CSE 12 – Basic Data Structures and Object-Oriented Design
Lecture 17

Greg Miranda and Paul Cao, Winter 2021

This lecture is being recorded

Topics

- Questions on Lecture 17?
- Linear Probing

Hash Table – draw the picture (Separate Chaining)

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

# of buckets – 6

(i.e. the size of the array)

```
set("Smith", 1);
set("Maria", 2);
set("Christine", 3);
set("Brown", 4);
set("Julia", 5);
set("Garcia", 6);
set("Miller", 7);
set("Davis", 8);
set("Wesley", 9);
set("Martinez", 10);
```

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

```
set(“Smith”, 1);
set(“Maria”, 2);
set(“Christine”, 3);
set(“Brown”, 4);
set(“Julia”, 5);
set(“Garcia”, 6);
set(“Miller”, 7);
set(“Davis”, 8);
set(“Wesley”, 9);
set(“Martinez”, 10);
```

Hash Table – draw the picture (Separate Chaining)

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

# of buckets – 4

(i.e. the size of the array)

expandCapacity() called in set()

LoadFactor – 0.75

```
set(“Smith”, 1);
set(“Maria”, 2);
set(“Christine”, 3);
set(“Brown”, 4);
set(“Julia”, 5);
set(“Garcia”, 6);
set(“Miller”, 7);
set(“Davis”, 8);
set(“Wesley”, 9);
set(“Martinez”, 10);
```

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

# of buckets – 4

LoadFactor – 0.75

```
set("Smith", 1);
set("Maria", 2);
set("Christine", 3);
set("Brown", 4);
set("Julia", 5);
set("Garcia", 6);
set("Miller", 7);
set("Davis", 8);
set("Wesley", 9);
set("Martinez", 10);
```

Hash Table – draw the pictures (Linear Probing)

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

# of buckets – 4

(i.e. the size of the array)

expandCapacity() called in set()

LoadFactor – .67

```
set("Smith", 1);
set("Maria", 2);
set("Christine", 3);
set("Brown", 4);
set("Julia", 5);
set("Garcia", 6);
set("Miller", 7);
set("Davis", 8);
set("Wesley", 9);
set("Martinez", 10);
```

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

# of buckets – 4

expandCapacity() called in set()

LoadFactor – .67

```
set(“Smith”, 1);
set(“Maria”, 2);
set(“Christine”, 3);
set(“Brown”, 4);
set(“Julia”, 5);
set(“Garcia”, 6);
set(“Miller”, 7);
set(“Davis”, 8);
set(“Wesley”, 9);
set(“Martinez”, 10);
```

Amortized analysis

- Reasoning: worst case scenario analysis assumes that worst case input happens all the time but it may not be true.

- Approach: Assume worst case but look at the whole picture

Example: insert into the end of arrays

What is the worst case runtime of this operation

A: O(1)

B: O(n)

C: O(logn)

D: I forgot what O is....

Example: insert into the end of arrays

Amortized analysis

Hashing - insert

If we insert an element in the hash table, what is the worst case runtime? We consider expanding capacity and rehashing after load factor is reached.

A. O(1)
B. O(n)
C. O(logn)

Hashing - insert

If we insert an element in the hash table, what is the worst case runtime? We consider expanding capacity and rehashing after load factor is reached.

A. O(1)
B. O(n)
C. O(logn)

Would separate chaining or linear probing matter?

Hashing - insert

If we insert an element in the hash table, what is the worst case runtime? We consider expanding capacity and rehashing after load factor is reached.

A. O(1)
B. O(n)
C. O(logn)

Would separate chaining or linear probing matter?

A. Yes
B. No

Amortized analysis of Hashing - insert

We assume that we double the size of table when we rehash. What is the amortized analysis result?