CS2040S: Data Structures and Algorithms

Recitation Plan for Week February 1–5

For: Week 4

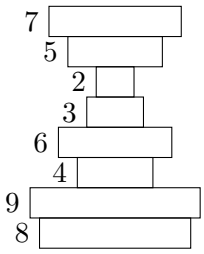
Goals: Applying classical sorting techniques to different models of computation.

Problem 1. (Flippant Pancakes)

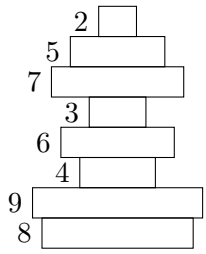
Image credit: https://www.memecenter.com/

It's summer break and you are starting work at ISTACK, a pancake restaurant specializing in serving packing by the stacks (which can go incredibly high!). Their secret is Griddlegorgon, a special griddle which can cook any arbitrary number of pancakes of any size at a single go! Peculiar to patrons of ISTACK, they tip the most when their pancakes are served in a pretty stack by sorted by size. But alas, Griddlegorgon produces a stack of pancakes in arbitrary order! Being the frugal college student seeking to maximize savings, you always strive to earn the most tips from your customers. Thus, given a stack of pancakes with varying sizes produced from Griddlegorgon, your goal is to organize it with the smallest on top and the biggest at the bottom, before serving them to the customer. Since you are holding a plate of pancakes in one hand, you only have one hand free to flip the pancakes. The only way to reorganize the pancakes is to pick up some pancakes **from the top of the stack** and flip them over.

Here is an example of flipping 3 pancakes in a stack of pancakes:



(a) Before flipping top 3 pancakes.



(b) After flipping top 3 pancakes.

Did you know? There is a lot of interesting historical context here! Bill Gates wrote a paper on pancake flipping (with Christos Papadimitriou) when he was an undergrad at Harvard, before dropping out to found Microsoft. They showed you could do it in $(5n+5)/3$ flips. David X. Cohen, a writer for The Simpsons and co-creator of Futurama, wrote a paper "On the problem of sorting burnt pancakes". It also turns out that, in general, finding the shortest sequence of flips for a given stack is NP-hard, which is a class of computationally difficult problems. You'll learn more about that in computational complexity theory when you take CS3230 :)

Problem 1.a. Given a stack of n pancakes, how many **flips** (in terms of n) does it take to order them such that the largest are at the bottom and the smallest on top? You should aim to do this with the least number of flips else your arm will get tired holding the entire stack of pancakes! If there's a constant in your number of flips, can you make it as small as possible?

every pancake need to be flipped twice to be at the bottom
base case: 2 pancakes, 1 flip
2 * (n - 2) + 2 = 2 * n - 3

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. The bottom is sorted. BubbleSort

Recall: Loop Invariant refers to the relationship between variables that is true at the beginning (or end) of each iteration of a loop.

Problem 1.c. One day, Griddlegorgon malfunctioned and always produces pancakes with a randomly burnt side (it's a complicated machine!). The pancakes are now unsightly, but otherwise still very much edible. After Speaking with your manager, it was decided that the stacks are to be presented to the customers such that the burnt side of every pancake faces down. Now how many flips does it take to not only sort the pancakes in the stack, but also to ensure burnt sides are facing downward?

now we need one more flip for the burnt side

base case: 1 pancake, 1 flip
3 * (n - 1) + 1 = 3 * n - 2

Problem 1.d. Bill Gates happens to be a frequent patron of the restaurant who has a habit of ordering pancakes of only two sizes: a large (size 8) and a small (size 3). What is now the best number of flips you need in order to serve a sorted stack of pancakes to him?

n - 1

Problem 1.e. David X. Cohen is another frequent patron of ISTACK and has a peculiar habit of requesting that his stack of pancakes (of various sizes) must be ordered according to their skin textures instead of their sizes. He desires the crunchiest (darkest in colour) on top and the fluffiest (lightest in colour) at the bottom. What is now the best number of flips you need?

Problem 1.f. Show that **any** pancake flipping algorithm requires at least n flips. What do we call such properties of the problem?

Did you know?

It turns out you can use pancake sorting to build a fault-tolerant network graph. For a fixed value of n, each node in the graph is labelled with a permutation of the integers $[1, n]$. That is, there are $n!$ nodes in the graph. Note that if the graph has N nodes, then $n = O(\log N)$ (from Sterling's approximation). We connect nodes u and v with an edge if one pancake flip converts u to v. This is reflected in the figure below.

You have already been shown that there is a path between any two nodes of length at most $2n - 3$ (i.e. the graph with N nodes has diameter $O(\log N)$). It also has degree at most $n = O(\log N)$. So it is a pretty interesting graph to build a network from. In fact, these are known as Cayley graphs and their diameter and degree is sub-logarithmic, since n is smaller than $O(\log N)$.

You can also do the same thing for burnt-pancake graphs, in which now each element in the permutation is signed. This has good fault-tolerance properties. For any $n - 1$ failures, you can still find a path of $2n + 4$ connecting any two nodes within the burnt pancake graph.

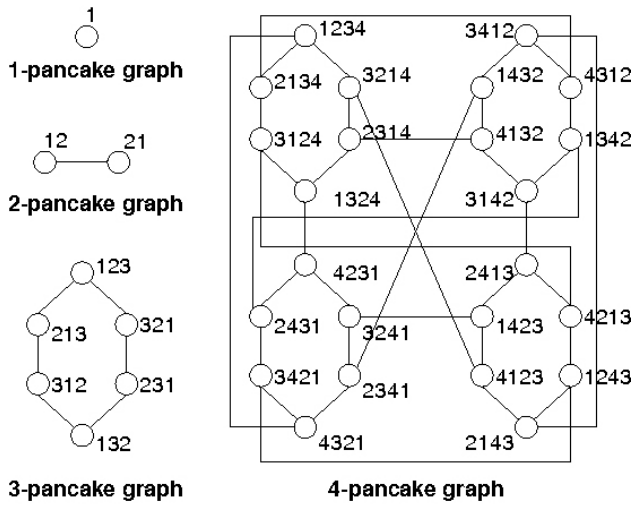


Figure credit: Asai, Shogo et al. "Computing the Diameter of 17-Pancake Graph Using a PC Cluster." Euro-Par (2006).

Problem 2. (DNA Sequence)

Note: We will think about Part (a) of this problem in Week 4, and Part (b) in Week 5.

Imagine, instead of pancakes, you are given a DNA string. You can reverse any segment of the string, and the cost of reversing a segment of length k is k. For example, below is a small segment of the COVID-19 sequence obtained from GenBank:

C A G T G A C A A T

And if you reverse $[2, 5]$ (i.e., items at indices 2, 3, 4, 5 in the sequence), then you get:

C A A G T G C A A T

This reversal costs 4 (length of reversal). For today, **assume you can examine the string for free** before making any reversals.

Problem 2.a. Assume your string is binary: only made up of letters 'A' and 'T'. Devise a divide-and-conquer algorithm for sorting. What is the recurrence? What is the running time?

Note: The only legal operations are reversals. You cannot simply count the 'A's and rebuild the string! You may examine the string as much as you want before deciding which reversals to do. That is, reading the string is free, only reversals have cost.

Problem 2.b. Now, assume your string consists of arbitrary characters. Devise a QuickSort-like algorithm for sorting (*Hint:* use the binary algorithm above to help you implement partition).

What is the recurrence? What is the running time? Assume for today that each element in the string is unique, i.e., there are no duplicates.

Problem 2.c. Extra. What if there are duplicate elements in the string? (*Hint:* think the most extreme case for duplicate elements) Can you still use the exact routine from the previous part? If not, what would you now do differently?