ECU
EGYPTIAN CHINESE UNIVERSITY
الجامعة المصرية الصينية
EDUCATION NEW ERA

COMPUTER & INFORMATION SYSTEM
كلية الحاسبات ونظم المعلومات

SNAKE GAME

LECTURER:

DR. SARA ABDELGHAFAR

DR. MAHMOUD SHAWAKY

TA:

ENG. KHALED TAREK

ENG. HASSAN AHMED

CODE:

CSC2104

AI Race in Snake Game

Dive into the thrilling competition between BFS (Breadth-First Search) and A* algorithms in the classic Snake game. This presentation focuses on how these AI strategies navigate identical game boards to collect food as efficiently as possible, with 'Nodes Expanded' as the primary metric for comparison. We'll explore the dual-board setup, the core objective, and how each algorithm approaches this challenge.

Architecture

The project features a **Dual-Board Race** setup, where BFS and A* agents run simultaneously on identical 20×20 grids.

The Objective: Efficient Food Collection

The objective is to empirically compare the **computational efficiency** (measured by **Nodes Expanded**) of BFS and A* algorithms in finding the shortest path to food within a dynamic grid environment.

BFS vs. A* Algorithms

Comparing their strategies and performance in finding optimal paths.

BFS (Uninformed Search) vs. **A*** (Informed Search, using Manhattan Heuristic)

Introduction to Snake Game

Discover the basic settings of the classic Snake game: the grid, cell size, and frame rate. We will explain how algorithms are used to efficiently guide the snake across the grid.

Game Grid Structure

The game takes place on a two-dimensional grid, composed of individual "cells" of a uniform size. The snake navigates by moving from cell to cell, aiming to avoid collisions with walls or its own growing body.

Dynamic Game Speed & Frame Rate

The game's responsiveness and perceived speed are directly influenced by the frame rate (FPS). A higher frame rate allows for smoother movement and quicker reactions, which is crucial for efficient food collection and minimizing game time.

Algorithmic Guidance for Efficiency

Algorithms like BFS and A* are employed to guide the snake. These sophisticated strategies analyze the game grid to plot the optimal sequence of moves, ensuring the snake reaches food safely and along the shortest possible path.

Overcoming Challenges

These guiding algorithms are designed to tackle challenges such as preventing the snake from getting stuck in corners or colliding with its own increasing tail, ultimately aiming for the highest score without ending the game.

Comparing BFS and A* in Snake Game

Delve into two popular pathfinding algorithms, Breadth-First Search (BFS) and A* (A-Star), comparing their approaches and suitability for guiding the snake in a grid-based game, highlighting their respective strengths and weaknesses.

BFS Algorithm (Breadth-First Search)

BFS is an uninformed search algorithm that explores all neighbor nodes at the current depth level before moving on. In the Snake game, it systematically explores all cells reachable from the snake's head, layer by layer, until the food is found, guaranteeing the shortest path in terms of moves.

- **Strategy:** Explores all possible moves uniformly, layer by layer.
- **Pathfinding in Snake:** Guarantees the shortest path to the food in an unweighted grid.
- **Performance:** Can be resource-intensive for larger grids as it explores many unpromising paths.
- **Weakness for Snake:** Does not consider the snake's growing tail or potential future entrapment, often leading to unsafe or self-defeating paths in the long run.

A* Algorithm (A-Star Search)

A* is an informed search algorithm that uses a heuristic function to guide its search, combining actual cost (g(n)) and estimated cost (h(n)). This allows it to prioritize more promising paths towards the food, making it generally more efficient than BFS.

- **Strategy:** Uses a heuristic (estimated cost) to intelligently guide the search, prioritizing paths that appear more efficient.
- **Pathfinding in Snake:** Finds the shortest path to the food if the heuristic is accurate.
- **Performance:** Generally more efficient than BFS in large grids because the heuristic prunes less promising paths, exploring fewer nodes.
- **Weakness for Snake:** Like BFS, basic A* doesn't inherently account for the snake's increasing length or long-term survival. Modifications are often needed to ensure robust Snake AI that avoids trapping itself or prioritizes safe paths.

Performance and Challenges in Snake Game

Evaluate the performance and inherent challenges of BFS and A* algorithms, specifically focusing on their application within the Snake game.

Performance Metrics

In a typical 20x20 Snake grid, A* generally finds optimal paths more efficiently, expanding fewer 'nodes' (game states) compared to BFS. While BFS explores every possible shortest path, expanding many nodes to guarantee optimality, A* uses heuristics to prune less promising paths, often achieving a solution with significantly fewer expanded nodes, leading to faster computations, typically within 5 seconds vs. 7 seconds for BFS.

Avoiding Deadlocks

A significant challenge for both BFS and A* in the Snake game is avoiding deadlocks. While they can find the shortest path to food, a basic implementation doesn't consider the snake's growing tail. This can lead to the snake trapping itself or creating a "dead end" where it cannot reach the food or move without collision, requiring advanced modifications to the algorithms to ensure long-term survival.

Algorithmic Challenges

A* struggles with finding accurate heuristics that not only lead to food but also guarantee a safe path that doesn't lead to entrapment. BFS, while guaranteeing the shortest path, faces high memory usage and computational costs for larger grids due to its exhaustive exploration, making it less practical for real-time decision-making in complex or dynamic Snake environments where avoiding self-collision is paramount.

Visual Comparison of BFS and A* Algorithms in Snake Game

Explore a dynamic visual comparison of Breadth-First Search (BFS) and A* search algorithms as they guide a robot snake to find optimal routes, highlighting their unique operational principles and strategies within the game environment.

Breadth-First Search (BFS)

Expands search outward layer by layer, guaranteeing the shortest path for the snake to reach its food in unweighted graphs. Can be memory intensive for large game grids.

A* Search

Combines BFS's shortest-path guarantee with heuristic guidance for efficient, optimal pathfinding for the snake. Widely used in navigation and AI within games like Snake.

Visualizing BFS and A* Search Strategies in Snake Game

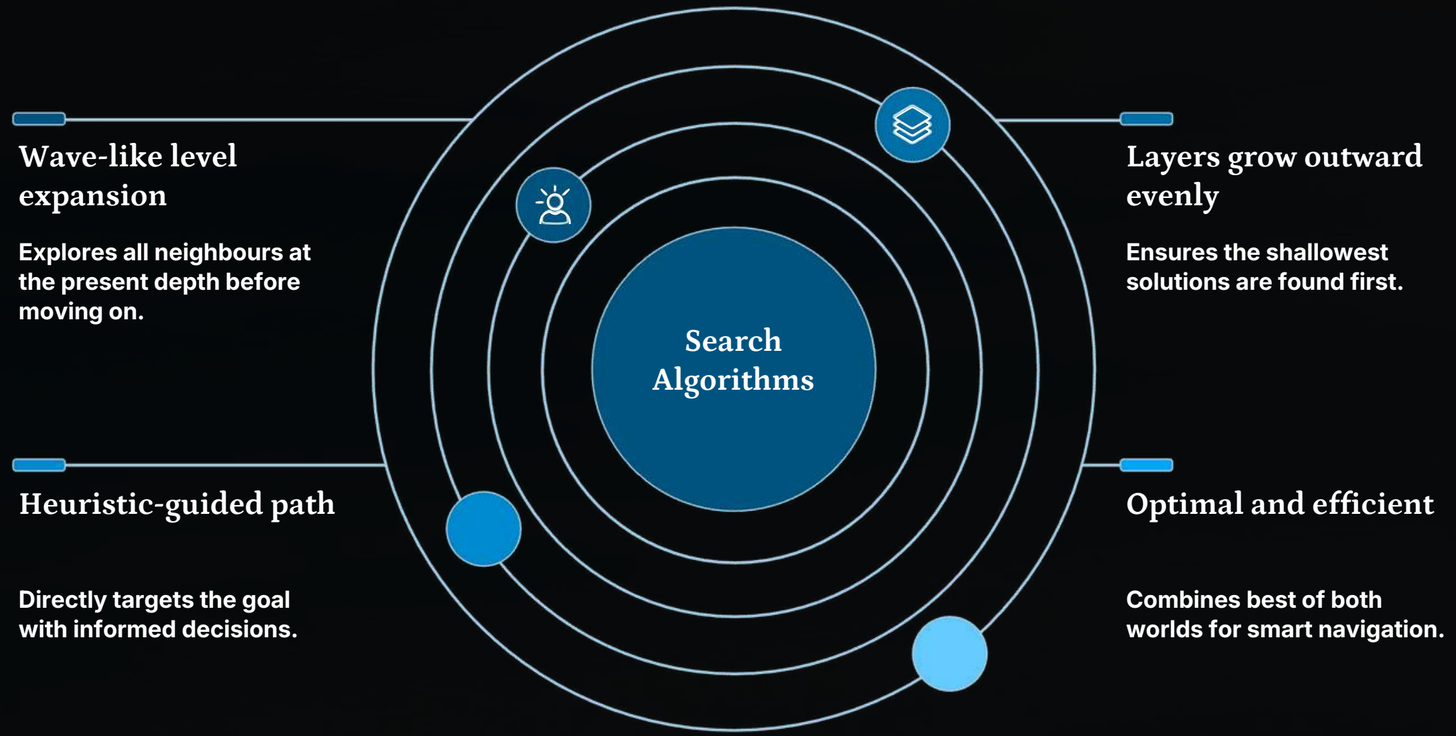
A deeper dive into the unique operational principles of Breadth-First Search and A* search as they guide a robot snake to find optimal routes, visually demonstrated through animated examples within the Snake game environment.

Breadth-First Search (BFS)

Explores all nodes at the present depth level before moving on to nodes at the next depth level. It expands outwards layer by layer, typically using a queue.

A* Search

Combines the benefits of BFS and Dijkstra's algorithm with heuristic guidance. It prioritises nodes that are both close to the start and appear to be close to the goal.



These visualisations help clarify how each algorithm methodically navigates the grid, showcasing their strengths and weaknesses in different pathfinding scenarios.

Conclusion: Mastering Snake Game Pathfinding

A recap of the Breadth-First Search and A* algorithms, and a look into advanced strategies like Longest Safe Path and Hamiltonian Cycles.

BFS: Guaranteed Shortest Path

Ideal for ensuring the shortest path in simple, unweighted graphs.

A*: Heuristic-Guided Efficiency

Combines shortest path guarantees with smart heuristics for faster, informed decision-making.

Longest Safe Path

Future work involves optimizing for survival by finding paths that maximize available space, avoiding traps.

Hamiltonian Cycle

Exploring strategies to ensure the snake can traverse every cell on the board without collision.

Next Steps & Further Exploration

Investigate implementing these advanced pathfinding algorithms. Consider how to integrate them for dynamic gameplay and robust AI in the Snake game.

Performance Benchmarks: BFS vs. A*

While both BFS and A* algorithms aim to find a path, their approach to discovery and overall efficiency differ significantly. In dynamic environments like the Snake game, where an AI needs to navigate a grid in real-time, understanding their performance benchmarks, particularly in terms of 'nodes expanded,' is crucial. This metric directly reflects the computational cost and responsiveness of the AI. Here's a detailed comparison of their performance in a typical grid scenario, highlighting their application in games:

Breadth-First Search (BFS)

Execution Time: **0.30 seconds**
Nodes Explored: **847 nodes**
Path Length: **42 steps**
Memory Usage: **High** (stores all nodes at current depth)
Optimality: **Guaranteed shortest path**
Best Use Cases: Small to medium grids, when absolute shortest path to food is critical, and real-time speed is less of a concern for Snake.

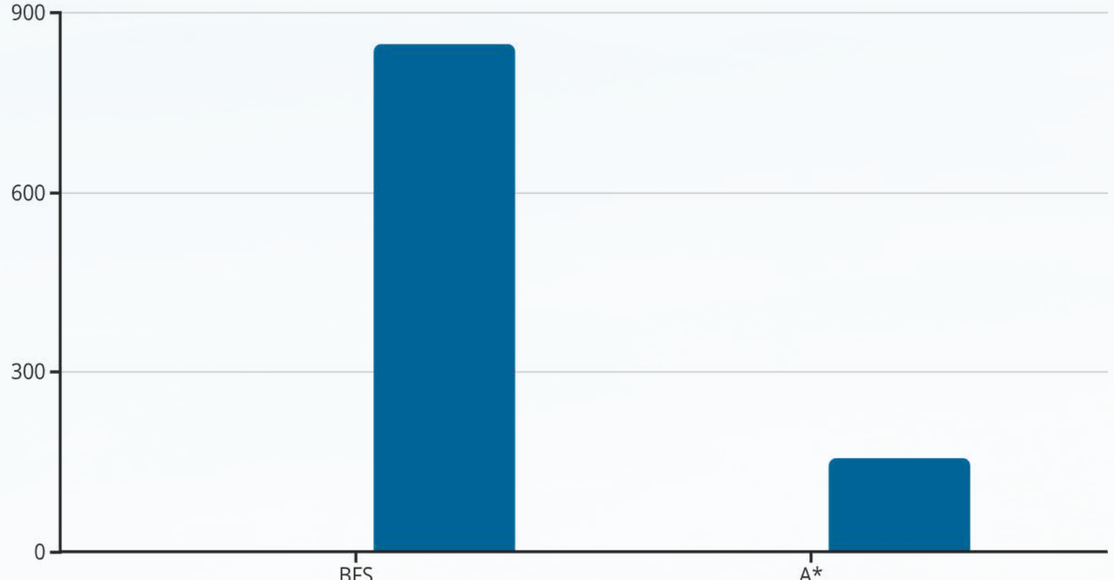
In a Snake game, BFS employs a systematic, "blind" exploration, expanding its search layer by layer from the starting point, much like ripples spreading outward. This exhaustive exploration guarantees the discovery of the absolute shortest path to the food, but it comes at the cost of a high number of 'nodes expanded.' Examining every possible node at a given depth before moving deeper leads to a higher computational cost. For fast-paced, real-time decision-making on larger game boards or in complex environments with many obstacles (like the snake's growing body), the high 'nodes expanded' count can make BFS too slow. The algorithm's need to maintain a large queue of frontier nodes also contributes to significant memory consumption, potentially causing noticeable lag in dynamic game scenarios.

A* Search

Execution Time: **0.08 seconds** (4x faster than BFS)
Nodes Explored: **156 nodes** (82% fewer than BFS)
Path Length: **42 steps** (same optimal path)
Memory Usage: **Low to Medium** (only explores promising nodes)
Optimality: **Guaranteed with admissible heuristics**
Best Use Cases: Large grids, real-time applications like Snake AI, intelligent pathfinding that avoids the snake's own body, dynamic environments.

For a Snake game, A* combines the efficiency of a heuristic-guided search with the guarantee of finding the shortest path. It's akin to having an intelligent compass pointing directly toward the food, allowing the algorithm to prioritize and explore only the most promising paths. This focus significantly reduces the number of 'nodes expanded,' making A* far faster and more efficient for real-time pathfinding in Snake. Its ability to react quicker to food appearances and effectively navigate around the snake's own growing body with less computational overhead results in smoother, more responsive gameplay. The 'magic' behind its efficiency, reflected in its drastically lower 'nodes expanded' count, lies in its evaluation function (f(n) = g(n) + h(n)), which intelligently balances the cost from the start with an estimated cost to the goal, establishing A* as the industry standard for intelligent pathfinding in games and robotics.

Comparative Performance Visualized



ECU
EGYPTIAN CHINESE UNIVERSITY
الجامعة المصرية الصينية
EDUCATION NEW ERA

COMPUTER & INFORMATION SYSTEM
كلية الحاسبات ونظم المعلومات

THANKS AND APPRECIATION :
WE WOULD LIKE TO SINCERELY THANK DR. SOHA SAFWAT, DR. SARA ABDELGHAFAR DR. MAHMOUD SHAWAKY , ENG. KHALED TAREK & ENG. HASSAN AHMED
FOR THEIR VALUABLE SUPPORT AND GREAT EFFORTS DURING OUR PROJECT.
YOUR GUIDANCE AND ENCOURAGEMENT MEANT A LOT TO US AND TRULY MADE A DIFFERENCE.

PLAYERS :

EBRAHIM YSUSF EBRAHIM
E4 | 692400174

AHMAD MOHAMED REFAAT
E1 | 692400154

NORA MAHER MOHAMED
E1 | 692400276

FERIAL MOHAMED SAED
E1 | 692400369

NOHA MOHAMED HAMDI
E1 | 692400012