WOMEN WHO
CODE/connect

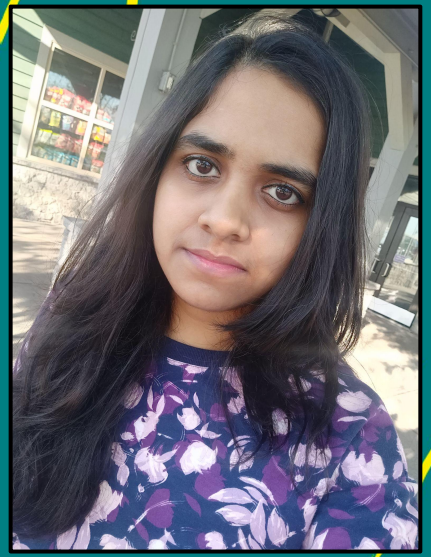
Understanding the Power of Hash Tables

Chethana Gopinath & Karen Wong

LeetCode Study Group by WWCODE PYTHON

November 2021

Meet the Team



Chethana Gopinath

Karen Wong

Soumya Vemuri

WOMEN WHO CODE

WWCode Python

WWCode Python Track

A group of Python enthusiasts who exchange their knowledge of Python and share their passion to help the community grow

What We Do

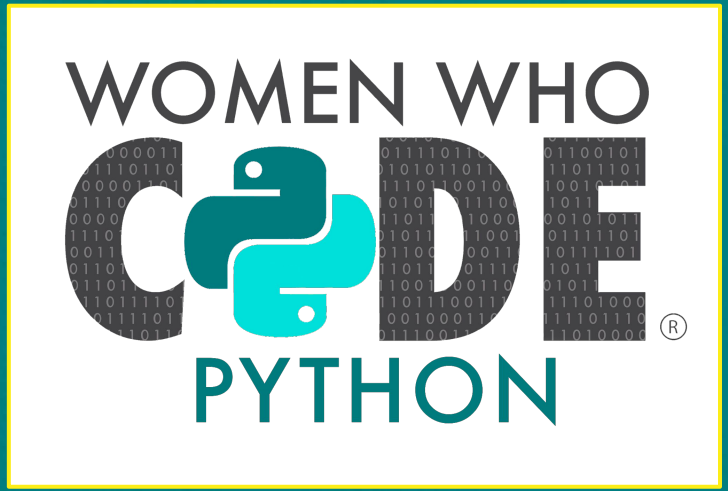
Offer a series of study groups and workshops of varying levels - from beginners to topic-specific advanced events

Our Website

https://www.womenwhocode.com/python/

Resources

https://github.com/WomenWhoCode/WWCodePython



LeetCode Study Group

Goal

- To solve a medium-level algorithm problem
- Recognize patterns of the problem and figure a strategy to solve it

Date

Every other Thursday

Time

8:00 PM Eastern Time

Today's Agenda

- **Hash Tables, Python Dictionaries and Sets**
  A brief overview of each

- **Deep Dive into "Group Anagrams"**
  Problem Discussion
  Test Cases
  Different approaches and analyzing their time complexities
  Live Coding

- **Next Problems to Tackle**

- **QnA**

Let's first tackle this problem....

```python
#we have a person details array where each tuple is a person record of ID and name

people = [(1, "p1"), (4, "p4"),
          (3, "p3"), (9, "p9"),
          (6, "p6"), (2, "p2"),
          (8, "p8"), (5, "p5"),
          ...]

#accessing a single person's name - maybe based on id=5?
res = ""
for person in people:
  id, name = person
  if id == 5:
    res += name
```

Can we do better with how we organize our data?

Looking at our previous problem now..

```
#NOW we have a person details dictionary with ID, name as the key, value pair
people = {
  1: "p1",
  4: "p4",
  3: "p3",
  9: "p9",
  6: "p6",
  2: "p2",
  8: "p8",
  5: "p5",
  ...}

#NOW accessing a single person's name - maybe based on id=5?
id = 5
res = people[id]

print(res) #p5
```

WOMEN WHO CODE®

Quick intro to Hash Tables

Key-Value pairs + **Hash function** + **Array**

name	**age**
p1	50
p2	20

hash(name) -> index (in array to store person info)

hash(p1) -> 1
hash(p2) -> 0

p2, 20	p1, 50

Collisions in Hash tables

name	age
p1	50
p2	20
p4	60
p3	30

hash(p1) => 1

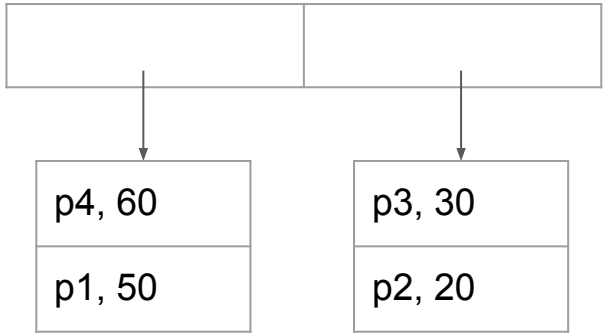
hash(p2) => 0

hash(p4) => 1

hash(p3) => 0

Chaining

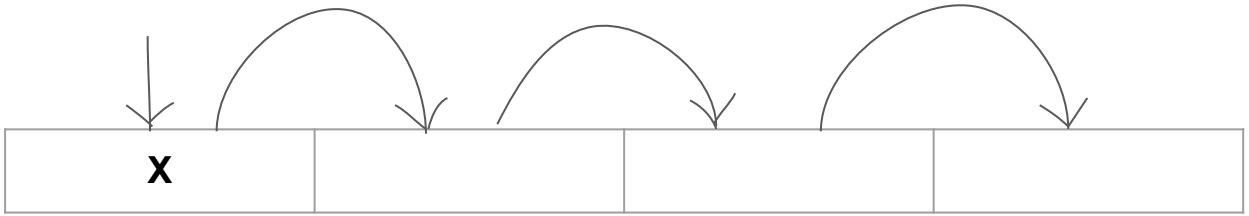
the collision is fixed by chaining them together as a linked list



Open Addressing

Probing (Linear, Pseudo-random probing etc)

We “probe” or look for for an empty space in our data structure when there’s a collision



Linear probing => **`j = (j + 1) % size`**

Python uses a probing algorithm of **`j = (5 * j ) + 1 + perturb`** where **`j`** is the index

Python in-built dict (and set)

Dict

- Implemented internally in Python using a hashtable
- Key-Value pairs where the keys are unique
- Ordered by insertion

Set

- Also implemented using a hashtable
- Unordered collection of data and has no duplicates

`people = {1: “p1”, 2: “p2”}` | `people_names = {“p1”, “p2”}`

Python Dictionaries

- Insert, Delete and Search => O(1)

Usage and thoughts

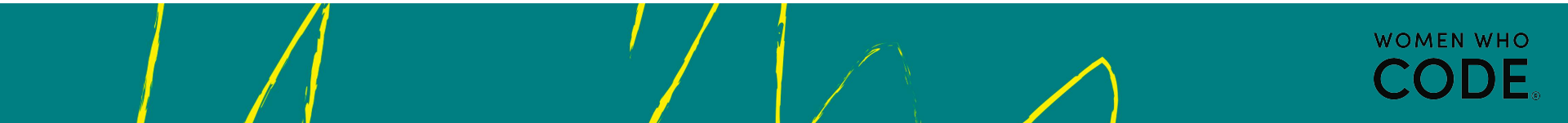
- When you can trade space for time
- If you need to have a key-value pair to easily lookup values based on the keys
- The more you do, the more you'll "see" :)

Group Anagrams

*"Given an array of strings **strs**, group the **anagrams** together. You can return the answer in any order.*

*An **Anagram** is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once."*

Link to Problem



Let's talk in depth

- Anagrams of "ate" => {"tae", "aet", "eat", "tea", ...}
- Also len(OG word) = len(each anagram of that word)
- strs = ["eat","tea","tan","ate","nat","bat"]
- Group the anagrams together and return the answer in any order => res = [[group1], [group2]..]

Let's Code!

https://replit.com/@codernewbie/WWCodePythonLeetcode

Next steps from here

Two Sum - probably the most popular problem on LC!

*"Given an array of integers **nums** and an **integer target**, return indices of the two numbers such that they add up to **target**. You may assume that each input would have **exactly one solution**, and you may not use the same element twice. You can return the answer in any order."*

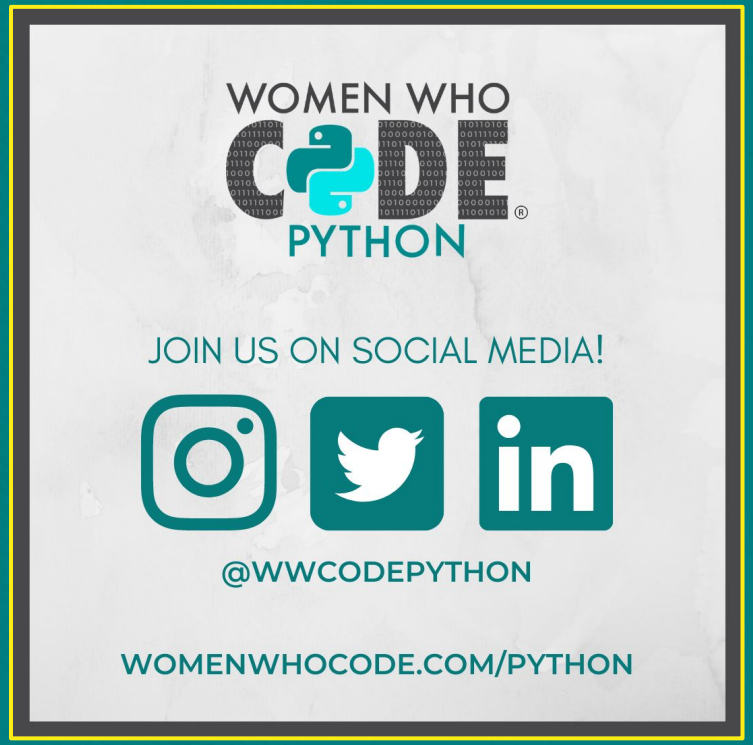
First Unique Character in String

*"Given a string **s**, find the first **non-repeating** character in it and return its index. If it does not exist, return **-1**"*

QnA Time!

WOMEN WHO CODE®

Stay Connected!



***Less Is More: How to Code Python in One Line* - Nov 30**

Advanced Level Session

***LeetCode Study Group* - Dec 2**

Depth First Search
Breadth First Search

Register Here:
https://www.womenwhocode.com/python/events

WOMEN WHO CODE

Thank you!

Keep connected @WWCODEPYTHON !

WOMEN WHO CODE®