CS 225

#24: BTree Analysis + Hashing Intro

March 14, 2018 · *Wade Fagen-Ulmschneider*

BTree Properties

For a BTree of order **m**:

1. All keys within a node are ordered.
2. All leaves contain no more than **m-1** nodes.
3. All internal nodes have exactly **one more child than keys**.
4. Root nodes can be a leaf or have **[2, m]** children.
5. All non-root, internal nodes have **[ceil(m/2), m]** children.
6. All leaves are on the same level.

BTree Analysis

The height of the BTree determines maximum number of lookups possible in search data.

...and the height of our structure: h ~ logm(n)

m ~ 100 sometimes >= 1000

Therefore, the number of seeks is no more than: logm(n).

...suppose we want to prove this!

BTree Proof #1

In our AVL Analysis, we saw finding an **upper bound** on the height (**h** given **n,** aka **h = f(n)**) is the same as finding a **lower bound** on the keys (**n** given **h,** aka $\mathbf{f^{-1}(n) == g(h)}$).

Goal: We want to find a relationship for BTrees between the number of keys (**n**) and the height (**h**).

BTree Strategy:

1. Define a function that counts the minimum number of nodes in a BTree of a given order.
   a. Account for the minimum number of keys per node.
2. Proving a minimum number of nodes provides us with an upper-bound for the maximum possible height.

t = ceil(m/2)

Proof:

1a. The minimum number of <u>nodes</u> for a BTree of order **m** at each level is as follows:

root: 1

level 1: 2

level 2: 2t

level 3: 2t^2

...

level h: 2 t^(h-1)

1b. The minimum total number of <u>nodes</u> is the sum of all levels:

minimum # of nodes: 1 + 2 * (t^h - 1) / (t -1)

---

2. The minimum number of keys:

minimum # of keys = 1 + 2 * (t^h - 1) / (t -1) * (t -1)
= 2 * t^h -1

---

3. Finally, we show an upper-bound on height:

(n+1)/2 >= t^h

log_{t} ((n+1) / 2) >= h

log_{ceil(m/2)} ((n+1)/2) ~ log_{m} (n)

So, how good are BTrees?
Given a BTree of order 101, how much can we store in a tree of height=4?

Minimum:

$2 t^h - 1 = 2 * (50)^4 - 1 = 12.5$ M

Maximum:

Hashing

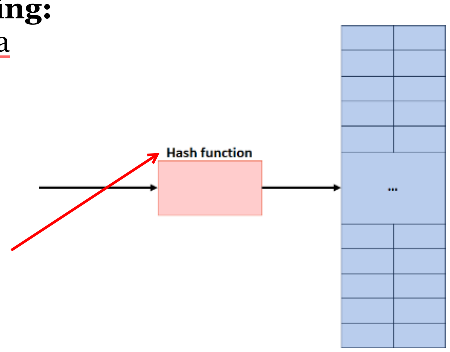
Locker Number	Name
103	
92	
330	
46	
124	

...how might we create this today?

Goals for Understanding Hashing:

1. We will define a **keyspace**, a (mathematical) description of the keys for a set of data.

2. We will define a function used to map the **keyspace** into a small set of integers.

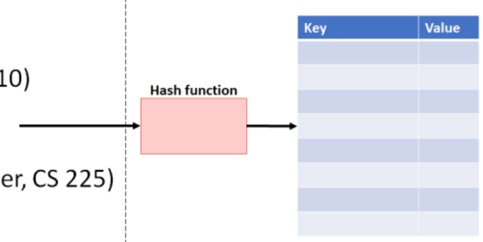


All hash tables consists of three things:

1. a hash function f(k)
2. an array
3. mystery element

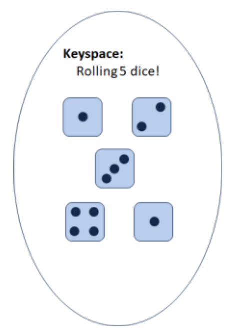
A Perfect Hash Function

(Angrave, CS 241)
(Beckman, CS 421)
(Cunningham, CS 210)
(Davis, CS 101)
(Evans, CS 126)
(Fagen-Ulmschneider, CS 225)
(Gunter, CS 422)
(Herman, CS 233)



...characteristics of this function?

A Second Hash Function



0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

...characteristics of this function?

CS 225 – Things To Be Doing:

1. Programming Exam B is ongoing
2. MP5 has been released; EC deadline is Monday back from break
3. lab_btree released this week; due Tuesday, March 27th at 11:59pm
   (That's the Tuesday evening after spring break)
4. Daily POTDs are ongoing!