1 Pancakes with a Problem

Suppose someone comes to you with this problem:[1]

> *The chef in our place is sloppy, and when he prepares a stack of pancakes they come out all different sizes. Therefore, when I deliver them to a customer, on the way to the table I rearrange them (so that the smallest winds up on top, and so on, down to the largest at the bottom) by grabbing several from the top and flipping them over, repeating this (varying the number I flip) as many times as necessary. If there are n pancakes, what is the maximum number of flips (in terms of n) that I will ever have to use to rearrange them?*

How do you solve it?

1.1 Step I: Getting the Problem Right

Well, perhaps the first thing to do is to make sure you understand the problem correctly. And nothing does that better than seeing a little example in action.

Suppose a five-pancake stack starts out in the following order:

(5 2 3 4 1)

Here's one way the waiter could sort it.

1. Flip the whole stack over. This puts the largest pancake on the bottom.

(1 4 3 2 5)

2. Flip the top four pancakes.

(2 3 4 1 5)

3. Flip the top three pancakes.

(4 3 2 1 5)

4. Flip the top four pancakes.

(1 2 3 4 5)

So, we know this stack can be sorted in four steps.

1.2 Step II: Choose the Right Representation

When playing with the examples up there, it is clear that drawing those pancakes is a bit of a pain — it takes time to get the relative sizes right, and is also a pain to argue precisely, and to write a solution down succinctly. So let's develop a notation to represent stacks: it'll make it easier to analyze and develop solutions. Plus, I don't really want to make diagrams for every stack in this set of notes. :)

[1] First posed as such in 1975 by Jacob E. Goodman (a.k.a. Harry Dweighter)

Since every pancake has a different size, let's rank the pancakes from 1 (smallest) to n (largest)—and denote pancake i with the number i. Then, we can even write a stack horizontally with the top pancake's rank written first, and the bottom pancake's rank last: e.g., a sorted stack would be (1 2 3 4 5), the starting configuration in the example above would be (5 2 3 4 1).

1.3 Step III: Let's Understand that Example a Little Better

We saw that (5 2 3 4 1) can be sorted in 4 flips. But is this the best we can do? Can we do it in 3 flips? Maybe? 2 flips? Hmm? 1 flip? zero?!?

That we know: the only stack that needs zero flips is a sorted stack, and (5 2 3 4 1) certainly isn't sorted.

In fact, one flip's not going to do either: you can try all single flips, and they yield:[2]

$$\begin{aligned}(5 \mid 2341) &\to (52341)\\ (52 \mid 341) &\to (25341)\\ (523 \mid 41) &\to (32541)\\ (5234 \mid 1) &\to (43251)\\ (52341 \mid) &\to (14325)\end{aligned} \tag{*}$$

In fact, observe that the only stacks that can be sorted using one flip looks like:

$$(i, i-1, i-2, \cdots, 1, i+1, i+2, \cdots, n) \tag{**}$$

for some $i \in \{2, 3, \ldots, n\}$, and our stack does not look like this.

Good: we've made progress. We know that the best way to sort our favorite stack (5 2 3 4 1) is at least 1 and at most 4. And since I am getting fast tiring from writing down *"best way to sort our favorite stack (5 2 3 4 1) is this"*, let me define X to be the *best way to sort the stack (5 2 3 4 1)*. So what we've shown so far is:

$$2 \leq X \leq 4.$$

A *lower bound* of 2. A result showing that no matter what we do, we need at least 2 flips to sort the stack (5 2 3 4 1). **Every** concievable method that solves this problem must take **at least** these many flips.

And an *upper bound* of 4. Proved by showing a sequence of flips—an *algorithm*, if you will—to sort (5 2 3 4 1) in 4 flips.

1.3.1 Step IV: Bracketing

OK, let's try to close the gap. Can it be that $X = 2$? Nope. You can again try the "brute-force" approach: try all sequences of two flips, and see none of them work. Or note that all the results of the single flips listed in (*) don't look like the single-flip-sortable structure in (**), and hence cannot themselves be sorted in one additional flip.

Which means more progress:

$$3 \leq X \leq 4.$$

[2] Note that we've augmented our notation: the vertical bar shows where we insert the spatula.

Can we get $X = 3$?

What we've been doing here is something we'll call *bracketing*. We want to understand what the value of X is. And we're getting some easy upper and lower bounds on it, and then slowly improving these bounds.

1.3.2 The Correct Answer

In our case, four flips are necessary. Let's see why.

First, one simple but crucial observation: the only way to get a pancake to the bottom is to have it at the top in a previous step—only one type of flip affects the bottom pancake, and that flip puts the top pancake down there.[3]

Now let's try making the argument. Imagine there was some method to sort (52341) in 3 flips. For any such method, the first flip has to put 5 on the bottom.

> Why? If the first flip didn't move 5 to the botton, the second flip would would have to bring 5 back on the top again, for it to be at the bottom (where it belongs) after the third and final flip. But that would not have sorted much.

Good. So the only way a 3-flip method could work is to move 5 to the bottom on the first flip. Hence, after one flip, we must be at

$$(1\ 4\ 3\ 2\ 5).$$

Since we don't have the time to touch 5 again. We can use similar logic to argue that the second flip must bring 4 to the top: since 4 has to end up in the second-to-last position, at some point it has to be at the top. We only have three flips, though, so we have to bring it to the top now. And the third flip puts 4 in the second-to-last position. But this gives us:

$$(1\ 4\ 3\ 2\ 5) \to (4\ 1\ 3\ 2\ 5) \to (2\ 3\ 1\ 4\ 5).$$

Since the stack is not sorted, there is no way to do this in three flips.

Note. We proved the result using the familiar "*redictio ad absurdum*" (reduce it to something absurd) or "proof by contradiction" method. Assume we can sort it in 3 flips, and show that is not possible. Since we already knew X was either 3 or 4, and it can't be 3, we now know that

$$X = 4.$$

2 The Pancake Numbers

We've proven that the optimal method for sorting (5 2 3 4 1) takes four flips. This prompts our next question: Can every stack of five pancakes be sorted in four flips? Or does there exist a 5-stack that requires at least five flips? In general, how difficult can the cook make things for the waiter using stacks of 5 pancakes?

To formalize this, let us define the "5^{th} pancake number":

$$P_5 = \max_{\text{all starting 5-pancake stacks } S} \text{Minimum number of flips required to sort } S$$

[3]This actually suggests a way to sort any stack of pancakes, but we'll get back to that later.

We can look at the problem of sorting pancakes as a two-player game: The cook chooses the "worst possible" stack of five pancakes (from among all the $5! = 120$ possible starting configurations), and the waiter sorts the stack using the "fewest possible" flips. Then P_5 is the number of flips required in this case.

What do we know about P_5? Well, we know that (5 2 3 4 1) requires 4 flips. What does this say about P_5 (if anything)?

What this tells us is that there exists some stack that requires four flips, so the worst possible stack must require at least 4 flips. I.e.,

$$4 \leq P_5 \leq ?$$

And now back to bracketing again. Can we give a better lower bound on P_5? (For instance, can we construct a nastier stack that takes, say, 5 flips?)

Equally interestingly, what *upper bound* can we give on P_5? Now an upper bound of (say) 7 will require us to show that no matter which of the $5! = 120$ stacks the cook chooses, the waiter can sort that stack in at most 7 flips.

The correct answer is indeed

$$P_5 = 5.$$

We will show the lower bound of 5 a little later; the upper bound of 5 we will not show here, the best upper bound we will show will be $P_5 \leq 7$.

2.1 The n^{th} Pancake Number

It is natural to define the "n^{th} pancake number":

$$P_n = \max_{\text{all starting n-pancake stacks } S} \text{Minimum number of flips required to sort } S$$

Indeed, the problem Harry Dweighter asked at the beginning of these notes can be rephrased as asking:

What is the value of P_n?

Hmm, what indeed. Now we need to figure out a solution for *all* n.

2.2 Step V: Try Small Examples

Let's figure out what P_n is for small n: maybe we'll see a pattern.

- P_0: 0. Any empty stack of pancakes is already sorted.
- P_1: 0. So is any stack of a single pancake.
- P_2: 1. With two pancakes in the stack, either the stack is already in the correct order or it is upside-down. At most, one flip would be required
- P_3: 3. For the lower bound, the stack (1 3 2) requires three flips. (Why?)

  To show the upper bound: any stack of three pancakes can be sorted in three flips. The first step gets the largest pancake to the bottom, which requires as many as two flips. Then, one more flip may be required to sort the top two. Hence 3 flips suffice.

Exercise: Prove that $P_{n+1} \geq P_n$.

2.3 Step VI: Generalizing the Upper Bound Argument

That argument showing that $P_3 \leq 3$ is quite sugegstive, and can be turned into an algorithm that sorts any stack of n pancakes. We'll call this the "Bring-to-top" algorithm.

> If the size of the stack is 1, we're done.
>
> Else, flip pancake n to the top, then flip it to the bottom. Repeat this algorithm on the top $n-1$ pancakes.

What is an upper bound on the number of flips this algorithm takes? Well, it takes at most 2 flips to get the largest pancake to the botton, at most 2 more for the next one, and so on. Hence at most $2n$. Actually, for the last pancake, it takes no flips. So $2n-2$ flips, at most.

We can improve this slightly by changing the algorithm:

> If the size of the stack is 1, we're done.
>
> If the size of the stack is 2, sort the two pancakes using at most 1 flip.
>
> Else, flip pancake n to the top, then flip it to the bottom. Repeat this algorithm on the top $n-1$ pancakes.

Now, if the number of flips required by this modified "bring-to-top" is denoted by $T(n)$, then it satisfies

$$\begin{aligned} T(1) &= 0 \\ T(2) &\leq 1 \\ T(n) &\leq 2 + T(n-1) \qquad\qquad \text{for } n \geq 3. \end{aligned}$$

Which solves to $T(n) \leq 2n-3$ for $n \geq 2$.

Of course, $P_n \leq T(n)$, since $T(n)$ is an upper bound on the number of flips used by modified-bring-to-top to sort any stack of n pancakes), and P_n is the best such upper bound. And so we get

$$P_5 \leq T(5) = 2 \cdot 5 - 3 = 7.$$

Exercise: Suppose I told you that $P_n = N$ for some value N. Use the argument behind these algorithms to show that $P_{n+k} \leq N + 2k$.

2.4 How about some lower bounds?

Lower bounds are a lot trickier. Our previous lower bound arguments were very specific (and essentially used brute force). How do we give arguments that apply to *any* algorithm we can use?

Here's what we'll do—for each n we'll find a really bad stack and show that this stack requires many flips. And to show this, we use the following simple, clever observation—which we call the "breaking-apart" argument:

Suppose that a stack S contains a pair of adjacent pancakes that will not be adjacent in the sorted stack. Then any sequence of flips that sorts stack S must involve at least one flip that inserts the spatula between the two members of this pair (and breaks them apart).

Furthermore, the same principle holds for the "pair" formed by the bottom pancake of S, and the plate.

Make sure you understand why this is true! Let's say that there was some adjacent pair in a stack that aren't consecutive numbers, e.g. (1 4). If we never insert the spatula between the 1 and the 4, they will always remain adjacent. But we know that they aren't adjacent in the sorted stack, which is a contradiction.

So consider any stack where each consecutive pair must be separated, e.g. (5 2 4 1 3), or (3 1 5 2 4) or (2 4 1 5 3). The above observation shows that we'll need a minimum of 5 flips in order to sort each such stack. This proves a lower bound of 5 on the pancake number P_5.

In fact, for any even $n > 2$, consider the lower bound stack

$$(2\ 4\ 6\ 8\ \cdots n\ 1\ 3\ 5\ \cdots (n-1)\ )$$

Each consecutive pair needs to be separated, so this shows that $P_n \geq n$ for $n = 4, 6, 8, \ldots$.

Exercise: Show how to construct such lower bound stacks for all odd $n \geq 5$, where every consecutive pair of pancakes needs to be separated.

(Note that this breaking-apart argument does give you the right lower bound for P_3: the best lower bound you can get this is 2. Whereas we know a lower bound of $P_3 \geq 3$ using other arguments.)

2.5 Some Known Pancake Numbers

The pancake numbers have their own webpage at the On-Line Encyclopedia of Integer Sequences. We know that $P_{17} = 19$, but we don't know (as of now) if $P_{18} = 20$ or 21.

It may seem surprusing at first— *"why don't we just write a program to compute it?"*, you may well ask. But note that there are $18! = 6.40237371 \times 10^{15}$ such stacks of 18 pancakes, and to compute P_{18} we would need to figure out which of these is the hardest to sort. Of course, our calculation just shows that brute-force search is not such a great idea to find this stack, and there may be a slicker way to find this worst-case stack than just looking at every single stack. Indeed, the answers we know use some of these ideas that avoid searching over *all* stacks, but we are still far from knowing if there is a reall fast solution that can compute P_n for large values of n. A great research challenge.

There is another complication: we don't know an algorithm that takes in a stack, and "quickly" outputs the optimal number of flips to sort this stack. We'll come back to these issues (of what a "fast" algorithm is, etc) later in the course, so let us leave it at that for now.

3 One Permutation to Another

Suppose we didn't want to sort the stack of pancakes: instead we wanted to go from a sorted stack (1 2 3 4 5) to some "target" stack (5 2 3 4 1). How should we do this?

Right—each step is reversible, so we can just take the sequence of flips to sort (5 2 3 4 1), and reverse this sequence of flips. Hence, to go from a sorted stack to any target stack, we take at most P_n flips!

Now for something harder: suppose you want to go from a start stack (5 2 4 3 1) to a target stack (4 3 5 1 2). How many flips? Well, you could always go from the start stack to the sorted stack in P_n steps, and then from the sorted stack to the target stack in another P_n steps. That's $2P_n$ steps.

Can we do better?

The answer is yes, there is a better way. The simple but crucial observation is that there's nothing special about the way we numbered the pancakes. We numbered them with the smallest being 1 and the largest being n, just because we wanted to end up with the smallest on the top and the largest on the bottom, and wanted $(1\ 2 \cdots\ n)$ to correspond to this order. If we wanted to end up with a different order, we should just rename them suitably!

Indeed, to translate (5 2 4 3 1) to (4 3 5 1 2), let's just rename the pancakes this way:

$$4 \to 1'$$
$$3 \to 2'$$
$$5 \to 3'$$
$$1 \to 4'$$
$$2 \to 5'$$

In this renamed system, the problem is to translate (3' 5' 1' 2' 4') to (1' 2' 3' 4' 5'), which is just a sorting problem. And we can do this in P_n steps.

This is yet another example where choosing the "right" names, the representation that is "correct" or appropriate for the problem, makes the solution almost obvious. This is an essential skill in problem solving, and something that we will emphasize throughout the course.

4 Applications

4.1 Application I: Pancake Networks

The pancake problem is relevant in the construction of interconnection networks, in which pancake sorting can provide an effective routing algorithm between nodes (processors) in the network, and also help construct a network that is resilient to failures, but in which messages can be routed very quickly.

Here's how we do it. Each node in the network is labeled ("named") by a stack of pancakes. (We imagine that there are $n!$ nodes, and they are labeled using distinct stacks of n pancakes.) Two nodes are connected by a direct link if their names are stacks that differ by exactly one flip. Hence a path from node S to node T in this network corresponds to a sequence of flips transforming stack S into stack T.

The reason this construction is really cool is that the pancake network is optimally reliable for its number of edges and nodes: even if up to $n-1$ nodes get hit by lightning, the network remains connected (that is, every node is accessible from every other node). Note that since each node is only connected to n other nodes, clearly this is the best such result we can hope for—we can disconnect any node from the rest of the network by zapping its n neighboring nodes.

We can also talk about the *diameter* of this network (the diameter of a graph is the maximum distance between any pair of nodes). But this is at most P_n, from the solution to our problem of altering one stack to another. Hence, in routing messages across such a network, the maximum delay between two nodes corresponds to the pancake number P_n. Recall that the size of the network is $N = n!$, whereas the diameter is at most $P_n < 2n$. We'll see later in the course that if $N = n!$, then $n \approx \frac{\log N}{\log logN}$. So not only is the network very robust against failures, we have a routing between nodes that gives us very short paths as well!

4.2 Application II: Sorting by Reversals, and Biological Applications

The pancake sorting problem is also known as *sorting by prefix reversals*: given a permutation, take some prefix of it and reversing it (which corresponds to picking up the top pancakes, and flipping them over. We can instead consider the problem where we choose some section in the middle of the permutation, and reverse that. For example,

$$(1\ 4 \mid 5\ 2\ 6\ 3 \mid 7) \to (1\ 4\ 3\ 6\ 2\ 5\ 7)$$

(With a little stretch of the imagination, you can think of this as being a three-handed waiter problem, where with one spatula he picks up some top substack of pancakes, then he picks some subsequent substack and flips it over.) This is called *sorting by reversals*, and we can ask for the number of flips required to sort a permutation using such reversals.

Why do we care about this? We can use this to define a notion of distance between organisms. Our chromosomes consist of a sequence of genes, which we can think of as a permutation—so two organisms correspond to two different permutations. We are interested in the mutations of the form where some portion of the chromosome breaks off and flips over—which is like sorting by reversals. Now we can define a notion of *evolutionary distance* between organisms as the number of flips required to go from one permutation to another using reversals.[4] This evolutionary distance gives us a sense of how close two different species are to each other (e.g., the head cabbage "Brassica oleracea capitata" and the turnip "Brassica rapa", which look and taste different, but have a lot of genetic material in common).

5 Some History of the Problem

The problem was first posed by Jacob Goodman, writing under the name "Harry Dweighter" (or "Harried Waiter"), back in 1975 in the "American Mathematical Monthly" (vol. 82, p. 1010, 1975). In general terms, it concerns the number of flips, or "prefix reversals", needed to sort the elements of an arbitrary permutation. Initial work on the problem established the limits for P_n that we saw above.

In 1979, William H. "Bill" Gates and Christos H. Papadimitriou improved on the bounds of n and $2n - 3$, showing that flips always suffice and that flips may be needed. They showed the bounds

$$17n/16 \leq P_n \leq 5/3(n + 1).$$

The paper was based on research conducted when Bill Gates was an undergraduate at Harvard University before he went on to found Microsoft, though it was only published a few years later.

[4] We are ignoring many details in this high-level analogy—a more detailed correspondence uses the fact that genes have directions, which correspond to "burnt" pancakes with distinct top and bottom sides, and which are represented by signed permutations.

In 1997, Mohammad H. Heydari and I. Hal Sudborough improved the lower bound, and worked out the pancake numbers up to 13. The upper bounds were improved by Chitturi and others, and the best current bounds are

$$15n/14 \leq P_n \leq 18n/11.$$

Moreover, Asai and others (in 2006) worked out the values of P_n up to $n = 17$ (where $P_{17} = 19$).

As indicated above, there are other variants of the problem: e.g., where the pancakes are burnt on one side, and the goal is not only to sort them but to also place them with the burnt side down. The problem was introduced in the Gates & Papadimitriou paper. Later, our own Manuel Blum along with his student David S. Cohen (now known as David X. Cohen, of *The Simpsons* and *Futurama* fame) gave upper and lower bounds.

An natural question is whether we can get a polynomial-time algorithm that takes a stack, and outputs the optimal number of flips to sort this stack. This problem was shown to be NP-hard, as recently as 2011, by Bulteau, Fertin, and Rusu. More recently, Fischer and Ginzinger gave an algorithm to compute a 2-approximation for this optimal number of flips.

6 Things To Do

Add in some exercises

Put in the pictures on page #1, and for the pancake networks.