Yige Sun

0.1 Problem to Solve

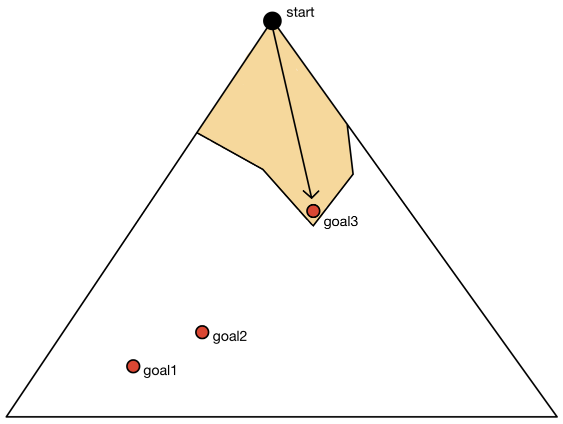
In this project, we will try to solve pancake problem. Pancake problem is: we are given a disordered stack of 7 differently sized pancakes (size from 1 to 7) and a spatula that can be inserted at any point in the stack and used to flip all pancakes above it. The goal is for the cook to have them in the "correct" order for the customer, that is, the large on the bottom up to the smallest on top ([7, 6, 5, 4, 3, 2, 1]).

0.2 How to Solve

I will use dynamic program(A star algorithm and uniform cost search algorithm) to solve pancake problem.
In order to use dynamic programming to get optimal solution for pancake problem, we need to keep 3 memos of trees:

- Forward Function: the number of stack positions for which the pancake at that position is not of adjacent size to the pancake below.
- Backward Cost Function: how many times we need to flip pancakes to get the node from initial state.
- Total Function = Forward Function + Backward Function.

AStar method recursively do next flip base on the lowest result of Total Function and UCS method will recursively do next flip base on the lowest result of backward cost function. Imagining what we did in greedy algorithm, we keep all previous steps, current step and all possible next steps as nodes in one tree; we can draw steps of UCS method and AStar in this fashion. Then we can get the figure:



In the picture above, we start searching from black node and try to find a path to arrive at red node. The orange space contains the nodes we visited before. Because Pancake Problem always exist one solution and we keep flipping base on the lowest result of total function or backward cost function, we will always get the optimal solution which means we will arrive at goal3.
Using AStar method, the time complexity would be $O(b^m)$ and the space complexity would be $O(b^m)$, where b denotes the branch of the tree and m denotes the maximum depth of the tree since the goal may be at the bottom of the tree. Using UCS method, the time complexity will be $O(b^{1+\frac{C'}{\epsilon}})$ and the space complexity will be $O(b^{1+\frac{C'}{\epsilon}})$, where ϵ denotes the lowest 'cost' in the all 'cost' generated by nodes and c' denotes the 'cost' of the final solution.

0.3 How to Build

i. We build heuristic function to calculate the 'cost'(i.e. forward function and backward cost function) of nodes we reached so far.

ii. We construct a flip function to simulate flip operation(flip all pancakes above one position).

iii. We calculate the total function for A star algorithm.

iv. Every time we expand one node, we will choose the next node with lowest 'cost'(for A star, the lowest 'cost' is lowest output of total function; for uniform cost search, the lowest 'cost' is lowest output of backward cost function).

v. Whole procedure:

   a) Append initial state into priority queue.
   b) While the solution is not found, we repeatedly execute actions from c) to e).
   c) If priority queue is empty which means we have nothing to expand although we didn't find a solution. Then return fail.
   d) If priority queue is not empty, we pop the node with lowest 'cost' out from the queue and then add its child nodes into queue.
   e) Sort the priority queue based on the 'cost' of nodes, this will guarantee we can pop the node with smallest 'cost'.
   f) Get and show solution we find.

0.4 How to Run

Please run main.py file on pycharm. Make sure the pancake size from 1 to 7. My testing python version is 3.9.

0.5 Conclusion

First I try the examples and get the number of flips we need to do using different methods:

Examples	Number of Flips by Using AStar Method	Time Use by AStar Method	Number of Flips by Using UCS Method	Time Used by UCS Method
6,5,1,2,7,4,3	5	0.005s	5	8.337s
7,1,2,3,6,5,4	3	0.0003s	3	0.019s
3,2,5,1,6,4,7	5	0.0006s	5	14.1762s
5,2,4,1,3,7,6	7	0.0324s	7	252.338s

From the table above, we can clearly see that UCS method will need more time to get the right solution and AStar method will need less time. Actually, in this problem UCS will need incredible time to get the answer since every time we expand one node, the cost will always +1, we are actually doing a breath first search. Since both UCS method and AStar method always expand the node which has the lowest 'cost', the solutions generated by them will be optimal solution. However, since AStar method have 1 extra heuristic function(forward function) and this function will 'guide' searching process, the AStar method will take less time to find the optimal solution. Besides two algorithms, I also learned from this project that although dynamic programming could help us to solve some problems easily, the performance of different algorithms varies a lot. It's better to have a 'guide' function(like forward function in this problem) when doing dynamic programming so that we can save more time.