pacman

姓名：覃果　学号： 2020012379　班级：软件02

2022 年 3 月 26 日

1　问题1

对 depthFirstSearch，breadthFirstSearch，uniformCostSearch以及aStarSearch加上启发函数四种算法在bigMaze，mediumMaze，smallMaze进行了测试，得到了如下的实验结果

(time,nodes expanded, cost)	DFS	BFS	UCS	aStar
smallMaze	(0, 93, 37)	(0, 94, 19)	(0, 92, 19)	(0, 53, 19)
mediumMaze	(0, 269, 246)	(0, 275, 68)	(0, 269, 68)	(0, 221, 68)
bigMaze	(0, 466, 210)	(0, 620, 210)	(0, 620, 210)	(0, 549, 210)

通过上表可以看出 UCS 相比于 BFS 扩展的节点更少，而 aStar 相比与 UCS 扩展的节点更少。BFS 是路径代价一样时 UCS 的特殊情形，而这在问题一中默认的路径代价都是1。那么为什么 UCS 比 BFS 扩展的节点更少呢？

这是由于在实现 BFS 的时候，我只维护了一个 explored 集合，而在实现 UCS 的时候维护了 explored 和 frontier 两个集合。此时，UCS 能保证在 frontier 中的点不再被 expend，而 BFS 在节点被 explored 之前可能被 expand 多次。

UCS 是 aStar 没用启发函数的特殊情形，是一种类似等高线的搜索。而 aStar 加入了我设计的启发函数，搜索的方向更加不平衡，能更快搜索到目标节点。

启发函数的定义如下：

$$yourHeuristic = manhattanDistance = |\Delta x + \Delta y|$$

admissible：pacman 只能走上下左右四个方向，从任意一个点到终点需要的距离必然大于两者间的 manhattanDistance

consistent：

$$n : (x_1, y_1), n' : (x_2, y_2), goal : (x_0, y_0)$$

$$\Rightarrow h(n) - h(n') = |x_1 - x_0| + |y_1 - x_0| - (|x_2 - x_0| + |y_2 - y_0|) \leq |x_1 - x_2| + |y_1 - y_2| \leq cost(n, n')$$

2 问题二

对于地图 mediumScaryMaze 设计了 MediumScarySearchAgent，采用了 uniformCostSearch，修改了代价函数为 1.1^x，使得 pacman 尽量往左走，避开怪物。可采用 python pacman.py -l mediumScaryMaze -p MediumScarySearchAgent 测试效果

对于地图foodSearchMaze 设计了 FoodSearchAgent，采用了 uniformCostSearch，修改了代价函数，将中间空白部分的代价提高，设置终点为右下角的最后一个豆子。使得 pacman 能够最短地吃到所有豆子，可采用 python pacman.py -l mediumScaryMaze -p FoodSearchAgent 测试效果

3 问题三

MinimaxAgent 和 AlphaBetaAgent 如果只采用朴素实现，很容易导致当周围没有豆子的时候， pacman 上下来回循环游走。于是修改了评估函数，将当前位置与所有食物的距离之和的倒数纳入评估。但是，当 pacman 两个方向有差不多的食物之时，仍然会陷入循环游走，只能等待 ghost 将 pacman 追逐出当前区域才有可能获得胜利。

为了解决这个问题，设计了三种方法分别应用于 MinimaxAgent, AlphaBetaAgent, 以及 ExploredAlphaBetaAgent。

3.1 随机游走法

当检测到 pacman 正在循环游走的时候，进入随机游走状态，从当前可以选择的方向中随机选择一个方向走。连续10次。如果下次仍然陷入循环，将连续随机游走的次数提升1.2倍。

当发现 ghost 在身边的时候，停下随机游走状态，换回 Minmax 搜索。

实验表示此方法能够提升胜率和得分，但是显得过于盲目，随机游走的时候有大量冗余操作。

3.2 探索集法

一个简单的事实是，pacman 去过的地方就以及不会有豆子了。所有走过的地方很难再提升我们的分数，对我们是没有意义的。于是维护了一个 explored 集合。每次搜索的时候遇到 explored 中的节点时，就不继续向下搜索了。

当发现 ghost 在身边，或者发现自己周围所有的点都被 explored 的时候，清空 explored 集合。

实验表示此方法能够大幅度提高分数，因为 pacman 对于没去过的地方会优先考虑，减少了不必要的移动。但是由于对于 explored 中的节点不向下搜索，只有当 ghost

靠近的时候清空 explored 才开始搜索，导致 pacman 容易被 ghost 包围，而发现被包围的时候可能已经没有可以逃脱的路线了。（高收入当然有高风险）

3.3 搜索法

我们不希望 pacman 死亡率太高，因为死亡的惩罚比较高。所以我们希望 pacman 能够有全局的视野，但是不能陷入循环，于是采用了如果检测到循环，就采用 BFS 搜索出走向距离 pacman 最近的那个豆子的路径。然后朝那个豆子走过去。

此时如果发现 ghost 在身边的时候，返回 AlphaBetaAgent 搜索。

实验表示此方法能够达到的分数上限比第二种方法略低，但是稳定性高，死亡率低。平均得分最优。

3.4 剪枝效果

采用 MinimaxAgent depth=3 时比较流畅，depth=4每一步就要将近1s多的时间。

采用 AlphaBetaAgent 和 ExploredAlphaBetaAgent depth=4 仍然非常流畅。