Intermediate Software Engineering

Intermediate-Software-Engineering-Summer-2022 (@ Intermediate)
Personal Member ID#: 52751

Hash Tables UMPIRE Cheat Sheet

Definition

- A hash table is a data structure that stores keys and values associated with unique keys. If two keys are indexed at the same location, a collision occurs.
- A hash table is unordered.
- A **Hash Set** is an implementation of a Set interface that does not allow storing of any duplicate values. In Python, there is a built-in set() collection.
- A **Hash Map** is an implementation of a Map interface that allows key/value pairs. Each key must be unique, but values may be duplicated. In Python, we use dictionaries which act in a similar manner as Hash Maps.

U-nderstand

What are some common questions we should ask our interviewer?

- Are there memory constraints?
- What's the required time complexity?
- What kind of data will the inputs be?
- Can I assume all the inputs will be valid?
- What if the input is empty?
- What should we return if there is no solution to the problem?
- What should we return if there are multiple solutions to the problem?

M-atch

Are there any special techniques that we can use to help make this easier?

- Hash tables are typically used to solve string/array or tree problems.
- They can be used for grouping or joining data together by some common attribute
- Common problems that use Hash Tables:
    - Finding duplicates
    - Finding the sum
    - Remove the least recently used (LRU) item
    - Mapping one input to another

    - Construct Binary Tree from Preorder and Inorder Traversal
    - Lowest Common Ancestor of a Binary Tree
- Hash tables cannot be used with O(1) space complexity if you are given other data structures as inputs

P-lan/Pseudocode

- Can you create any **magic** helper methods that would simplify the solution? (ie `items()`, `keys()`, `setdefault()`, `sorted()`)
- Talk through different approaches you can take, and their tradeoffs
- Be able to verbally describe your approach and explain how an example input would produce the desired output

Tips:

- Try to avoid nested loops
    - This is usually a brute force solution and is $O(n^2)$ time complexity

E-valuate

Time Complexity

	Best Case	**Worst Case**
Lookup	O(1)	O(N)
Insert	O(1)	O(N)
Delete	O(1)	O(N)

Note: Best cases assume that there are no collisions, worst cases assume that every entry is a collision