Today

- Quiz 6
- Intro to Hash Tables

Assignments

- HW 5 due
- HW 6 out
- Midterm exam next Thurs
Hash Tables Overview

A hash table is a type of data structure for fast data access (e.g., $O(1)$)

- For KV Map this means improved performance for:
  - ... contains
  - ... update (via operator[])
  - ... insert
  - ... erase

The basic idea:

- Keep an array of elements (called “buckets”)
- Define a “hash” function $h : \text{Key} \rightarrow \text{Index}$
- Make $h$ fast relative to $n$ (i.e., $h$ should be constant time relative to $n$)

```
0
1
2
n - 1

The hash function $h$ maps keys to array indices
```

"Table"
Challenges when it comes to defining hash tables

- How do we define a good hash function?
- How do we deal with the fixed size nature of an array?

“Perfect” hash functions

- Map each key to a unique array index ... i.e., no two keys map to same index
- Hard if you do not know all the key values to expect
- You may also have more keys than table (array) indexes

Most hash functions in practice aren’t “perfect”

- Can map different keys to the same index ... creating a “collision”
- Two general ways to deal with collisions (more later)

If our hash functions evenly distribute data, can deal with collisions efficiently
Hash function examples

**Approach 1: “Selecting Digits”**

- Select specific parts of (integer) key to use as the hash value
- Example: assuming keys are 9-digit employee numbers
  - Let $h(k) = 4\text{th and } 9\text{th digit of } k$
  - E.g., $h(001364825) = 35$
  - Given key 001364825, insert entry at table[35]
- Fast hash function, but may not evenly distribute data ... why?
- How can we use with strings? ... replace characters with ASCII codes

**Approach 2: “Folding”**

- Add (sum) digits of the key
- Example: again assuming 9-digit employee numbers
  - $h(k) = i_1 + i_2 + \cdots + i_9$ where $k = i = i_1 i_2 \cdots i_9$
  - Since $h(001364825) = 29$, insert entry at table[35]
- Also fast, but also may not evenly distribute data
- In this example, only hits ranges from 0 to 81
- Can pick different (similar) schemes ... e.g., $i_1 i_2 i_3 + i_4 i_5 i_6 + i_7 i_8 i_9$