Credits: Slides originally by Jin Zhe

CS2040S

Recitation 2
AY24/25S2

Question 1

Problem

Input: Given a stack of pancakes with varying sizes

Output: Order it with the smallest on top and the biggest at the bottom

Constraint: You can only flip pancakes from the top

Problem 1.a.

Given a stack of n pancakes, how many flips (in terms of n) does it take to order it?

Guiding question

How many flips does it take to get a specific pancake to the bottom of the stack?

Guiding question

How many flips does it take to get a specific pancake to the bottom of the stack?

Answer: 2:

1. Flip from target pancake to bring it to the top
2. Flip entire stack to bring pancake to the bottom

Guiding question

What is the base case? At worst, how many flips are required to sort it?

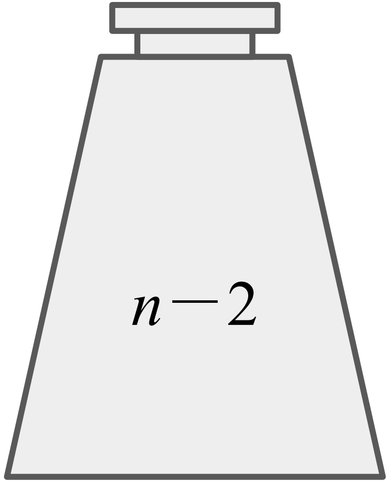
Guiding question

What is the base case? At worst, how many flips are required to sort it?

Answer: 2 pancakes. 1 flip need at worst.

Realise that base cases in pancake problems are special cases that require *less* flips than the usual cases and therefore require special treatment.

Flips needed



1 flip

$2(n-2)$ flips

Problem 1.b.

From your proposed pancake flipping strategy in the previous part, what is the invariant at each step of the sorting process?

Out of all the sorting algorithms covered in lectures so far, which one of them is the most analogous to your strategy?

Guiding question

What can you say about the problem after $2k$ flips according to our strategy?

Guiding question

What can you say about the problem after $2k$ flips according to our strategy?

Answer: After $2k$ flips, k pancakes are sorted at the bottom of the stack.

Guiding question

Which sorting algorithm(s) most closely resembles this invariance?

Guiding question

Which sorting algorithm(s) most closely resembles this invariance?

Answer: Max-selection sort or bubble-sort. Both these entail a sorted region at the end that grows by 1 with each completion of their subroutine.

Problem 1.c.

Now each pancake has a burnt side

You want to order it also with the burnt side facing down

How many flips do you need now?

Guiding question

Do we need a new strategy or can we built on the previous one? What has changed?

Guiding question

Do we need a new strategy or can we built on the previous one? What has changed?

Answer: We can build on the previous strategy. All we need is one extra step in the worst case: after flipping target package to the top, we need to flip it once before we flip the entire unsorted stack. This is to orientate it such that the burnt side is facing down in its final state.

Guiding question

What is the base case now? How many flips at worst does it need?

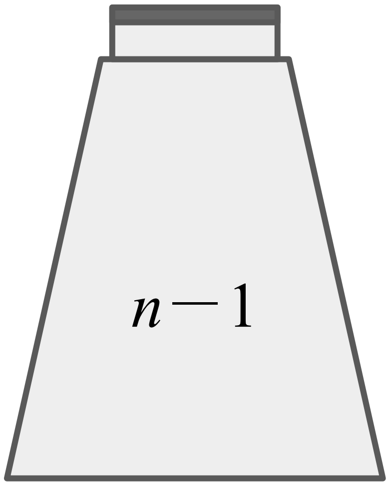
Guiding question

What is the base case now? How many flips at worst does it need?

Answer: Now the base case is just a single pancake which requires 1 flip to orientate its burnt side down.

The second last pancake cannot be ordered in less than 3 flips so it falls under the regular case.

Flips needed



1 flip

$3(n-1)$ flips

Problem 1.d.

What if there are only 2 sizes of pancakes?

How many flips do you need now?

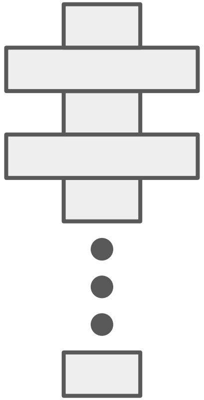
Guiding question

What is a possible worst case if there are only 2 sizes of pancakes?

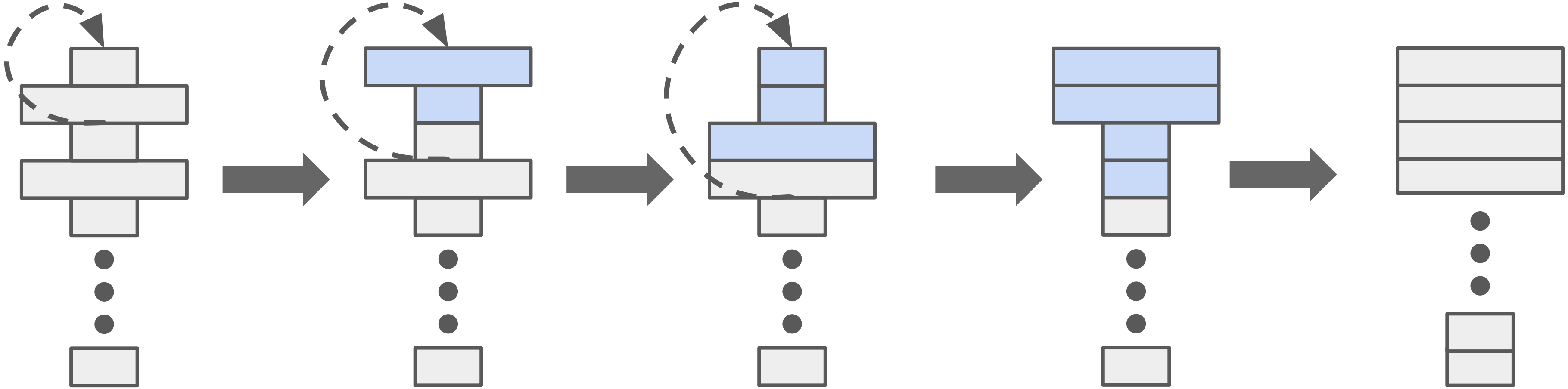
Guiding question

What is a possible worst case if there are only 2 sizes of pancakes?

Answer:



Solution



Realise that for the worst-case to happen, we would also need the smaller pancake to be the bottom-most so that one final flip would be needed in the end.

Flips needed

Starting from the top after the second pancake, we’ll have to flip every time we encounter the next pancake (which is of a different size than the previous).

This means we’ll have taken $n-2$ flips to reach the last pancake. However don’t forget that since the last pancake is a small one, we’ll need 1 final flip to orientate the entire stack. So total flips needed is $n-1$.

Find it hard to see? Do up a table and reason inductively.

Problem 1.e.

What if we want to order pancakes according to their skin textures instead of their sizes? I.e., the crunchiest (darkest in colour) on top and the fluffiest (lightest in colour) at the bottom?

Guiding question

Does this change the problem?

Guiding question

Does this change the problem?

Answer: No it doesn’t. Skin texture is just another abstraction of the measurement which we use to sort the pancakes. We simply need to relabel each pancake using their texture measurements.

Test yourself!

Do items in a collection need to be measurable in order for them to be sortable via comparison-based sorting?

Test yourself!

Do items in a collection need to be measurable in order for them to be sortable via comparison-based sorting?

Answer: No, they just need to be *mutually comparable*. E.g. I can sort a bag of apples and oranges if I say that oranges are *greater* than apples, without specifying any amounts attached to them (doesn’t matter how much greater). Look at how we implement custom comparator classes in Java.

Problem 1.f.

Show that any pancake sorting algorithm requires at least n

Flips.

What do we call such properties of the problem?

Lower bound

Definition: Complexity of the *best possible* solution we can have on the *worst possible case* input for the problem.

In other words, a ‘theoretical limit’ for a problem by which no solutions can be faster than.

Guiding question

What is one possible worst case input?

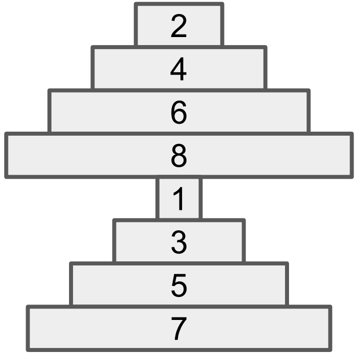
Hint: Consider adversarial inputs for sorting algorithms. The worst case is often ‘very close’ to the best case with a single property that is reversed/wrong.

Guiding question

What is one possible worst case input?

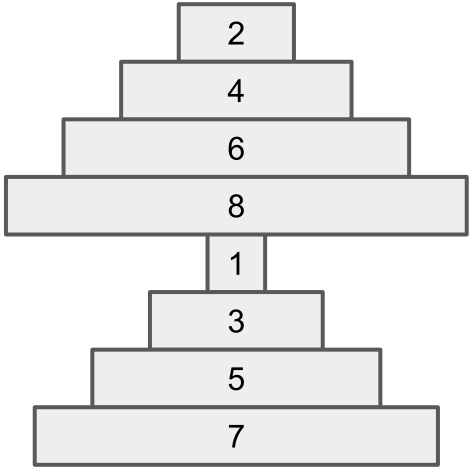
Hint: Consider adversarial inputs for sorting algorithms. The worst case is often ‘very close’ to the best case with a single property that is reversed/wrong.

Answer:



And possibly many more variants..

Worst case



For such worst case problems, we would *minimally* have to separate *every* consecutive pancake by inserting our spatula in between and them and flipping.

Guiding question

How many of such “inbetweens” do we have to flip in a worst case?

Guiding question

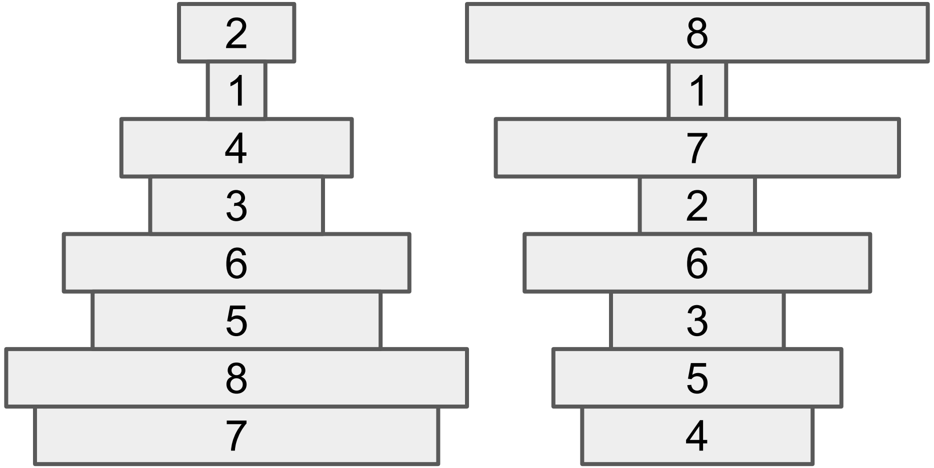
How many of such “inbetweens” do we have to flip in a worst case?

Answer:

Exactly n. Don’t forget the “inbetween” between the the bottom-most pancake and the table!

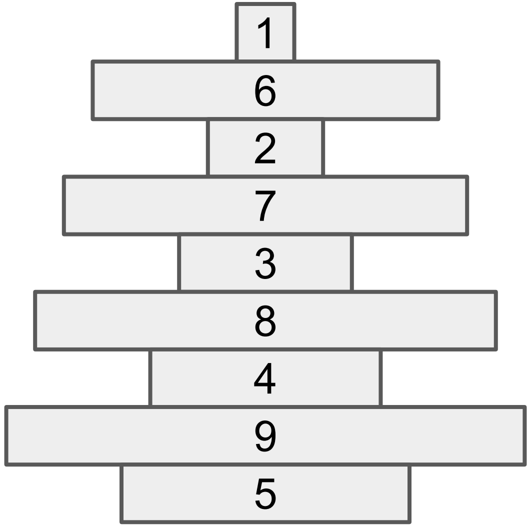
Trick: you can treat the table as a pancake of infinite size.

Worst cases?



These are not quite “worst” because there are consecutive pairwise pancake(s) in the stack, just in reversed order. A smart algorithm might not have to go between every consecutive pair of pancakes!

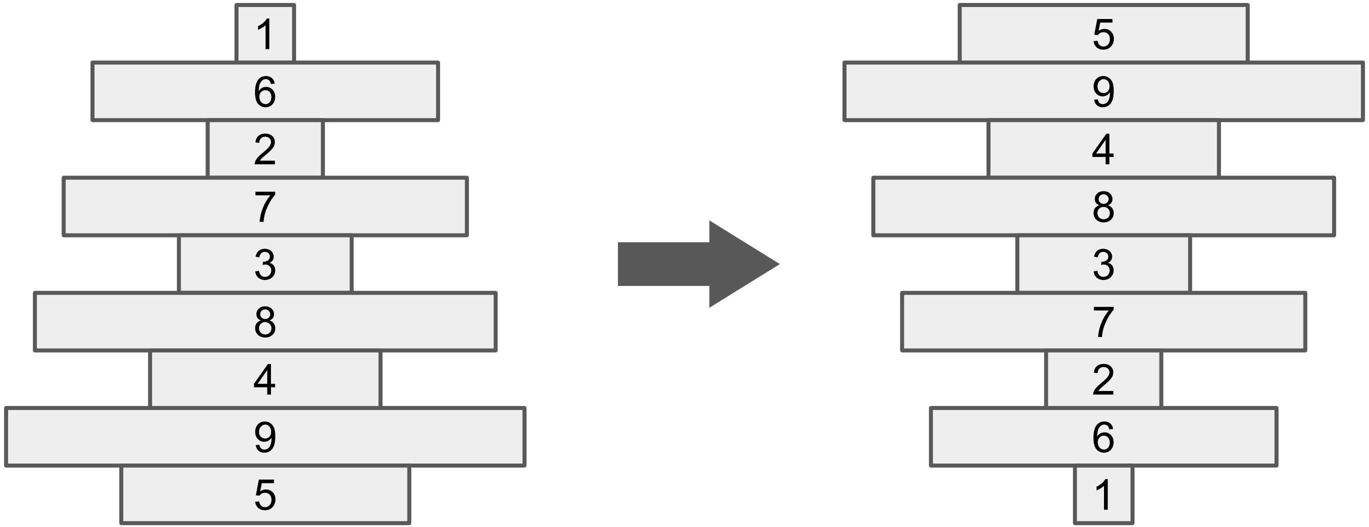
Worst cases?



Most of you came up with an interleaving version similar to this. One might feel that since pancake 1 is already in the correct position, perhaps this isn't "worst" enough?

Actually according to our analysis, this is also a worst case because we still need at least 9 separations :)

Worst cases?



We can also look at the flipped version to sidestep the discussion of whether or not having pancake 1 at the top makes it a not-so-worst case.

Thanks for pointing this out during class :)