|  |
| --- |
| 哈尔滨工业大学(深圳) |
| **《人工智能》实验报告** |
|  |
| **实验一**  **搜索策略pacman**   |  |  | | --- | --- | | 学 院: | 计算机科学与技术 | | 班 级: | 1班 |   姓 名:   |  |  | | --- | --- | | 学 号: |  | | 学 期: | 2022春 | | 实验地点: |  | | 实验教师: |  |   目录  [1 实验内容 2](#_Toc97199669)  [2 实验记录 2](#_Toc97199670)  [2.1. 实验环境 2](#_Toc97199674)  [2.2. 问题分析、算法设计和实现 2](#_Toc97199675)  [2.2.1. 问题1：深度优先算法 2](#_Toc97199676)  [2.2.2. 问题2：宽度优先算法 3](#_Toc97199677)  [2.2.3. 问题3：代价一致算法 3](#_Toc97199678)  [2.2.4. 问题4：A\*算法 4](#_Toc97199679)  [2.2.5. 问题5：找到所有的角落 4](#_Toc97199680)  [2.2.6. 问题6：角落问题（启发式） 5](#_Toc97199681)  [2.2.7. 问题7：吃掉所有的豆子 5](#_Toc97199682)  [2.2.8. 问题8：次最优搜索 6](#_Toc97199683)  [2.3. 实验结果 7](#_Toc97199684)  [3 总结 7](#_Toc97199685)  [3.1. 问题及解决方法 7](#_Toc97199688)  [3.2. 实验的启发、总结及建议 7](#_Toc97199689)  [4 参考文献 7](#_Toc97199690)  *实验报告内容包含但不限于以下内容，如有补充请用红色\*标注。*  *注意标注每部分的作者，作者可标注在小标题后面。* |

# 实验内容

*概述本次实验的内容，解决了哪些问题，用到了哪些算法。*

# 实验记录



## 实验环境

*阐述实验用到的环境，包括操作系统、开发软件及版本号。*

## 问题分析、算法设计和实现

*对本实验的8个问题，逐个阐述对问题的理解分析，给出解决方案，描述所使用的数据结构和算法，描述实验结果。*

### 问题1：深度优先算法——190110121-杨航

1. 问题描述和分析

Pacman在起点出发，利用深度优先算法找到通向目标点的一条路径。

1. 解决方案和算法描述

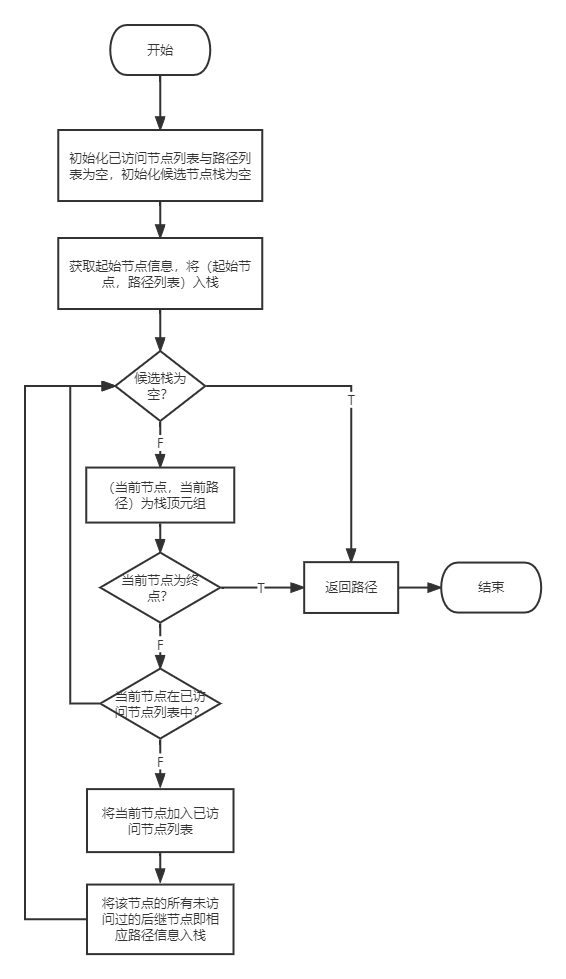
*给出解决方案，描述所使用的数据结构和算法，算法描述要求用流程图+文字说明。*

数据结构：栈

算法：深度优先搜索算法

算法描述：从起点处出发，尽可能深地搜索当前点的继承节点，直到到达目标结点。

当前结点不是目标结点且为曾访问过时，首先将当前结点标记为已访问，再将当前所到点加入到候选栈当中，每次从栈顶的状态结点开始扩展；当当前结点不是目标结点且无法继续扩展，则从该结点的父节点重新寻找路径，直到找到一条可通道路。



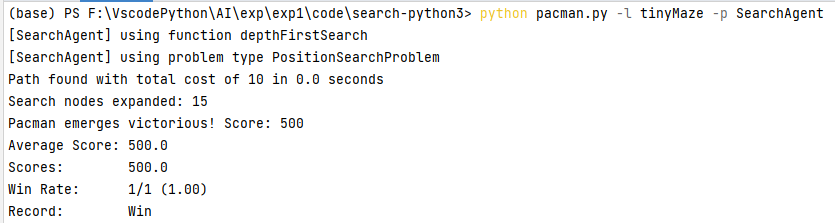
Q1-流程图

1. 实验结果

*列出你使用的测试命令，给出系统中间输出、最终输出等信息，要求有屏幕截图及说明。*

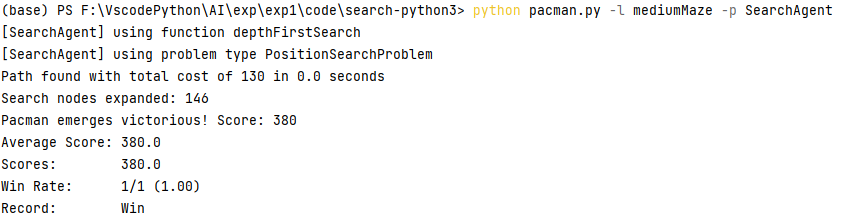
* 测试命令1：

python pacman.py -l tinyMaze -p SearchAgent



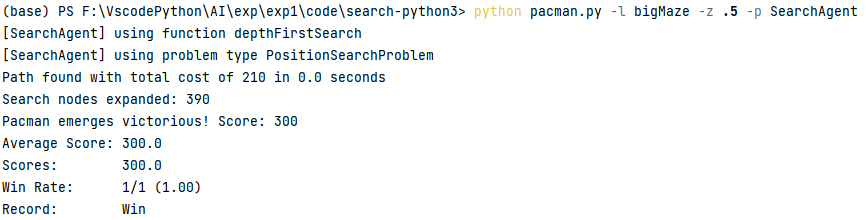
可知结点使用深度优先搜索方法，搜索代理为寻找某个特定点时的方法。该次搜索共扩展了15个点，路径长为10。

* 测试命令2：python pacman.py -l mediumMaze -p SearchAgent



可知结点使用深度优先搜索方法，搜索代理为寻找某个特定点时的方法。该次搜索共扩展了146个点，路径长为130。

* 测试命令3：python pacman.py -l bigMaze -z .5 -p SearchAgent



可知结点使用深度优先搜索方法，搜索代理为寻找某个特定点时的方法。该次搜索共扩展了390个点，路径长为210。

### 问题2：宽度优先算法 by 谢俊安

1. 问题描述和分析

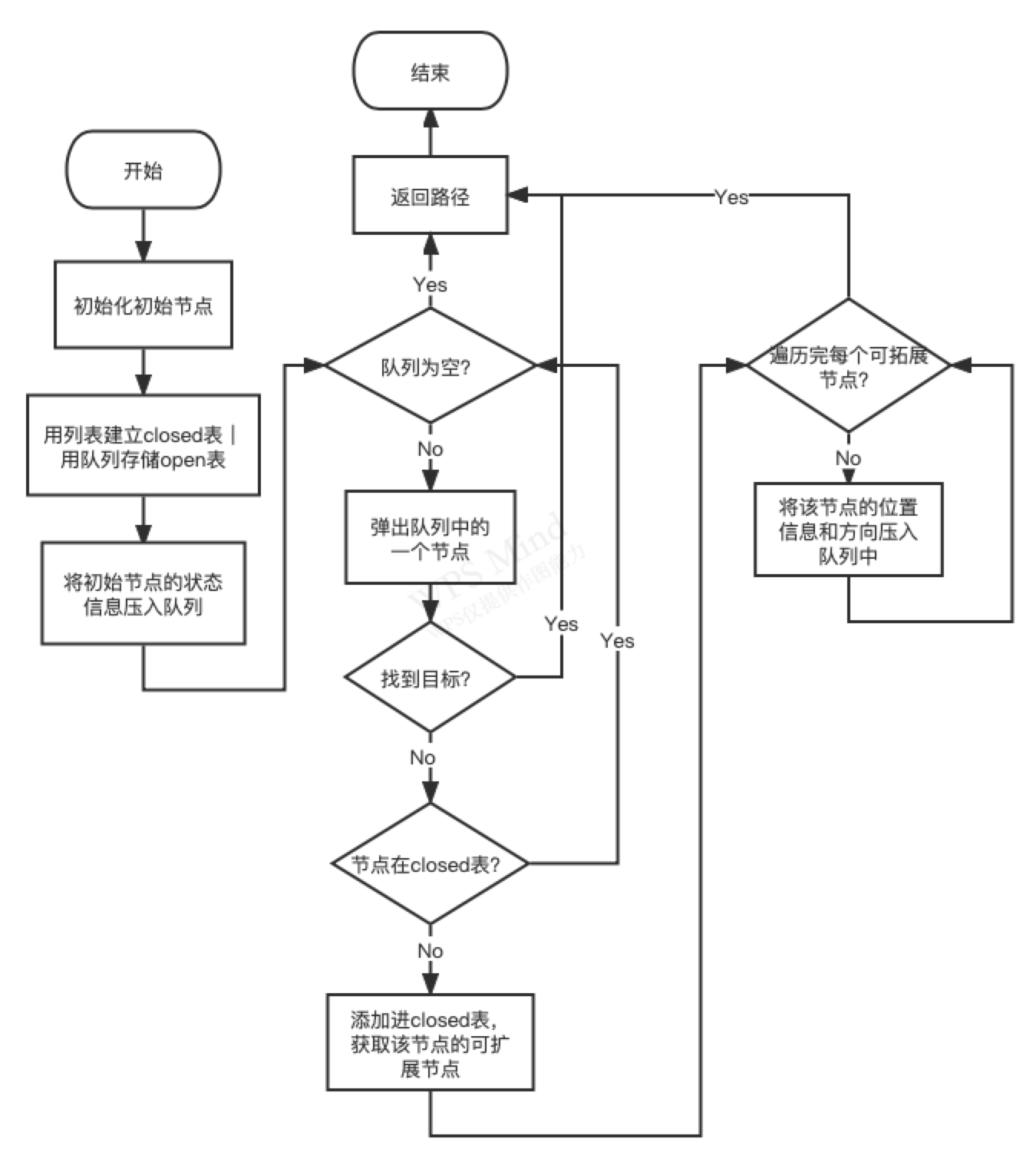
Pacman在起点应用BFS算法探索找到一个特定位置的豆子，并返回路径。

1. 解决方案和算法描述

数据结构：队列

算法描述：

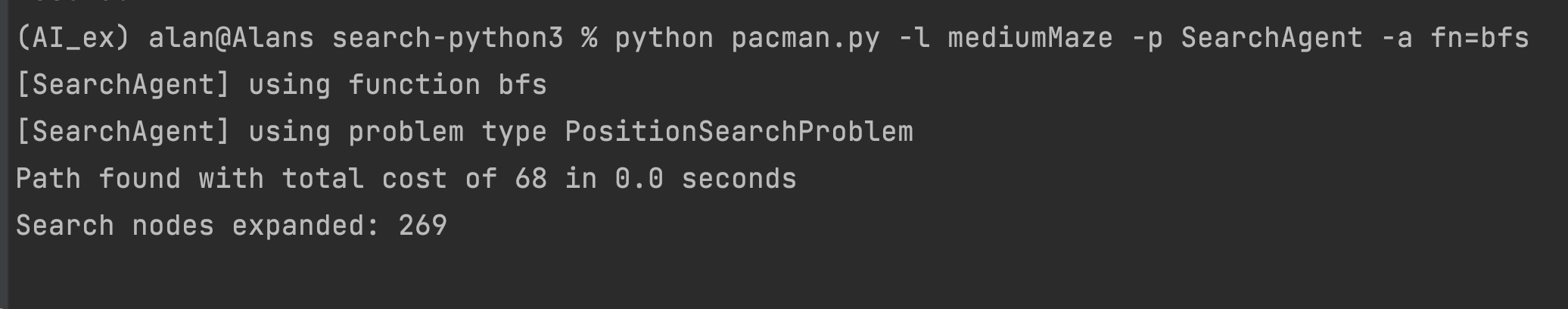
利用队列“先进先出”的机制，实现BFS的算法细节。将队列中的节点一个个弹出，并找到他们的孩子节点。如果找到目标节点，则成功，结束；否则执行该操作直到队列中无其他元素（即遍历完毕）若仍找不到节点则失败。



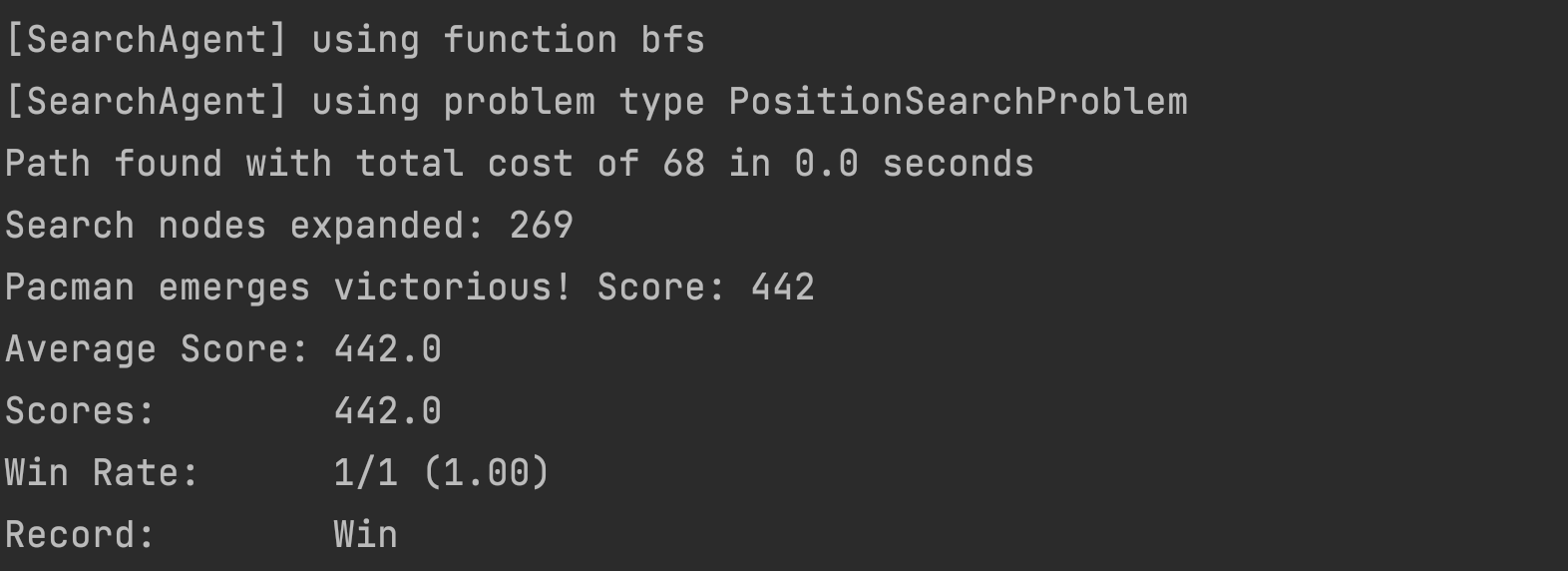
1. 实验结果

*列出你使用的测试命令，给出系统中间输出、最终输出等信息，要求有屏幕截图及说明。*

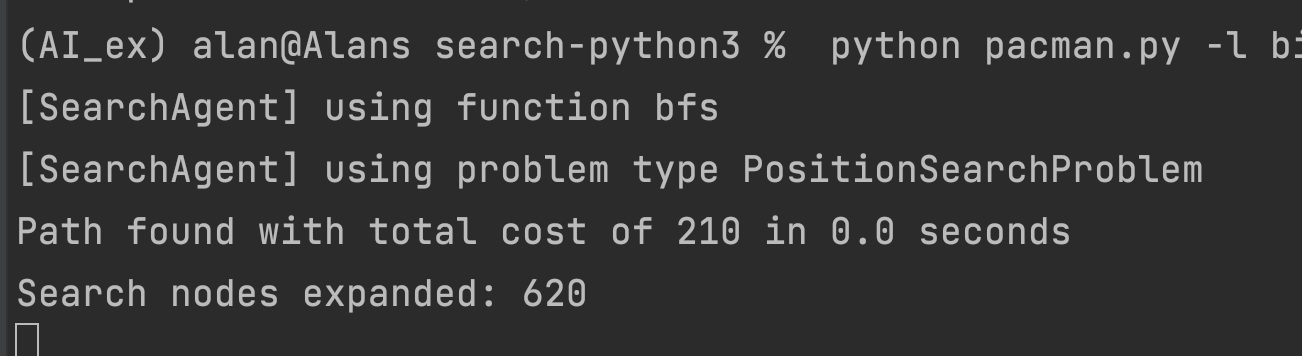
1. 命令：python pacman.py -l mediumMaze -p SearchAgent -a fn=bfs
   1. 中间输出：pacman通过BFS算法探索了269个节点，花了68的代价，执行最后返回的路径。



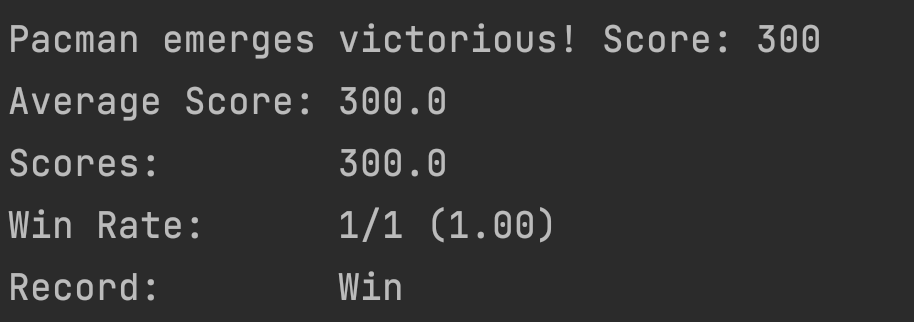
* 1. 最终输出：成功找到最终节点。



1. python pacman.py -l bigMaze -p SearchAgent -a fn=bfs -z .5
   1. 中间输出：pacman通过BFS算法探索了620个节点，花了210的代价，执行最后返回的路径。



* 1. 最终输出：成功找到最终节点。



### 问题3：代价一致算法

1. 问题描述和分析
2. 解决方案和算法描述

*给出解决方案，描述所使用的数据结构和算法，算法描述要求用流程图+文字说明。*

1. 实验结果

*列出你使用的测试命令，给出系统中间输出、最终输出等信息，要求有屏幕截图及说明。*

### 问题4：A\*算法

1. 问题描述和分析
2. 解决方案和算法描述

*给出解决方案，描述所使用的数据结构和算法，算法描述要求用流程图+文字说明。*

1. 实验结果

*列出你使用的测试命令，给出系统中间输出、最终输出等信息，要求有屏幕截图及说明。*

### 问题5：找到所有的角落 by 谢俊安

1. 问题描述和分析

基于 BFS搜索算法找到迷宫角落里的四个豆子，并返回一条访问四个豆子的路径。

1. 解决方案和算法描述

*给出解决方案，描述所使用的数据结构和算法，算法描述要求用流程图+文字说明。*

解决方案：更新几个函数，利用BFS算法即可找到，具体如下：

函数：

1. 初始化：

将地图的边界值保留在类中，即right & top。

1. def getStartState(self):

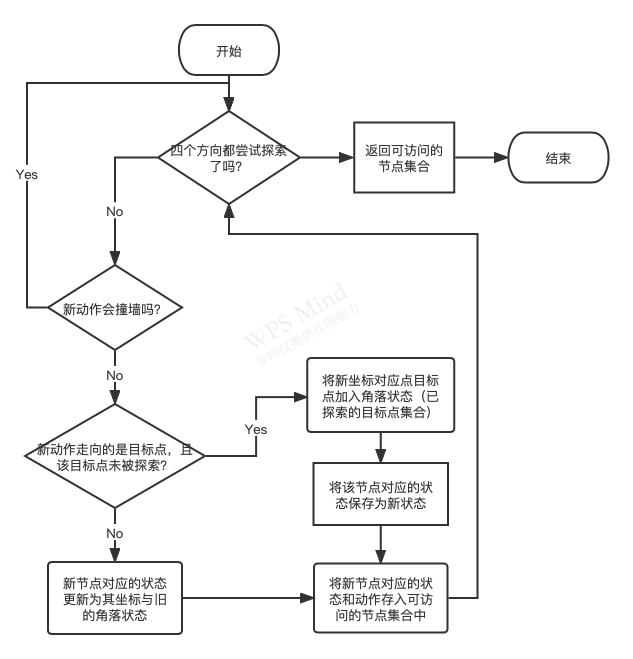
获取初始节点状态的函数需要改变，新的状态包含pacman的位置和四个角落的状态（有没有访问过）

1. def isGoalState(self, state):

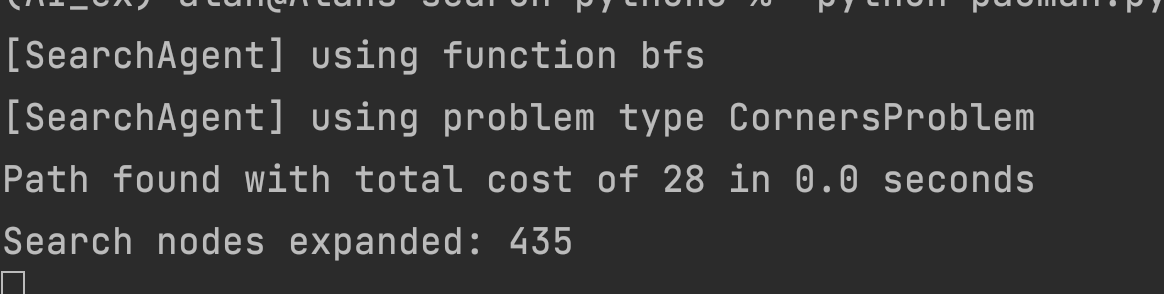
目标达成不再是抵达某一个节点，更新为四个角落都成功访问。

1. def getSuccessors(self, state):

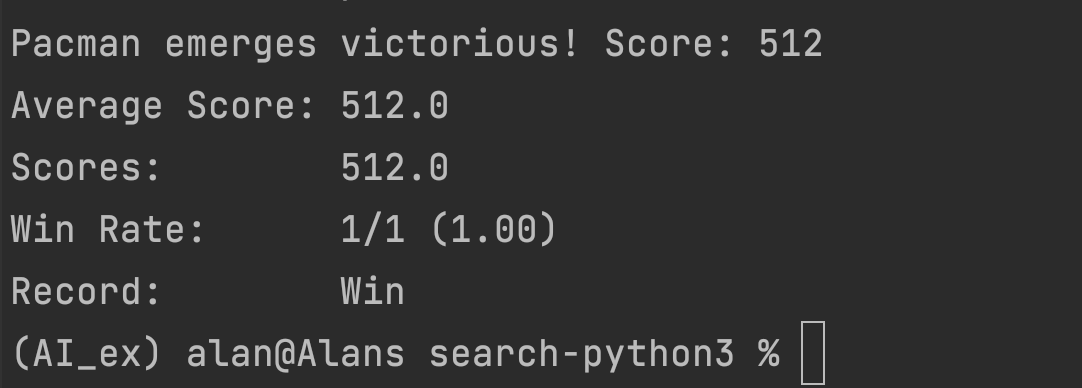
找到当前节点的可访问子节点的函数需要更新，具体体现为该节点与其子节点的状态更新除了包括自身的坐标以外，还要包含角落的状态。具体流程图如下：



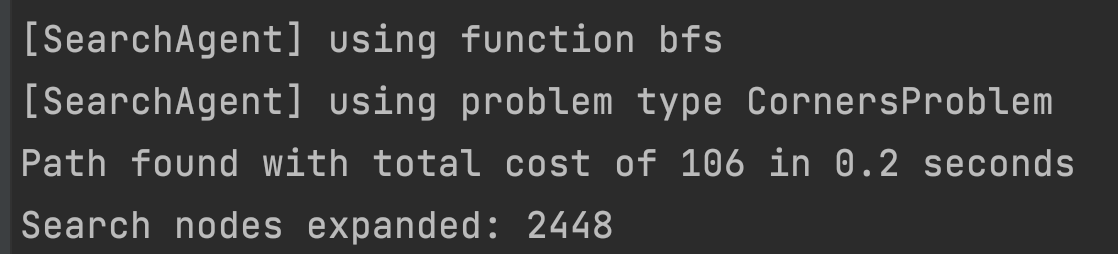
1. 实验结果
2. python pacman.py -l tinyCorners -p SearchAgent -a fn=bfs,prob=CornersProblem
   1. 中间输出：pacman通过BFS算法探索了435个节点，花了28的代价，执行最后返回的路径。



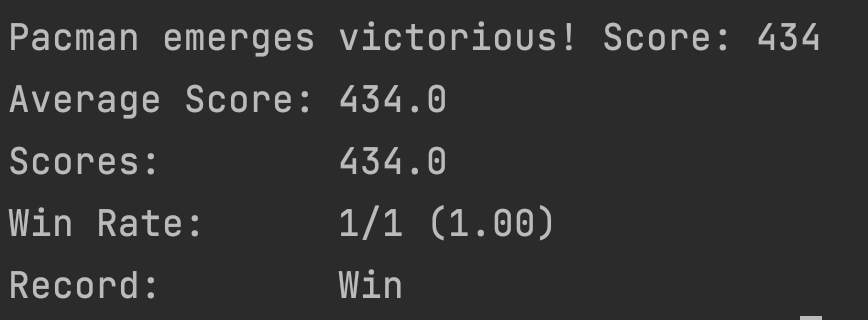
* 1. 最终输出：成功找到最终节点。



1. python pacman.py -l mediumCorners -p SearchAgent -a fn=bfs,prob=CornersProblem
   1. 中间输出：pacman通过BFS算法探索了2448个节点，花了106的代价，执行最后返回的路径。



* 1. 最终输出：成功找到最终节点。



### 问题6：角落问题（启发式）

1. 问题描述和分析
2. 解决方案和算法描述

*给出解决方案，描述所使用的数据结构和算法，算法描述要求用流程图+文字说明。*

1. 实验结果

*列出你使用的测试命令，给出系统中间输出、最终输出等信息，要求有屏幕截图及说明。*

### 问题7：吃掉所有的豆子

1. 问题描述和分析
2. 解决方案和算法描述

*给出解决方案，描述所使用的数据结构和算法，算法描述要求用流程图+文字说明。*

1. 实验结果

*列出你使用的测试命令，给出系统中间输出、最终输出等信息，要求有屏幕截图及说明。*

### 问题8：次最优搜索——190110121-杨航

1. 问题描述和分析

定义一个距离函数，在所有的食物结点里找到离当前点最近的一个，依次访问这些结点，直到所有食物结点全部访问。

分析：即对于每个当前状态，找到所有食物中最近的一个作为当前的目标点，调用搜索算法（此处使用过DFS、BFS、A\*(启发函数为曼哈顿) ，效果最好的为A\*搜索算法，因此使用A\*）。循环上述操作，直到所有的食物被吃光。

1. 解决方案和算法描述

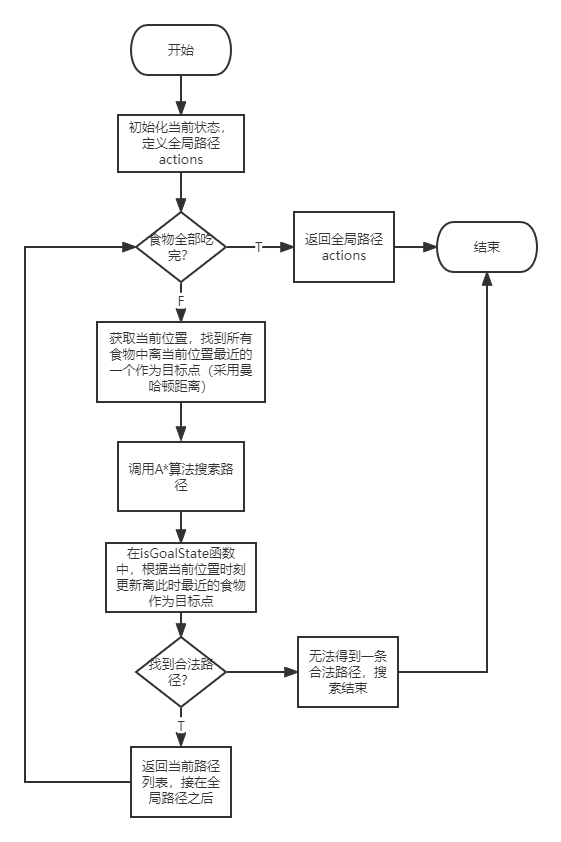
*给出解决方案，描述所使用的数据结构和算法，算法描述要求用流程图+文字说明。*

解决方案：在需要吃掉所有的食物的时候，首先根据曼哈顿距离确定离当前位置最近的一个食物，作为下一目标点。调用已经完成的算法获得去到当前目标点的一个路径。

需要注意的是，在移动的过程中需要时刻更新目标点，因为也许在移动过程中会离最初的目标点越来越远，因此除了在初始状态的时候需要确定当前的目标点，在isGoalState函数中也同样添加了更新目标点的操作，保证时刻的目标点都是离当前位置最近的一个。

数据结构：优先队列

算法：A\*算法

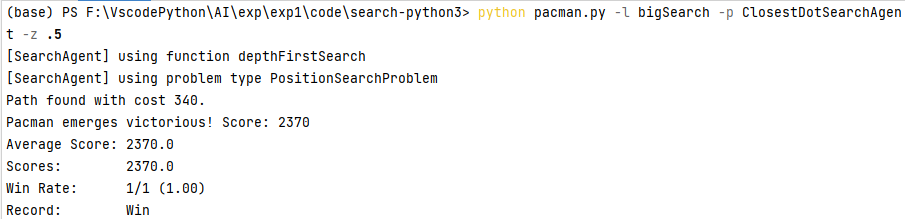


Q8-流程图

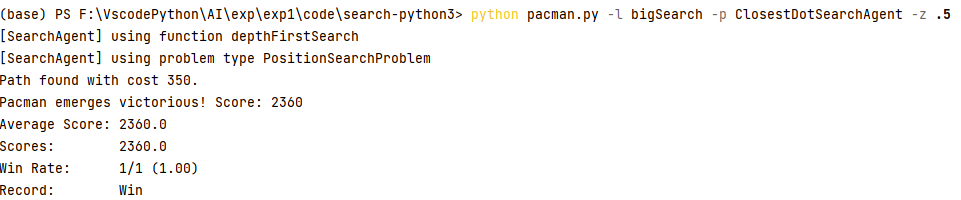
1. 实验结果

*列出你使用的测试命令，给出系统中间输出、最终输出等信息，要求有屏幕截图及说明。*

测试命令：python pacman.py -l bigSearch -p ClosestDotSearchAgent -z .5



上述截图为采取A\*搜索算法（由另一同学完成）进行全局搜索时的成绩，路径长为340，得分2370。在选定A\*前，还分别尝试了dfs、bfs的搜索算法，得到的结论为A\*算法（启发函数为曼哈顿距离）为三者之间最优，dfs表现最差，且是远差于bfs与A\*。



上图为bfs搜索算法的得分，与A\*相差不大。而dfs在多目标搜索的时候甚至达到了负分的情况（探索BigSearch的时间太长，但得分很低因此未等待最后结果），直觉认为dfs栈的特性不适合与此问中时刻根据位置更新目标点的方法结合进行多目标搜索。

## 实验结果

*使用autograder.py对代码进行测试，截图并分析结果。*

# 总结



## 问题及解决方法

**190110121-杨航：**

问题：对这种代理的方法使用不太熟悉，加上一开始因为文件的规模太大迟迟不敢开始，无法静下心来认真地阅读各个方法、类的调用，所以对整个游戏的总体有点迷糊。看了很久的代码感觉自己在几个函数之间来回跳动，加上由于模式多样，使用不同的代理有不同的解决函数，导致看着看着就不知道自己到底在看什么了。

解决方法：光是看代码还是没有用的，在前期无法对所有的细节都有透彻的理解，还是需要先动手写代码。在开始完成DFS的时候才发现之前考虑的“需要怎么获取起始位置、怎么判断哪边是墙”这类我觉得很繁琐的问题都已经被封装完成，起始需要做的只是很小的一部分（并且是以前常写的算法）。

**190110115-谢俊安：**

问题：

* 1. 接触一个新的项目不知道从何入手，完全懵逼的状态。
  2. 对于python语法及其库的不熟悉，比如列表、元组嵌套的问题，比如.append 这类功能函数的调用，不了解根本不知道有这个渠道。

解决办法:

1. 根据已有的实现的函数和功能出发，按照代码的逻辑顺着理解一遍，基本都能有清楚的认知，也就可以下手了。
2. Python 的问题只能死磕，不会就查，实际地写下代码之后就能够理解了。

## 实验的启发、总结及建议

**190110121-杨航：**

启发：

在明白大体的框架后，不应该紧揪着一些不能理解的小细节不放，应该先大概构思解决问题的方法，才能直到自己需要哪些信息。放在本实验中便是：起始位置（可根据代理中的getStartState()获得）、是否到目标位置（可根据代理中的isGoalState()判断）、当前可走路径（可根据代理中的getSuccessors()获得）……由于在前4个小问中已有上述已经完成好提供API的函数，因此在做后4问的时候便对代理与问题的关系更加清晰。在完成代码的过程中，也会对刚开始的时候所遇到的不理解的细节有更好的认识。

建议：

参考文献中的第二个网站很值得在开始前阅读，里面给出的提示和注意事项能让在开始前有一些启发，比指导书更为详细一些。

**190110115-谢俊安：**

1. 对于BFS及利用其寻找多点最短路径问题有了更加深刻的理解；
2. 不理解代码的时候溯源去找到源代码、看已有框架是如何调用的；
3. 刚入手的时候可以直接进入调试状态去查看各个类、功能函数里具体有哪些变量，方便理解。
4. 对于从问题2到问题5的跨变，其实只是节点的状态重新定义了，但是对于整体代码的撰写其实都产生了很大的影响，以至于做完问题5再回头看问题2会一阵错愕以为自己写错了。

# 参考文献

190110121-杨航：

<http://ai.berkeley.edu/contest.html> 为伯克利官方文档的一部分，与指导书相似，但比指导书更为详细一些。

<https://inst.eecs.berkeley.edu/~cs188/fa18/project1.html#Q5> 也与指导书类似，但该文档中有更多的提示与注意事项，并且给出了一些关于搜素结点数目和路径长度的参考，以便与自己的方案进行对比。

190110115-谢俊安：

感觉指导书就足够了，当然还有万能CSDN对于我理解python的帮助..