1/6/25: Introduction

- need to install:
    - docker desktop
    - anaconda or miniconda python
    - datagrip or dbeaver
    - VS code for python development (3.10 or higher)
    - ability to interact with git and github
- should review
    - shell cmds
    - docker and docker compose
        - whats a container, whats an image, volume, mapping
- make an aws account

1/8/25: Foundations

- baseline efficiency is linear search
    - start at beginning of list and progress from the first element to the last element until you find what youre looking for or get to the last element
- record: a collection of values
    - ex. a row in a table
- collection: set of records of the same entity type
    - ex. a table
- search key: a value for an attribute from the entity type
    - ex. >1
- need n * x bytes of memory
    - each record takes up x bytes
    - n is the number of records
- contiguously allocated list (array)
    - all n*x bytes are allocated as a single chunk of memory
- linked list
    - each record needs x bytes + additional space for 1 or 2 memory addresses
    - takes up a bit more space
    - need some way to know if you are at the front of the list
- arrays vs linked list pros and cons
    - arrays are faster for random access but slow for inserting anywhere but the end
    - linked lists are faster for inserting anywhere in the list, but slower for random access
- binary search: cut in half each time and figure out which half target is in
    - has to be sorted
    - input: array of values in sorted order, target value
    - output: the location (index) of where target is located or some value indicating target was not found
    - usually recursive

        - can be dangerous if you have a huge dataset
- time complexity
    - linear search
        - best case: O(1)
        - worst case: O(n)
    - binary search for an array (contiguously allocated list)
        - best case: O(1)
        - worst case: O(log2n)
        - super inefficient on a linked list
- database searching
    - problem: can only efficiently search a table buy the primary key (since it is the only sorted column)
    - we need an external data structure to support faster searching than a linear scan
    - solution: database index
- binary search tree
    - has fast insert like a linked list and fast search like an array
    - every node in the left subtree is less than its parent and every node in the right subtree is greater than its parent

1/9/25

-

1/16/25

- need to recursively descend to find all the files??
- OOP in python
    - constructor
    - add_value function appends value to a list
    - consider duplicates using a set object
    - abstract method: if a class implements this method, then implementations of the class also need the method
    - pickle it

```
def _rotate_left(self, x: AVLNode) -> AVLNode:
        T2 = y.left

        y.left = x
        x.right = T2

        ht_x_left = (0 if not x.left else x.left.height)
        ht_x_right = (0 if not x.right else x.right.height)
        x.height = 1 + (ht_x_left if ( ht_x_left >
```

1/27/25: Moving Beyond the Relational Model

- benefits of the relational model
    - mostly standard data model and query language
    - ACID compliance
    - works well with highly structured and large amounts of data
- ACID properties
    - atomicity
    - consistency
    - isolation
    - durability

2/19/25: Intro to the Graph Data Model

- graph database
    - data model based on the graph data structure
    - composed of nodes and edges
- types of graphs
    - connected (vs unconnected)
        - for every pair of nodes, there is a path
    - weighted (vs unweighted)
        - edge has a weight property
    - directed (vs undirected)
        - edges define a start and end node
    - acyclic (vs cyclic)
        - graph contains no cycles

practical

- pip install pymupdf
- ollama pull mistral:latest