Course: CS131 Artificial Intelligence

Assignment: Informed Search

Name: Mingwei Cui

A*:

Design:

A* search to sort the pancakes will employ a priority queue, which is implemented using heap and keeps the list of potential nodes to be visited. Based on its total cost, each node is assigned to the priority queue. A node is a representation of a path to a state. If the total cost of two nodes is the same, the node introduced first has higher priority. From the largest on the bottom to the smallest on the top, my understanding is that the pancake is stacked, thus the smallest pancake on top is output first.

Function:

1. Backward cost: Defined by the number of pancakes have been flipped
2. Heuristic function: Defined by the number of stack positions for which the pancake at that position is not of adjacent size to the pancake below
3. Total cost: Backward cost + Heuristic cost
4. Priority queue: Sort the queue (implemented by heap) according to the total cost, so smaller cost node has higher priority.
5. Expansion: Add to the heap any conceivable node that the present node can reach. Each new node shows how the stack might look following each conceivable flip.
6. Verify: Check if the heap is empty
7. Solution: show the steps that how the algorithm flips the pancake

Uniform Cost:

Design:

Uniform Cost search to sort the pancakes will employ a priority queue, which is implemented using heap and keeps the list of potential nodes to be visited. Based on its Backward cost, each node is assigned to the priority queue. A node is a representation of a path to a state. If the total cost of two nodes is the same, the node

introduced first has higher priority. From the largest on the bottom to the smallest on the top, my understanding is that the pancake is stacked, thus the smallest pancake on top is output first.

Function:

1. Backward cost: Defined by the number of pancakes have been flipped
2. Heuristic function: None
3. Total cost: Just backward cost
4. Priority queue: Sort the queue (implemented by heap) according to the total cost (backward cost), so smaller cost node has higher priority.
5. Expansion: Add to the heap any conceivable node that the present node can reach. Each new node shows how the stack might look following each conceivable flip.
6. Verify: Check if the heap is empty
7. Solution: show the steps that how the algorithm flips the pancake

Test Part:

Input State: "5, 6, 1, 7, 2, 4, 3, 8, 9, 10"

A* search uses 6 steps and 0.37s to obtain the right order while Uniform Cost search uses 6 steps and 2.45s to obtain the right order.

Input State: "7, 6, 1, 8, 2, 4, 3, 5, 9, 10"

A* search uses 7 steps and 3.87s to obtain the right order while Uniform Cost search uses 7 steps and 29.68s to obtain the right order.

Input State: "5, 6, 1, 8, 2, 4, 3, 7, 9, 10"

A* search uses 7 steps and 47.68s to obtain the right order while Uniform Cost search uses 7 steps and 357.95s to obtain the right order.

The time complexity of this A* search algorithm grows exponentially, so if we try more input number or make the input state more out-of-order, especially if the last number is not the largest one, it would cost so much more time.