Hashing

Topics for this week:
- dictionary data structure
  - Python built-in
  - design our own
- hashing
  - hash function, hash code
  - collision handling

Project + midterm
Problem: Word Counting

We have a text document and we want to count the number of occurrences for every word.

Why?

Example:
See spot
See spot run
See spot run quickly down the street to see another dog

“see” occurs 4 times
Problem: Word Counting

See spot
See spot run
See spot run quickly down the street to see another dog

Naïve approaches:

1. Store the words in a list
   
   ```
   ["see", "spot", "see", "spot", "run", "...
   ```

2. Store the words with the respective counts in a list
   
   ```
   [ ("see", 4), ("spot", 3), ("run", 2), ...
   ```

3. Store the words like in 2 but keep them alphabetically sorted
   
   ```
   [ ("another", 1), ("dog", 1), ("down", 1), ("quickly", 1), ...
   ```

- Compute the # occurrences for a given word is $O(n)$ too much!

- $O(\log n)$ lookup time but $O(n)$ insert time
Problem: Word Counting

Built-in Python dictionary:
- stores **key/value** pairs

  key e.g. "see"
  value e.g. 4
Problem: Word Counting

Dictionary Data Structure - how does it work internally?

- hash function (and hash code):

  ```python
def hash_code(key):
    return the number from 0...25 that corresponds to the first letter of key
  
e.g. hash_code("see"): 18
  hash_code("another"): 0
```

- hash table/array - what do we store there? (and where?)

  - in our example of size 7

```plaintext
Red size list

\{ ("another", 1) \}  
\{ ("see", 1) \}  
\{ ("spot", 1) \}  
\{ ("see", 1) \}  
\{ ("another", 1) \}  
\{ ("see", 1) \}  
\{ ("spot", 1) \}  
```

- collision: its original location was occupied, we went down the table
Problem: Word Counting

Dictionary Data Structure - how does it work internally?

- supports operations/functions:
  
  - `get(hashTable, key)`
    
    ```java
    like update value on the next slide except the last line is
    return hashTable[i].value
    ```

  - `put(hashTable, key, value)`: 
    
    ```java
    hc = hash code (key)
    i = 0
    counter = 0
    while hashTable[i] is occupied and counter < size:
      i += 1
      if i == size:
        i = 0
        counter += 1
      if counter >= size:
        return None
      else:
        hashTable[i] = (key, value)
    return True
    ```

  - `insert`
Problem: Word Counting

Dictionary Data Structure - how does it work internally?

- supports operations/functions:

- `updateValue(hashTable, key, value):`

  ```python
  hc = hash(key)
  i = hc % size
  counter = 0
  while hashTable[i].key != key and counter < size:
    i = (i + 1) % size
    counter += 1
  if counter ≥ size:
    return False
  hashTable[i].value = value
  return True
  ```

- Running time: \(O(1)\) if good hash function (almost no collisions)
Problem: Word Counting

Good hash functions?

How do we deal with collisions?

open addressing

chaining

Good table size?