Hashmap Structure

{ Key: value }   Table struct : { → count → capacity → Entry* entries }

↳ Array of Entries

Entry struct : { StringObj* key, Value value } → StringObj { → chars → hash }

hash ↓ index

⇒ { { chars hash length }; Value value; }

Methods → API

→ Init : set everything to 0

→ Clear: Clear entries, then init.

→ lookup:

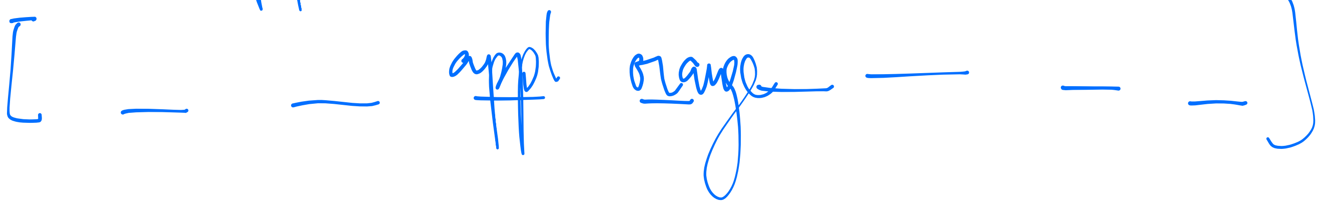
→ insert:

→ Delete:

Open Addressing

Hashing → linear probing cuz too complicated to do double hashing

lookup "apple" $\overset{\text{hash}}{\Longrightarrow}$ 2   orange $\overset{\text{hash}}{\Longrightarrow}$ 2 → apple already there



⇒ keep going +1 (.next) until empty slot

⇒ AMORTIZED O(1) → O(1) complexity on average.

RESIZING THE HM

⇒ base_size

→ capacity above 0.7 → increase_size (when inserting)

→ if cap below 0.1 → decrease

⇒ x2 or /2 everytime.

Easy.

DELETING ITEMS

⇒ mark as DELETED_ITEM ENTRY

$\Rightarrow$ every deleted element will just point to DELETED_ITEM by ref