代码 searchGeneric. py.

(i) 尝试运行各种搜索算法，以确保你理解它们的属性（如上），它们的实现和算法之间的关系。你可以编辑 searchTest.py 来调用罗马尼亚地图问题上的搜索方法。

(ii) 所提供的代码按照搜索问题（searchProblem. py）中罗马尼亚地图的相关定义中定义的顺序将继承者添加到边界中。通过试验继承者的不同顺序（如上面的字母顺序），确定算法对这种顺序的敏感度。

(iii) (更难）尽管面包优先搜索是A* 搜索的一个特例（解释一下！），在searchGeneric.py中写一个扩展Searcher的BreadthFirstSearcher类，直接实现面包优先搜索，即维护一个来自预先扩展的节点的状态集，并且只在一个扩展的节点的状态不在探索集或已经在前沿的节点中时将其加入前沿，并且在产生目标状态（未扩展）时终止。你将需要对构造函数和搜索方法进行编码。