Hash Tables

https://www.google.com/url?sa=i&url=https%3A%2F%2Fdev.to%2Fracheladaw%2Fintro-to-hash-tables-57n5&psig=AOvVaw1oEe895oa003fNg1ufDCDN&ust=1607131034988000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCICk-9GUs-0CFQAAAAAdAAAAABAD

What are hash tables and why do we use them?

- Hash table: a data structure that can store and lookup elements efficiently (i.e. fast)
- Hash tables are comprised of two key parts:
  - Arrays
  - Hash Function
    - Takes in a key value and returns an index into the table
    - Should (*ideally)* provide a uniform distribution of keys across the hash table (array)
      - When keys *aren't* uniformly distributed, they are *'clustered'*.

Collisions

Collisions occur when the index generated by the hash function for a certain key is already occupied.

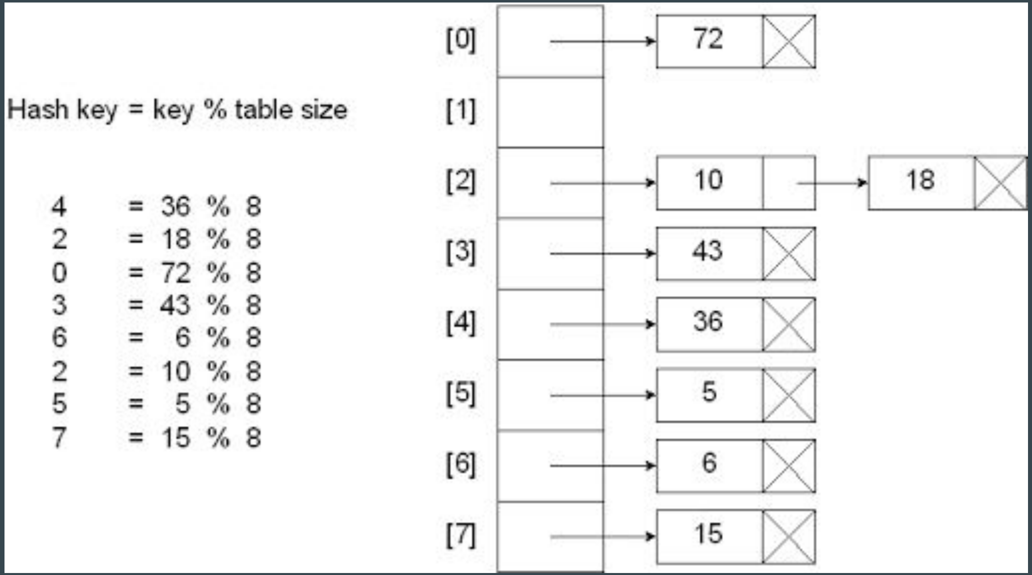
- Example: hash(5) = 10 → table[10] = 5
        - hash(40) = 10 → table[10] is occupied!

Good hash functions will attempt to minimize collisions. *Great* ones will never have ‘em.

- Why not just always use great hash functions?
    - They’re hard to implement ($$$)
    - They’re not always necessary

Collision resolution techniques

- Chaining - Each bucket ~= a linked list.
  - Collision? → Just link the new element. Simple as that!
- Probing
  - **Linear -** table[hash(x)] is occupied → check table[hash(x) + 1]
    - Occupied? → check table[hash(x)+2]
      - And so forth, until an empty bucket is found.
    - Drawbacks?
      - Clustering! If our function results in lots of collisions, values will be clustered around one another.
      - Takes a while to find the next empty bucket.
  - **Quadratic -** table[hash(x)] is occupied → check table[hash(x) + 1^2]
    - Occupied? → check table[hash(x)+2^2]
      - And so forth, until an empty bucket is found
    - Helps avoid clustering, but needs a big ol' table.



http://faculty.cs.niu.edu/~freedman/340/340notes/340hash.htm

Real-world usage

University assigns an Identikey to each student.

Say they want to find the student ‘Honda Rhoenigman’.

- Option 1: Start at 0000001. Is that her? No?
    - Check 0000002. Is that her? No?
- Option 2: Hash(Honda Rhoenigman) = 1294831.

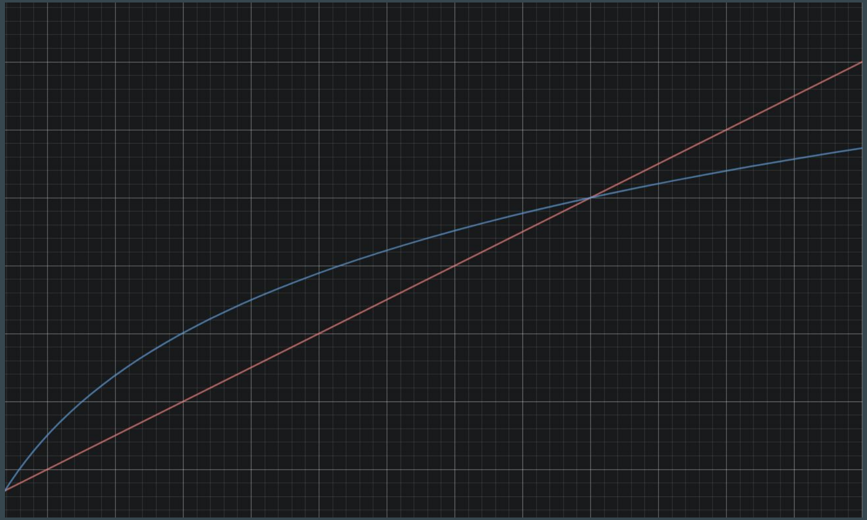
Inappropriate use cases

When the data we wish to store is sufficiently small, hashing may actually be more costly than it is worth.

Our knee-jerk is that ‘linear’ time is horrible.

- Value(linear search) < Value(hash(key))

But! For small datasets, it may be more sensible/cost-effective to just linear-search an array than to implement hashing.



Food for thought!

Final Remarks

- Interview Grading Zoom links are under locations/description in your appointment slot!
  - Please show up with your code ready to run!
  - Email/Slack us if you’re having trouble signing up or cannot make it to your appointment
    - Please make sure it’s not a last minute notice unless an emergency arises

THANK YOU!