Part 8: Memory Palaces

Chapter 19: Underworld 101: Data Structures & algos

70: Algorithms, Search, Binary Search, Sorting, Merge Sort. Big O, little o, and aysmptotic notation.

You DO NOT NEED to memorize every algorithm, or every detail
Treat these as examples to understand
a. ways to solve problems
b. ways to measure, evaluate, and think about how large complex algorithms and programs work. You need to understand what happens and how to trace and plot the course of programs, when you make a it branch and loop 100000000's of times a second

Did you know there are different algorithms for multiplication?
You probably only learned one in 3rd grade, long multiplication. Stack the numbers, carry the 1

Well theres like 8 different ways to do that

Watch the first 5 minutes
https://www.youtube.com/watch?v=JCbZayFr9RE&list=PLEGCF-WLh2RLHqXx6-GZr_w7LgqKDXxN_&index=3

I'll wait here for your head to explode.

Again, don't feel like you need to remember this particular multiplication algorithm. When I want to multiply two numbers, I just use the * operator. If you are curious, check out some more variations:
https://en.wikipedia.org/wiki/Multiplication_algorithm
https://en.wikipedia.org/wiki/Lattice_multiplication

Then watch this 4 minute video

https://www.youtube.com/watch?v=yRM3sc57q0c&list=PLEGCF-WLh2RLHqXx6-GZr_w7LgqKDXxN_

Sorting algorithms

Watch this:
https://www.youtube.com/watch?v=BeoCbJPuvSE

Again, don't feel like you need to remember any particular sorting algorithm. When I want to sort a list, I just use the .sort() method.

The point here is to understand, and measure how different algorithms work, and how to think about them, and how to measure them.

There is where asymptotic notation comes in

log2 n
n^2
n!

Here are two amazing courses on alrogithms and data structures if you want to go deeper
https://www.youtube.com/watch?v=yRM3sc57q0c&list=PLEGCF-WLh2RLHqXx6-GZr_w7LgqKDXxN_
https://youtube.com/playlist?list=PLDN4rrl48XKpZkf03iYFl-O29szjTrs_O&si=f2RYPPn-LInbSQpV

71: Data structures: Queues, Stacks. Sets. Linked Lists. Hash Tables, Dictionaries. Graphs. BFS, DFS. Trees, Binary Search Trees. Tries.
Yeah just explain these, and give the major tricks for working with them
You don't need to go crazy deep on most of these unless you are doing heavy mathmatical or deep backend stuff

More resources:
Blind 75:
https://www.teamblind.com/post/New-Year-Gift---Curated-List-of-Top-75-LeetCode-Questions-to-Save-Your-Time-OaM1orEU

https://leetcode.com/

https://www.techinterviewhandbook.org/

Highly recommended:
https://www.amazon.com/Cracking-Coding-Interview-Programming-Questions/dp/B0C441PBY3/?content-id=amzn1.sym.cf86ec3a-68a6-43e9-8115-04171136930a

Chapter 20: Cathedrals: Code architecture

Chapter 20: Cathedrals: Code architecture

72: Code architecture, design patterns, different styles, functional programming tiramisu recipie

Similar to algorithms, there is a HUGE design space of structuring code together, system design is one area that focuses on this, which I won't talk too much about here

The other big area is Code architecture.
It really focuses on how do you actually structure pieces of your code and organize files together
This becomes incredibly important as your app grows. Especially are large companies with thousands of coders working together

To give you a taste of how different things can get:
So far we have focused on imperative programming, but there are other styles too. One other big one is functional progrmming.

Here's a really good example of how differently you can organize information, a recipie for tiramisu, in a functional flow diagram, instead of a standard traditional step by step list
https://www.lihaoyi.com/post/WhatsFunctionalProgrammingAllAbout.html

Many languages and frameworks adopt certain patterns and designs. As you dive deeper you'll learn more and more. Theres many ways to build the same app

Another big area here are code design patterns. Basically smaller tricks and code organization.
The gang of 4 is the classic text. It's a little old, but still very relevant. It's a great way to learn how to think about structuring code together. These days you'll find tons of awesome exaples. Again, don't worry about memorizing these, but they are great to take a look at and familiarize yourself with

Overall, take inspiration from what works, and discard the tradition that doesn't