# Artificial-Intelligence Project 1

Weipo, Lai 19340980040

Search Algorithms

1. Deep First Search (DFS)

深度优先搜索（DFS）是一种用于遍历或搜索树或图的算法。这种算法尽可能地深入到树的分支中进行搜索。当节点v的所有边缘都被探索过后，搜索将回到发现节点v的边缘的起始节点。这个过程一直持续到找到源节点之前的所有节点为止。如果仍有未被发现的节点，则选择其中一个节点作为源节点，重复这一过程，直到所有节点都被访问。

1. Breadth First Search (BFS)

它是一种图搜索算法。简单地说，BFS从根节点开始，沿着树的宽度遍历树的各个节点。如果所有的节点都被访问了，算法就会被中止。

1. Uniform-cost Search (UCS)

在计算机科学中，统一成本搜索是一种树状搜索算法，用于遍历或搜索加权树、树状结构或图。搜索从根节点开始。搜索继续进行，访问从根节点开始总成本最低的下一个节点。节点以这种方式被访问，直到达到一个目标状态。通常情况下，搜索算法涉及扩大节点，将所有由有向路径连接的未扩大的相邻节点添加到一个优先级队列。在队列中，每个节点都与它从根部出发的总路径成本有关，其中成本最低的路径被赋予最高的优先权。处于队列头部的节点随后被扩展，加入下一组连接的节点，其总路径成本从根部到各个节点。

Dijkstra’s algorithm: 最初的版本只适用于寻找两个顶点之间的最短路径 后来，後來更常见的变体将一个顶点固定为源节点，然后寻找从该顶点到图中所有其他节点的最短路径，产生一个最短路径树

1. A\* Search

A\* search algorithm是一种为图平面上有多个节点的路径寻找最低通过成本的算法。它经常被用于计算游戏中NPC的移动，或用于计算网络游戏中BOT的移动。

该算法结合了最佳优先搜索和Dijkstra算法的优点：它可以保证找到一条最优路径（基于评价函数），同时通过进行启发式搜索提高算法的效率。

Q1: Finding a Fixed Food Dot using Depth First Search

We need to build a DFS to find a fixed food dot. Using the stack data structure, we can get a deep first search algorithm.

Q2: Breadth First Search

Just need to change the stack in Q1 to queue, we can get the answer.

Q3: Varying the Cost Function

Same as Q2. Just need to change the stack in Q1 to PriorityQueue, we can get the answer.

Q4: A\* Search

The difference between Q3’s PriorityQueue is the cost. f(x) = g(x) – h(x)

g(x) represents the cost from the start to current state, h(x) represents the estimate distance between the current state and the goal state. Path found with total cost of 54 in 0.1 sec and search nodes expanded is 535

Q5: Finding All the Corners

we should eat all dots in 4 corners. We change the visited\_posi (visited positions) in about BFS from dict to list.

Q6: Corners Problem: Heuristic

Define our heuristic function to solve the problem. Attention to the h(x) part of the A\*

1. we add all the manhattan distance between the pacman and all the unvisited corners together.

2. the second one is choose the minimal manhanttan between the pacman and all the unvisited corners

If we choose the first one, expanded node number is just 502. and it's 1475 for the second one. so that the first is better.

Q7: Eating All the Dots

If there are different amount of dots unvisited, we might use different distance function. If there are many unvisited nodes, we don't want to visit too many nodes, might choose less expanded nodes function that is path distance

The unvisited node is few, so we just choose the manhatten’s distance as the heuristic, and we can get the result where it costs just 12.2 seconds and 9551 expanded nodes

Q8: Suboptimal Search

Find the closest dot, we can get the answer. We can get the path length by using path distance function in Q7.