CSE 250 Recitation

April 23 - 24: Expected Runtime, Hash Tables

Expected Runtime Example #1

```
def mystery(data):
  if randInt() % 100 == 0:
    sum = 0
    for d in data:
      sum += d
  else:
    sum = data[0] * data.size()
  return sum
```

Exercise:

Write out the growth function, T(n), representing the runtime of this function.

What are the unqualified bounds?

Expected Runtime Example #1

```
def mystery(data):
  if randInt() % 100 == 0:
    sum = 0
    for d in data:
      sum += d
  else:
    sum = data[0] * data.size()
  return sum
```

Exercise:

Write out the growth function, T(n), representing the runtime of this function.

$$T(n) = \begin{cases} n & \text{if X \% 100 == 0} \\ 1 & \text{otherwise} \end{cases}$$

What are the unqualified bounds?

O(n), Ω(1)

Expected Runtime Example #1

Discussion: What is E[T(n)]?

$$T(n) = \begin{cases} n & \text{if X \% 100 == 0} \\ 1 & \text{otherwise} \end{cases}$$

Remember: $E[X] = \sum_i P_i \cdot X_i$

Expected Runtime Example #1

Discussion: What is E[T(n)]?

$$T(n) = \begin{cases} n & \text{if X \% 100 == 0} \\ 1 & \text{otherwise} \end{cases}$$

Remember: $E[X] = \sum_i P_i \cdot X_i$

$$E[T(n)] = \frac{1}{100} \cdot n + \frac{99}{100} \cdot 1$$

Expected Runtime Example #1

Discussion: What is E[T(n)]?

$$T(n) = \begin{cases} n & \text{if X \% 100 == 0} \\ 1 & \text{otherwise} \end{cases}$$

Remember: $E[X] = \sum_i P_i \cdot X_i$

The first outcome happens 1/100 times

$$E[T(n)] = \frac{1}{100} \cdot n + \frac{99}{100} \cdot 1$$

Expected Runtime Example #1

Discussion: What is E[T(n)]?

$$T(n) = \begin{cases} n & \text{if X \% 100 == 0} \\ 1 & \text{otherwise} \end{cases}$$

Remember: $E[X] = \sum_i P_i \cdot X_i$

The second outcome happens 99/100 times

$$E[T(n)] = \frac{1}{100} \cdot n + \frac{99}{100} \cdot 1$$

Expected Runtime Example #1

Discussion: What are the bounds of E[T(n)]?

$$E[T(n)] = \frac{1}{100} \cdot n + \frac{99}{100} \cdot 1$$

Expected Runtime Example #1

Discussion: What are the bounds of E[T(n)]? **O(n)**

$$E[T(n)] = \frac{1}{100} \cdot n + \frac{99}{100} \cdot 1$$

Expected Runtime Example #2

```
def mystery(data):
  if randInt()%data.size()==0:
    sum = 0
    for d in data:
      sum += d
  else:
    sum = data[0] * data.size()
  return sum
```

Exercise:

Write out the runtime, T(n), and the expected runtime, E[T(n)] for this function.

What are the bounds on these growth functions?

Expected Runtime Example #2

```
def mystery(data):
  if randInt()%data.size()==0:
    sum = 0
    for d in data:
      sum += d
  else:
    sum = data[0] * data.size()
  return sum
```

$$T(n) = \begin{cases} n & \text{if X \% n == 0} \\ 1 & \text{otherwise} \end{cases} \in O(n), \Omega(1)$$

$$E[T(n)] = \frac{1}{n} \cdot n + \frac{n-1}{n} \cdot 1 \in O(1)$$

Discussion: Sets vs Maps

Remember: A hash table is a data structure…it can be used to implement multiple ADTs, like Sets and Maps

How would you implement Sets using a hash table? What about Maps?

- What are the differences?
- What are the runtimes of the main operations?

Come up with some examples of Sets vs Maps.

Misc Points

1. Determining the bucket is a TWO step process
   a. Call the hash function
   b. Take the result mod N
2. If two objects map to the same bucket, it doesn't mean they are equal!
   a. **`.hashCode()`** determines the hash code of an Object in Java
   b. **`Math.floorMod(hc, N)`** determines the bucket from a hash code (**`hc`**) and # buckets (**`N`**)
   c. **`.equals()`** tells you if two objects are equal

Discussion: Any questions about modular arithmetic?

Note: In PA3 you won't be calling **`.hashCode()`** directly, you'll use the hash functions passed to **`CuckooMap`**

Hashing w/Chaining

hash(A) = 636

hash(B) = 712

hash(C) = 459

hash(D) = 12

hash(E) = 157

Exercise

1. Start with a 5-bucket hash table (with chaining) and insert A-E.
   a. What is the load factor?
2. Rehash to an array of size 10.
   a. What is the load factor?

Hashing w/Open Addressing

hash(A) = 636

hash(B) = 712

hash(C) = 459

hash(D) = 12

hash(E) = 157

Exercise

1. Start with a 5-bucket hash table (with open-addressing) and insert A-E
2. Confirm lookup works for all 5 keys
3. Rehash to an array of size 10
4. What if we try to lookup F which hashes to 22?
5. Remove B...confirm lookup still works

Hashing w/Cuckoo Hashing

$h_1(A) = 312$ $h_2(A) = 636$

$h_1(B) = 714$ $h_2(B) = 243$

$h_1(C) = 457$ $h_2(C) = 684$

$h_1(D) = 121$ $h_2(D) = 871$

$h_1(E) = 154$ $h_2(E) = 939$

Exercise

1. Start with a 5-bucket hash table (with Cuckoo Hashing) and insert A-E
2. Rehash as needed…

Cuckoo Hashing Exercise

Imagine we are inserting **A**, **B**, and **C** into a hash table using Cuckoo Hashing…

1. Come up with unique hash values for **A**, **B**, and **C** that would require the hash table to rehash if there are 10 buckets
2. Do the same that would require the hash table to rehash for 20 buckets
3. Can you pick a set of unique hash values that would require the hash table to resize for both 10 **and** 20 buckets, but not 40?