Burnt Pancake Solver

Corey Zhang

University of Delaware

Course: CISC 681/481

Introduction

The Burnt Pancake puzzle challenges you to sort a stack of four pancakes by size which smallest on top, largest on bottom while ensuring every burnt side faces down. Only prefix flips are allowed: choosing the top *k* pancakes, reversing their order, and flipping each pancake over. This project implements two classic search algorithms, Breadth-First Search (BFS) and A* search, to find the optimal sequence of flips for any starting configuration.

Implementation

I built a clear, modular Python program to solve this puzzle. The core function, flip(state, position), takes the current stack representation (an 8-character string like “1b2w3b4w”) and a flip position *k*, then reverses and side-flips the top *k* pancakes. To guide the A* search, heuristic(state) computes the size of the largest pancake out of place, providing a simple but effective estimate of remaining work.

The bfs(start) function performs an unweighted breadth-first exploration: it keeps a FIFO queue of (state, path) pairs and tracks visited states to avoid repeats. In contrast, astar(start) maintains a list of (f, g, state, path) tuples, where g is the flip cost so far and f = g + h uses our heuristic. Each iteration sorts the queue with a helper bubble_sort and expands the state with the lowest f value. Both solvers build a path of (state, flip_position) steps that leads from the initial configuration to the solved stack.

Once a solution path is found, the printa and printb routines display each intermediate stack, marking the flip point with a “|”. For A*, each line shows the cumulative cost g and the heuristic value h; for BFS, the sequence of flips is displayed without cost annotations.

Input & Output

Users enter an input string such as "1b2w3b4w-a", where each digit 1–4 represents pancake size, w or b indicates clean or burnt side up, and -a or -b selects A* or BFS. The solver then prints a sequence of stack states showing exactly how to flip the pancakes into the correct order.

Results

In tests on the course server, both algorithms reliably solved every configuration. As expected, A* explored far fewer states than BFS thanks to the heuristic, demonstrating the power of informed search even with a simple estimate.