CSE12 - Lecture 16

Wednesday, May 8, 2024 10:00 AM

PA6 released - due next Wednesday @ 8am
PA5 - hard deadline - Thursday @ 8am
PA3 Late/Resubmit - hard deadline - Thursday @ 8am
Exam 1 grading - 23 points -> 18 points
Exam 2 - May 15th

- https://ucsd-cse12-sp24.github.io/lectures/exam2.html

Lecture 16

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 – 0.75

```
      key    value
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);
```

Draw the picture of the HashTable using Separate Chaining (using expandCapacity)

Does the run-time change with expandCapacity()?
→ only for the worst case

What is the run-time for this HashTable (do picture first): $2N * 2N$

set()
Worst Case $\Theta(N^2)$
Best Case: $\Theta(1)$

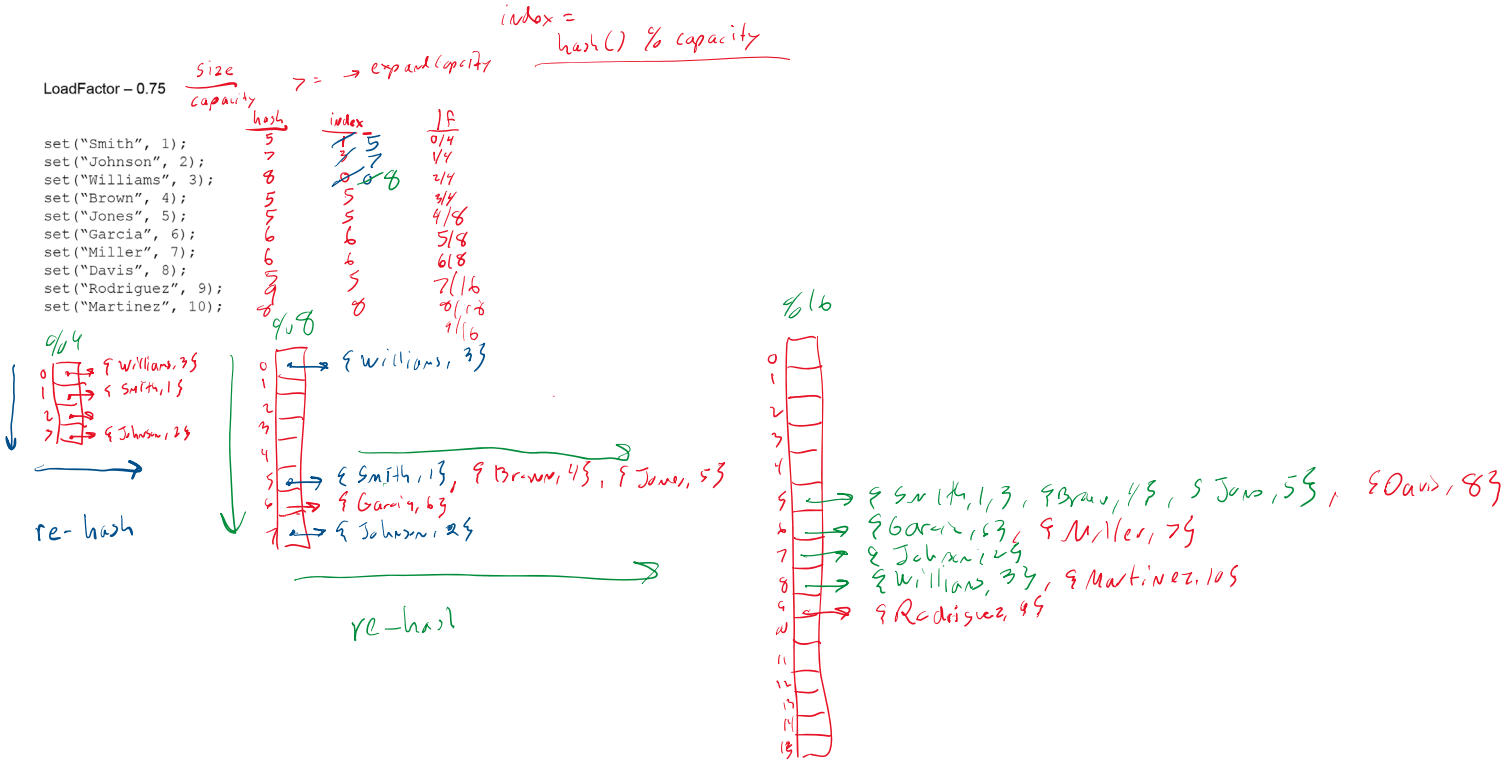
What conditions make up the best case for set()?
No collisions, No expandCapacity → empty or nearly empty buckets

get()
Worst Case $\Theta(N)$
Best Case: $\Theta(1)$

What conditions make up the best case for get()?
even distributions → empty or nearly empty buckets

Why is the hash function important?
bad hash → collisions
good hash → even distribution

Name: _____ PID: _____ Code: 8991


index =
hash() % capacity
LoadFactor – 0.75
size / capacity >= → expand capacity
hash
index
LF
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);
%4
%8
%16
re-hash
re-hash