CSE12 - Lecture 17

Friday, May 12, 2023 8:00 AM

Map and HashTable

Hash Function (same as previous)

```
int getIndex(String k) {
    return k.length;
}
```

# of buckets – 4
(i.e. the size of the array)

expandCapacity() called in set()

LoadFactor – 0.67

```
set("Smith", 1);
set("Johnson", 2);
set("Williams", 3);
set("Brown", 4);
set("Jones", 5);
set("Garcia", 6);
set("Miller", 7);
set("Davis", 8);
set("Rodriguez", 9);
set("Martinez", 10);
```

hash	index	lf
5	1 5	0/4
7	3 7	1/4
8	~~0~~ ~~0~~ 8	1/4
5	5	3/4
5	5	4/8
6	6	5/8
6	6	6/8
5	5	7/16
9	9	8/16
8	8	9/16

Draw the picture of the HashTable using Linear Probing
(using expandCapacity)

What is the run-time for this HashTable (do picture first):

set()

Worst Case $\Theta(1) * \Theta(N) * \Theta(N) \rightarrow \Theta(N^2)$

Best Case: $\Theta(1)$

What conditions make up the best case for set()?

minimal or no collisions; no expand capacity

get()

Worst Case $\Theta(N)$

Best Case: $\Theta(1)$

What conditions make up the best case for get()?

No collisions, minimal collisions

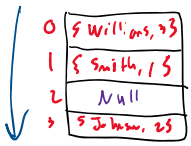
What happens if we remove something, then try to find something that collided it?

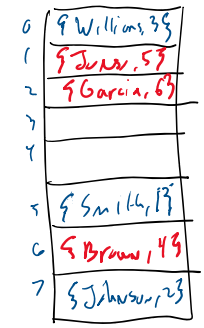
```
remove("Brown");
get("Davis");
```

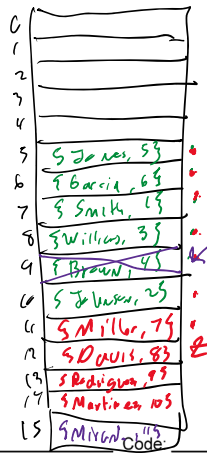
What happens if we add something else?

```
set("Miranda", 11);
```

KeyValuePair<String, Integer>[] contents = ...







NULL? tombstone

& & compress

Not used in size
is used in loadFactor

Key → null
inherit from KeyValuePair? .equals() return false

expand capacity
& remove during re-hash

Name: _____ PID: _____ Code: 9325

Amortized Analysis

get()

What is the run-time for ArrayList add()?

1 Worst Case $\Theta(N)$

1 Best Case: $\Theta(1)$

1 Average Case: $\Theta(1)$

find() wc N
bc 1
ac $\frac{N}{2}$

What is the run-time for HashTable set() using Separate Chaining and a good hash function?

LF. .75 Java

get()

1 Worst Case $\Theta(1) + \underline{\Theta(N)} \rightarrow \Theta(N)$

1 Best Case: $\Theta(1)$

1 Average Case: $\Theta(1)$ per set()

What is the run-time for HashTable set() using Linear Probing and a good hash function?

LF. 67 Python

Worst Case $\Theta(N)$

Best Case: $\Theta(1)$

Average Case: $\Theta(1)$ per add

Why is the hash function important?

even distribution
↳ less collisions

String
↳ hashcode()