To-do:

1.Define the problem as a searching problem

2.Define a possible cost function (backward cost)

3.Define a possible heuristic function (forward cost)

4.Implement an A\* algorithm

Bonus: Use Uniform-Cost-Search to implement

------------------------------------------------------------------

1.Define the problem:

* 1. So after reading the description I naturally think this question as a sorting problem rather than a

searching problem and here are some web pages that discuss it in detail:

<https://en.wikipedia.org/wiki/Pancake_sorting>

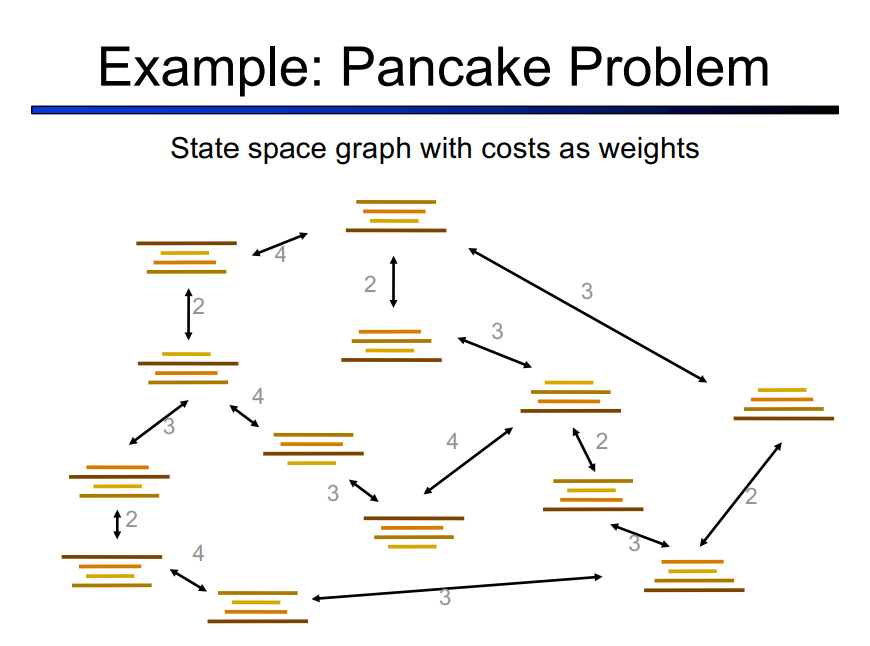
<https://leetcode.com/problems/pancake-sorting/>

<http://www.cs.cmu.edu/~arielpro/15251f17/notes/pancakes-notes.pdf>

It is a little bit similar to Hanoi tower problem, we need to build it from the bottom layer then do recursion on the rest until the top layer is done.

* 1. Then I want to transform this sorting problem into a searching graph, based on this article:

<https://courses.cs.washington.edu/courses/cse473/17sp/slides/03-hsearch-473.pdf>



I can define this problem as:

1) Initial state: the initial pancake stack I am given

2) Goal state: the ordered (smaller on top, bigger on bottom) pancake stack

3) Other states: all possible pancake stacks other than these two

4) Possible actions: every possible stack I can turn my current stack into by flipping it

5) Successor function: what shape will the current stack turn into

6) Goal test: have I reached the goal stack or not

7) Path cost function: basically 1 (each flip costs 1 or any constant effort)

2.Define backward cost:

Backward cost is how many flips I have done so far, in other words, how many states I have already visited.

3.Define heuristic function:

1) The strategy I will be using is tree search:

from initial stack, choose a possible child note for expansion according to the cost function, if the node is the goal state then the algorithm stops and return the total cost, else keep expanding the node and add the resulting nodes to the search tree

2) Heuristic function:

the gap method from the paper *Landmark Heuristics For The Pancake Problem*

4.Implement A\* and UCS algorithms:

see the source code