CSC3206: Artificial Intelligence

A1 Assignment 1

Group Name: bt smeli

Group Member:

No	Name	Student ID
1.	Bryan Tang Kui Yan	18041772
2.	Lai Chun Hou	18043547
3.	Lai Weng Xi	18043521
4.	Lim Wei Jun	18034223

Table of Contents

Introduction

Uninformed Search Algorithm

Uninformed search algorithm also known as blind search since the node is generated without prior additional information for the searching process to the solution or any domain knowledge regarding the tree. Hence, all movements are followed blindly, in which all nodes are generated without information, because there is no function to check the cost or distance between nodes to perform path search efficiently. Therefore, uninformed search algorithm operates in the brute force manner.

In addition, the provided information for the problem are the initial state and goal state. Thus, the algorithm searches for all possible options until the goal state is reached.

Types of uninformed search algorithms are:

- Breadth First Search
- Depth First Search
- Uniform Cost Search
- Iterative Deepening Depth First Search
- Bidirectional Search

Informed Search Algorithm

Informed search algorithm also known as heuristic search. In contrast to uninformed search algorithm, information/ domain knowledge is provided, in which there are prior information for the searching process also known as the heuristic function. A heuristic function estimates the minimum cost required to get to the goal state from the current state. The information is derived from the heuristic function and the existence of these information help in improving the search's performance. Moreover, the goal test of this algorithm is always done during the expansion of nodes.

In addition, the provided information for the problem includes the initial state, goal state and the heuristic function. Therefore, the algorithm searches for the most optimal solution based on the function.

Types of informed search algorithm are:

- Greedy Best First Search
- A* Search

Differences between uninformed and informed search algorithm.

	Uninformed Search	Informed Search
Searching method	No prior additional knowledge for searching.	Uses knowledge for searching.
Cost	High cost. (due to time and space complexity)	Low cost.
Searching time for solution	Slower.	Faster. (heuristic function)
Completion	Tree is always complete.	Tree might be or might not complete.
Efficiency	Lower.	Higher.
Goal test	During generation of nodes.	During expansion of nodes.

The performance of the search algorithm is evaluated based on:

Completeness

This measure assesses if the search algorithm guaranteed to find solution to the problem. (if it exists)

Optimality

This measure assesses if the search algorithm finds the optimal solution. (lowest cost solution)

Time complexity

This measure assesses the time required for the search algorithm to find the solution to the problem.

Space complexity

This measure assesses the memory space needed for the searching. (store the generated nodes)

Problem Formulation

Problem description

- One food at any time; Non-increasing snake length; Snake length of one; reaching at least 15 points.

Both the uninformed and informed search algorithms need to be implemented in such a way that the snake will eat the food in the maze. Furthermore, each food eaten by the snake will increase the point by 1, and the algorithm must score at least 15 accumulated point before the snake game ends. Lastly, the length of the snake will be static (remain as 1), and the search tree of each food eaten by the snake will be shown. (each steps taken towards the goal state)

Problem formulation

State space includes state, actions and transition model.

State – snake, whether the food is eaten.

Actions – Direction of the snake moves towards the food. (n, s, e, w)

Transition model – Based on the 4 actions above, the snake can move in 4 direction:

n – move the snake UP

s – move the snake DOWN

e – move the snake to the RIGHT

w – move the snake to the LEFT

Initial state – The starting location of the snake in the maze, which is at the coordinates In(9,3)

Goal state – The food.

Goal test – When the snake eats the food. {In(Food)}

Path cost – Each action increases the path cost by 1.

Uninformed Search Algorithm

Breadth First Search (BFS)

Breadth first search algorithm expand on the shallowest unexpanded node, it focuses on the steps. The order of nodes' generation and expansion followed the First In First Out (FIFO) queue method in the frontier. The searching process of this algorithm begins from the root node to the successor node of the root node. All nodes of the same level will be expanded before progressing to the node in the next level. Therefore, this search algorithm searches in the form of breadthwise. Furthermore, solutions are always found in the BFS, if there are more than 1 solution to the problem, then the lowest path cost solution will be chosen first. In the BFS, the redundant path removes the latter redundant node, and the goal test of this algorithm is by generation, thus, when the goal node is generated, the search will be terminated.

However, the efficiency in term of time and space is low if the goal state is positioned at the end branches of the tree. Due to FIFO method, it will consume high amount of time for the searching process and space to save the generated nodes, which leads to the increase in time complexity and space complexity.

Implementation

The snake traverses to the neighbor coordinates of the initial snake coordinates, the distance cost of each neighbor coordinates to the goal state is compared each explored coordinate will be stored in the list and it will not be visited again until it reaches the goal state.

The search function will receive 3 parameters namely the initial location, snake location and the maze size. These 3 parameters will be first defined in the player class through the setup and problem dictionary.

The search algorithm will identify the path to the food location by expanding all possible direction which is north, south, west, and east until the snake location reach the food location. Which is known as the children node and constantly check if the children node location has reached the food location. As we are searching with breadth-first search, the frontier will be arranged in searching horizontally first then vertically. The explanation will be explained with this given scenario where snake location is located at [0,0] and food location is located at [3,3].

Initial snake location is located at [0,0], hence, the returned children node will be [0,1] and [1,0]. Current frontier will be arranged with the children node [0,1] and [1,0]

Then the next node to be search is [0,1] and it will return children node of [0,0],[0,2],[1,1] Current frontier will be arranged as [1,0],[0,2],[1,1]

Hence, the third expansion will be searching through node [1,0] instead of [0,2]. Ensuring the search is horizontally completed first only move to vertically.

To fulfil the concept, we created a frontier variable with type of list to store the children node and to determine which node to go through next. Frontier = []

The children will be identified with the function: expandAndReturnChildren.

Every node will return 4 children node of moving up, down, left, and right. The children node will be inserted into the empty children list and return to the algorithm.

Node movement explanation:

0,0	1,0	2,0
0,1	1,1	2,1
0,2	1,2	2,2

Moving up will be subtracting the y-value by one,

Moving down will be adding y-value by one,

Moving left will be subtracting x-value by one,

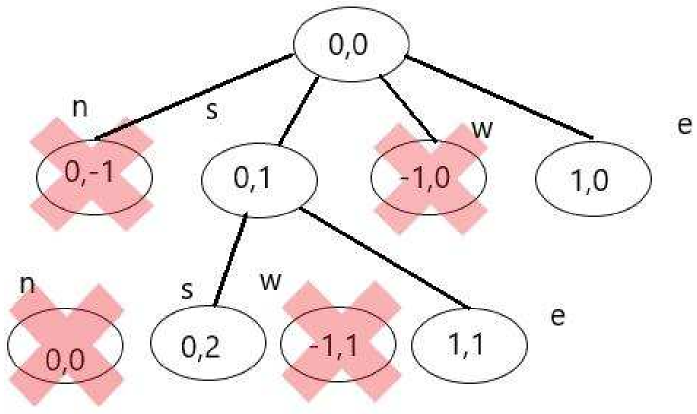
Moving right will be adding x-value by one.

This can be explained with the table shown above displaying the coordinate of each node.

For instance, moving node [0,0] down will be [0,1]. Hence, we will need to add the y-value by one.

Now, to explain why [0,-1], [-1,0] is not presented in the frontier is because our return children function will go through a code to ensure the children node will not exceed the maze. In second expansion, [0,0] is not presented in the frontier is because we will be filtering out loopy path to avoid redundancy with our explored and frontier list.

Next, the returned children acquired in the expandAndReturnChildren function will be checked with the food location to see if they match. Before checking, the children will go through the explored and frontier list to ensure it is not a loopy path. The code will continue expanding until the node have reach the food location.



<u>First expansion</u>
Current frontier:
[0,1] and [1,0]

<u>Second expansion</u>
Current frontier:
[1,0] [0,2],[1,1]

Informed Search Algorithm

Greedy Best First Search

Greedy Best First Search uses the heuristic function, h(n) in the expansion of nodes. The expansion of node is based on the estimated closest node to the goal state. This algorithm is only workable on graph search. Hence, if it is used for tree search, the loopy path will be remained, and the search will not reach the goal state as it will keep looping back and forth between the loopy nodes, which leads to infinite loops. Moreover, backtracking is possible in this algorithm, as it keeps the parent node in the frontier. With the help of the heuristic function, the searching will try to find the optimal solution of the problem, however, it is not guaranteed.

Implementation

Heuristic function h(n) – it will search through axis with the lower cost first; in this case, lower distance. Ensure the snake location reaches the same row or column as the food location depends on which axis they search first.

In this algorithm, we will search through either x or y axis first. In our code, it will first receive two parameters, which is the snake location and food location. Then, it will determine if it will be searching through x or y axis. If it will search through y-axis first, it will ensure the snake location will move until it reaches the same row as the food location. Then it will only start search through x-axis to ensure it reaches the same column as food location. When the snake location is in same column and row of food location means the snake have eaten the food.

This will be determined with the findBestFirstPath function. Where we will find the distance between y-axis of snake location and food location, and x-axis of snake location and food location. We will then compare which path have a shorter distance.

For instance, snake location = [0,0] and food location = [3,5]

The distance from snake to food through y-axis is 5 (5-0=5). Which mean it will need 5 move to reach the same row as food location.

Distance from snake to food through x-axis is 3 (3-0=3). Which mean it will need 3 move to reach the same column as food location.

In this case, searching through x-axis will have a lower cost. Hence, the algorithm will go through x-axis first then y-axis (horizontal then vertical).

If we do not compare the distance of x and y axis. By default, the snake will move vertically (y-axis) then horizontally (x-axis) which in the UI may appear to be dull. By comparing x and y-axis prior to the snake moving, the direction the snakes move would appear to be slightly in a zigzag motion (as how snake would crawl).

Discussion & Analysis

Uninformed Search Algorithm – BFS

Result & Performance

The solution obtained from this search algorithm able to find the shortest path cost from the initial node (snake location) to the goal node(food). The time taken for the snake to find its food are based on the size of the maze, smaller maze size higher performance and vice versa. The result and performance of using this search algorithm are consistent. In terms of consistency, the algorithm used by the snake from initial state to goal state for every iteration is the same.

In terms of the performance, the snake must be able to eat the food from the initial state. The algorithm implemented is workable and uses the shortest path solution. We make sure that our program codes are well encapsulated and abstract to reduce the time taken to complete the algorithm.

Informed Search Algorithm – Greedy best first search

Result & Performance

The result of the snake achieving its food is identical to the uninformed search algorithm. However, the algorithm behind it as well as the motion/direction of the snake's movement is slightly different when compared to the Breadth-First Search algorithm.

The performance of this algorithm is better than uniformed search in terms of the time required for the searching and the space needed to store the generated node. This is because by using uninformed search the goal is unknown, hence, expansion of every step is needed. Whereas in informed search, goal is given and by comparing the number of steps of x and y-axis, we can determine which direction the snake should move. In fact, the performance of this informed search algorithm is better as compared to the uninformed search algorithm. This is because in this algorithm, expansion of every step is not required. In this search algorithm, comparison of x and y-axis determines the direction of the snake moves hence, it terms of performance this algorithm is more efficient.

Conclusion

Uniformed search algorithm requires more time and space in the searching operation due to the lack of knowledge on the problem, whereas the informed search contain the heuristic function that provides information on the goal state, which helps to reduce the time and space complexity. Furthermore, uninformed search will surely find the optimal solution of the problem, whereas informed search algorithm does not guarantee optimal solution.

Moreover, breadth first search and greedy best first search is similar in a way with different searching technique, where both search algorithm look for the solution based on the lowest cost path from the initial node to the goal node. However, the implementation of the algorithms is slightly different. All in all, both of the algorithm can be used to find the least-cost path to reach the goal node but greedy best first search is way more efficient as compared to breadth first search as greedy best first search is an integration of breadth first search.