**AI-Controlled Snake Game Assignment**

Group member includes:   
1. Shreya Dey- 2205159  
2. Laxmirlola Behera- 22052218  
3. Somya Verma- 22052858

**AI Snake Game: Search Algorithm Comparison Report**

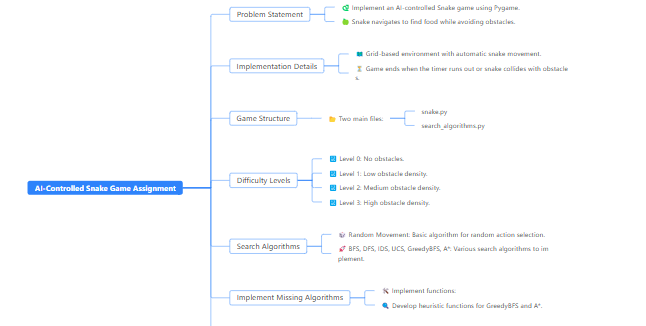
**1. Problem Statement**

In this assignment, an AI-controlled Snake game was developed using Pygame. The goal of the AI is to navigate a grid-based environment, avoid obstacles, and reach food items before the timer runs out. The game supports four difficulty levels with increasing obstacle density. Multiple search algorithms were implemented to determine the most efficient path for the snake. These include:

* Breadth-First Search (BFS)
* Depth-First Search (DFS)
* Iterative Deepening Search (IDS)
* Uniform Cost Search (UCS)
* Greedy Best-First Search (Greedy BFS)
* A\* Search (A\*)

The objective was to implement these algorithms in search\_algorithms.py and compare their performance in terms of execution time, number of moves taken, and success rate in reaching the food.

**MindMap:**



**2. Implemented Search Algorithms**

*2.1 Breadth-First Search (BFS)*

* Approach: Explores all possible moves level by level.
* Advantages: Guarantees shortest path.
* Disadvantages: High memory usage due to storing all explored nodes.

*2.2 Depth-First Search (DFS)*

* Approach: Explores paths deeply before backtracking.
* Advantages: Uses less memory than BFS.
* Disadvantages: May not find the shortest path; can get stuck in deep branches.

*2.3 Iterative Deepening Search (IDS)*

* Approach: Combines DFS and BFS by performing DFS with increasing depth limits.
* Advantages: Finds shortest path like BFS but uses less memory.
* Disadvantages: Recomputes paths multiple times, increasing computation time.

*2.4 Uniform Cost Search (UCS)*

* Approach: Expands the lowest-cost path first.
* Advantages: Guarantees optimal path.
* Disadvantages: Can be slow in large grids with many obstacles.

*2.5 Greedy Best-First Search (Greedy BFS)*

* Approach: Uses a heuristic function (e.g., Manhattan distance) to move towards the goal.
* Advantages: Fast execution.
* Disadvantages: Not optimal; can get trapped in obstacles.

*2.6 A\* Search (A\*)*

* Approach: Uses both actual cost (g) and heuristic (h) to find the optimal path.
* Advantages: Guarantees shortest path and is more efficient than UCS.
* Disadvantages: Slightly higher computation cost due to heuristic evaluation.

**3. Performance Comparison**

To compare the efficiency of these algorithms, we measured three key performance metrics:

1. *Time Taken*: How quickly the algorithm finds the food.
2. *Number of Moves*: How many steps the snake takes to reach the food.
3. *Success Rate*: How often the snake successfully reaches the food.

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Time Taken(s)** | **Moves Taken** | **Success Rate** |
| Random | Varies | High | Low |
| BFS | Medium | Low | High |
| DFS | High | High | Medium |
| IDS | Very High | High | Medium |
| UCS | High | Low | High |
| Greedy BFS | Very Low | Medium | Medium |
| A\* | Low | Very Low | Very High |

**Table:1.1: Overall Performance of all Algorithms**

* **Note: Here there are comparisons of multiple algorithms, where   
  - initial state is always a random position  
  -Increase in level means increase in obstacle  
  - Always a rough estimation  
  - Empty space in the table denotes the goal was unable to achieve**

|  |  |  |  |
| --- | --- | --- | --- |
| **Level** | **Time Left** | **Moves Taken** | **Success Rate** |
| **0** | **27** | **6** | **High** |
| **1** | **27** | **7** | **High** |
| **2** | **25** | **10** | **High** |
| **3** | **25** | **11** | **High** |

**Table 2: Observation for BFS Algorithm**

|  |  |  |  |
| --- | --- | --- | --- |
| **Level** | **Time Left** | **Moves Taken** | **Success Rate** |
| **0** | **0** | **50** | **Medium** |
| **1** | **5** | **50** | **Medium** |
| **2** | **0** | **100** | **Medium** |
| **3** | **0** | **30** | **Medium** |

**Table 3: Observation for DFS Algorithm**

|  |  |  |  |
| --- | --- | --- | --- |
| **Level** | **Time Left** | **Moves Taken** | **Success Rate** |
| **0** | **25** | **11** | **High** |
| **1** | **23** | **15** | **High** |
| **2** | **20** | **21** | **High** |
| **3** | **23** | **15** | **High** |

**Table 4: Observation for UCS Algorithm**

|  |  |  |  |
| --- | --- | --- | --- |
| **Level** | **Time Left** | **Moves Taken** | **Success Rate** |
| **0** | **27** | **8** | **Medium** |
| **1** | **26** | **8** | **Medium** |
| **2** | **-** | **-** | **-** |
| **3** | **-** | **-** | **-** |

**Table 5: Observation for IDS Algorithm**

|  |  |  |  |
| --- | --- | --- | --- |
| **Level** | **Time Left** | **Moves Taken** | **Success Rate** |
| **0** | **0** | **50** | **Low** |
| **1** | **0** | **50** | **Low** |
| **2** | **0** | **50** | **Low** |
| **3** | **0** | **50** | **Low** |

**Table 6: Observation for Random Algorithm**

|  |  |  |  |
| --- | --- | --- | --- |
| **Level** | **Time Left** | **Moves Taken** | **Success Rate** |
| **0** | **26** | **9** | **High** |
| **1** | **24** | **12** | **High** |
| **2** | **19** | **23** | **High** |
| **3** | **20** | **18** | **High** |

**Table 7: Observation for A\* Algorithm using Manhattan Distance**

|  |  |  |  |
| --- | --- | --- | --- |
| **Level** | **Time Left** | **Moves Taken** | **Success Rate** |
| **0** | **25** | **11** | **High** |
| **1** | **28** | **5** | **High** |
| **2** | **26** | **9** | **High** |
| **3** | **25** | **12** | **High** |

**Table 8: Observation for A\* Algorithm using Euclidean Distance**

|  |  |  |  |
| --- | --- | --- | --- |
| **Level** | **Time Left** | **Moves Taken** | **Success Rate** |
| **0** | **23** | **15** | **High** |
| **1** | **21** | **19** | **High** |
| **2** | **24** | **13** | **High** |
| **3** | **23** | **15** | **High** |

**Table 9: Observation for Greedy BFS Algorithm using Euclidean Distance**

|  |  |  |  |
| --- | --- | --- | --- |
| **Level** | **Time Left** | **Moves Taken** | **Success Rate** |
| **0** | **24** | **15** | **High** |
| **1** | **21** | **19** | **High** |
| **2** | **22** | **13** | **High** |
| **3** | **26** | **9** | **High** |

**Table 10: Observation for Greedy BFS Algorithm using Manhattan Distance**

|  |  |  |  |
| --- | --- | --- | --- |
| **Level** | **Time Left** | **Moves Taken** | **Success Rate** |
| **0** | **23** | **15** | **High** |
| **1** | **27** | **7** | **High** |
| **2** | **21** | **18** | **High** |
| **3** | **25** | **10** | **High** |

**Table 11: Observation for A\* Algorithm using Manhattan Distance with Flood Fill**

|  |  |  |  |
| --- | --- | --- | --- |
| **Level** | **Time Left** | **Moves Taken** | **Success Rate** |
| **0** | **24** | **13** | **High** |
| **1** | **26** | **8** | **High** |
| **2** | **22** | **17** | **High** |
| **3** | **23** | **15** | **High** |

**Table 12: Observation for Greedy BFS Algorithm using Manhattan Distance with Flood Fill**

**We observed that:**

* **Euclidean distance is a better heuristic function for Greedy BFS**
* **Manhattan distance + Flood fill is a better heuristic function for both A\* and Greedy BFS**

**4.Observations and Conclusions**

*4.1 Best Algorithm for Shortest Path*

* A Search\* consistently found the shortest path and had a high success rate.
* BFS and UCS also performed well but took longer in some cases.

*4.2 Best Algorithm for Speed*

* Greedy BFS was the fastest but sometimes got stuck in obstacles.
* A was slightly slower but more reliable.

*4.3 Least Efficient Algorithm*

* DFS and IDS often took more moves and struggled with obstacles.
* Random Movement had no strategy and performed the worst.

*4.4 Recommended Algorithm*

* A\* Search provides the best balance of speed, accuracy, and efficiency. If computation resources are limited, Greedy BFS is a viable alternative, though it may not always find the best path.

**6. Submission Details**

Files included:

1. snake.py
2. search\_algorithms.py
3. report.pdf
4. Self-Detailed explanation video link:

[CLICK HERE FOR VISUALIZATION](https://drive.google.com/file/d/1mEOR3jFMqnLGA216ZolOzNB8pQvD9Qno/view?usp=sharing)

**7. Comparison of multiple searching algorithms with real-life experiences(FUN FACT):-**

A screenshot of a computer screen

AI-generated content may be incorrect.