搜索实践实验报告

周雨豪 2018013399 软件92

问题一

tinyMaze

	耗时 (ms)	展开节点	路径代价
DFS	0.55	22	8
BFS	0.46	15	8
UCS	0.47	15	8
A*	0.42	14	8

mediumMaze

	耗时 (ms)	展开节点	路径代价
DFS	6.05	292	246
BFS	5.70	269	68
UCS	5.68	269	68
A*	4.70	221	68

bigMaze

	耗时 (ms)	展开节点	路径代价
DFS	12.90	722	210
BFS	12.41	620	210
UCS	12.24	620	210
A*	11.35	549	210

根据测试结果来看，DFS 耗时最长且展开节点最多，BFS 和 UCS 由于每一步 cost 相等所以在此等价，均优于 DFS，A* 耗时最少且在节点展开数上也更优，因为使用曼哈顿距离作启发函数，所以相比 BFS 会展开更少的无关节点。

三种算法中除了 DFS，其余都一定能够找到最优路径。例如在 mediumMaze 中，DFS 搜索结果的路径代价就远高于最优解。

所有算法耗时大致与展开节点数成线性关系，而在 bigMaze 的测试结果中 A* 展开节点数显著较少的情况下耗时仍与BFS几乎相同，其原因应该是计算启发函数导致的额外耗时。

A* 启发函数定义为 pacman 到 goal 的曼哈顿距离，是良定义的：由于吃豆人的移动模式是上下左右，而且考虑到避开墙壁，所以曼哈顿距离一定不大于实际移动代价，因此满足 admissibility；在状态 n 下，吃豆人做任何移动的代价是 1，因此证明 h(s) $\leq$ 1 + h(s') 即可，吃动人做出一个动作可能：(1) 不移动，满足 h(s) $\leq$ 1 + h(s') ，(2) 离食物更近，这时 h(s) = 1 + h(s')，(3) 离食物更远，也满足。因此满足 consistency。

问题二

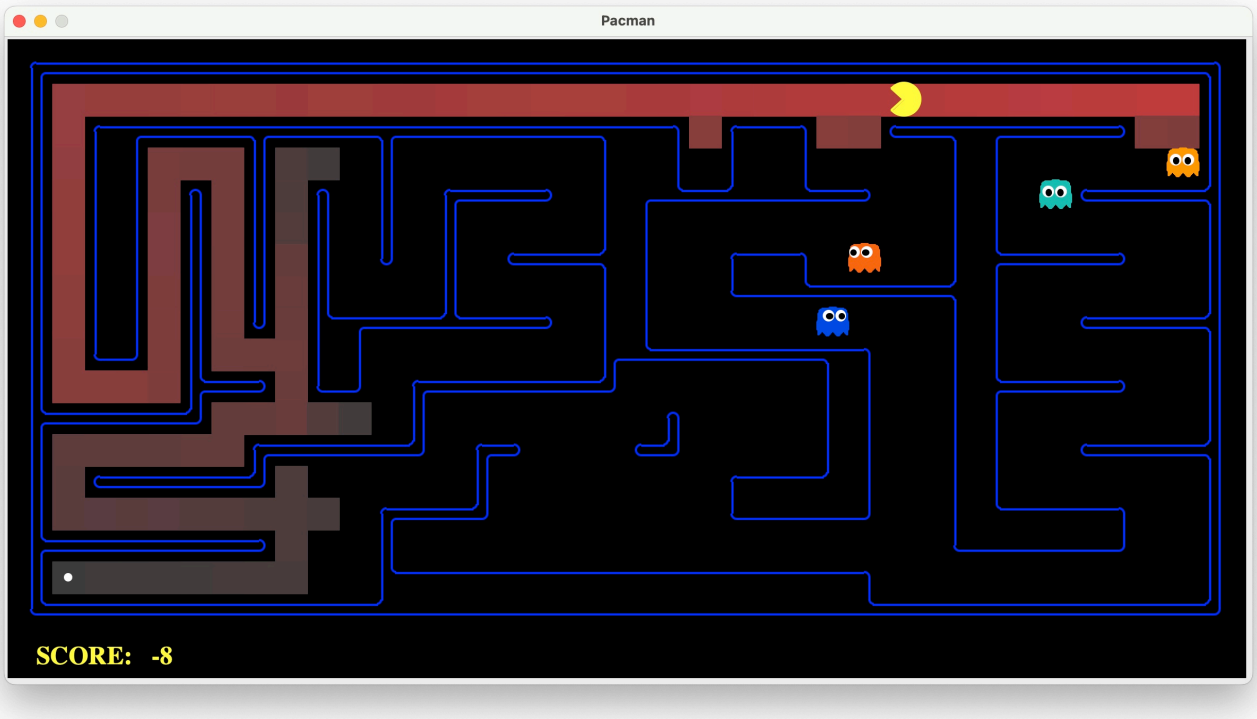
针对两个测试地图写了两个 agent，测试命令如下：

```
python pacman.py -l mediumScaryMaze -p mediumScaryAgent
python pacman.py -l foodSearchMaze -p foodSearchAgent
```

搜索算法选用 A*，由于题目规定只通过重写代价函数来改变 agent 行为，所以两个 agent 都是针对相应地图的设计编写，不具有通用性，但是可通过测试。

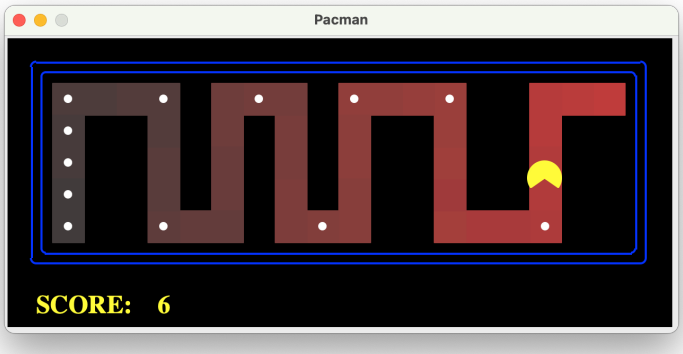
mediumScaryMaze

pacman 的任务是避开怪物吃到 (1, 1) 的食物，观察到怪物初始位置都在地图右上方，因此代价函数的设计思路就是让 pacman尽可能从左侧找路而不是走右边（否则很可能碰到怪物）。代价函数设计为 $f = 2^x + 2^{-y}$，即地图下方比上方代价高，右侧比左侧代价高，这样 pacman 会沿着左上方寻路而避开怪物。



foodSearchMaze

由于问题的 goal 设置为 (1, 1)，所以设计思路就是让 pacman 从右至左上下迂回地吃完所有食物。实现是将特定区域（图中迷宫里的黑色区域）的代价设得非常高，让 pacman 不会通过这些区域，类似于用极高的代价生成的墙壁，迫使 pacman以一个特定的迂回路线走到目标。



问题三

自己设计的迷宫见 myMaze.lay。为避免陷入原地循环加入了随机化。录屏的两个测试指令如下。

```
python pacman.py -p MinimaxAgent -l myMaze.lay -a depth=3
python pacman.py -p AlphaBetaAgent -l myMaze.lay -a depth=3
```

不过大部分情况下吃豆人不能吃到所有食物（大概是因为怪物部分动作的随机化）。