lecture 32
hashing

Map

keys

values

A map is a set of (key, value) pairs.

- For each key, there is one value.
- Two keys could have the same value.

Special Case:

keys $K = \{0, 1, \ldots, m-1\}$

$n = 2$

keys in this map

Hash Function

$h : K \rightarrow \{0, 1, \ldots, m\}$

Hash function: two steps

hash coding

Compression

$h : K \rightarrow \{\text{integer}\} \rightarrow \{0, 1, \ldots, m\}$

Hash table
Hash function: two steps

\[ h : K \to \{ 0, \ldots, m-1 \} \]

\[ h = \text{compression} \circ \text{hashCoding} \]

Composition of functions

Example: hash coding of string

\[ h : K \to \{ \text{integers} \} \to \{ 0, 1, \ldots, m-1 \} \]

\[ s.\text{hashCode}() = \sum_{i=0}^{s.\text{length}-1} s[i] \times \text{Length} - i - 1 \]

Eg. \( x = 2^{16} \times 31 \times 1 \)

Recall last lecture

\[ \text{hashCode} = \sum_{i=0}^{s.\text{length}-1} s[i] \times \text{Length} - i - 1 \]

\[ x = 2 \]

\[ x = 31 < 2^5 \]

\[ \text{hashCode} = \sum_{i=0}^{s.\text{length}-1} s[i] \times \text{Length} - i - 1 \]

\( x = 1 \)

Eg. "eat" and "ate" would have the same hash code.

Java String.hashCode()

\[ s.\text{hashCode}() = \sum_{i=0}^{s.\text{length}-1} s[i] \times \text{Length} - i - 1 \]

\[ x = 31 < 2^5 \]

\[ (\text{int}) 'o' \]

\[ (\text{int}) 'e' \times 31 \]

\[ (\text{int}) 'l' \times 31^2 \]

\[ (\text{int}) 'i' \times 31^3 \]

\[ (\text{int}) 'h' \times 31^4 \]

Hash codes

\[ h : K \to \{ \text{integers} \} \to \{ 0, 1, \ldots, m-1 \} \]

Hash Coding

\[ x \mod m \]

(often take \( m \) prime)
Hash code:
41
16
25
21
36
35
53

hash values (hashCode % 7)
6
2
4
0
1
0
4

Collisions
Hash function maps two keys to the same index in the hash table.
- Two keys have same hash code
- Two keys have different hash codes but are compressed to same hash value

K integers
hashCoding
Compression
Hash Table Implementation

Using an array of linked lists is called "separate chaining."
Each linked list is called a "bucket."

A "perfect hash function" puts at most one entry (k,v) in each bucket.

A good hash function puts at most a few entries in each bucket. \( \Rightarrow O(1) \) access
(You are free to change the hash code and the compression function so it has whatever properties you want)

Load factor "entries"
\[
\frac{\text{number of } (k,v) \text{ pairs in map}}{\text{number of buckets (m)}}
\]
Rule of thumb:
keep load factor < \( \frac{3}{4} \)

Java API

class HashMap<K, V> {
    V put(K k, V v)
    V get(K k)
}

In fact, the parameter is "Object".
Overriding equals(), hashCode()

class Object
    boolean equals()
    int hashCode()
    Object clone()
    String toString()

class MyClass
    boolean equals()
    int hashCode()

\[
x. equals (y) \quad \Downarrow \quad x. hashCode() = y. hashCode()
\]

Maps

array (or linked list)

Hash Map

Sets

Can hash tables be used for a set of keys? (Yes)

What operations can we perform?

- add/remove (c)?
- add/remove (key)?
- find (i), find (key)?

Java API

class Hash Set < K >

void add(K)
boolean remove(K)
boolean contains(K)
boolean isEmpty(K)?

Note: uses K.hashCode()