Burnt Pancake Solver

Corey Zhang

University of Delaware

Course: CISC 681

Introduction

The Burnt Pancake puzzle challenges you to sort a stack of four pancakes by size—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 eight-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

Input: 1w2b3w4b-a (A* search)
Output:

1w2b|3w4b g:0, h:0
2w|1b3w4b g:2, h:2
2b1b3w4b| g:3, h:2
4w|3b1w2w g:7, h:4
4b3b1w2w| g:8, h:4
2b1b|3w4w g:12, h:2
1w2w3w4w g:14, h:0

Input: `1w2b3w4b-b` (BFS)
Output:

1w2b|3w4b
2w|1b3w4b
2b1b|3w4b
1w2w3w4b|
4w|3b2b1b
4b3b2b1b|
1w2w3w4w

Input: `1b2b3w4b-a` (A* search)
Output:

1b2b|3w4b g:0, h:0
2w|1w3w4b g:2, h:2
2b1w3w4b| g:3, h:2
4w|3b1b2w g:7, h:4
4b3b1b2w| g:8, h:4
2b1w|3w4w g:12, h:2
1b|2w3w4w g:14, h:0
1w2w3w4w g:15, h:0

Input: `1w2b3b4w-b` (BFS)
Output:

1w2b|3b4w
2w|1b3b4w
2b1b|3b4w
1w2w3b|4w
3w|2b1b4w
3b2b1b|4w
1w2w3w4w

Input: `1b2w3b4b-a` (A* search)
Output:

1b2w3b4b| g:0, h:0
4w|3w2b1w g:4, h:4
4b3w|2b1w g:5, h:4
3b|4w2b1w g:7, h:4
3w4w|2b1w g:8, h:4
4b3b2b1w| g:10, h:4
1b|2w3w4w g:14, h:0
1w2w3w4w g:15, h:0