Data Structures

BTree Analysis

CS 225
Brad Solomon & G Carl Evans

October 6, 2023

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN
Department of Computer Science

Learning Objectives

Review BTree Properties

Analyze the performance of the BTree

BTree Properties

A **BTrees** of order **m** is an m-way tree:

- All keys within a node are ordered
- All leaves contain no more than **m-1** keys.
- All internal nodes have exactly **one more child than keys**

- Root nodes can be a leaf or have **[2, m]** children.
- All non-root, internal nodes have **[ceil(m/2), m]** children.

- All leaves are on the same level

BTree Analysis

We saw for AVL that finding an upper bound on the height (given **n**) is the same as finding a lower bound on the nodes (given **h**).

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

BTree Analysis

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

…and the height of the structure is: _______________.

Therefore: The number of seeks is no more than ___________.

…suppose we want to prove this!

BTree Analysis

Strategy:

We will first count the number of nodes, level by level.

Then, we will add the minimum number of keys per node (**n**).

The minimum number of nodes will tell us the largest possible height (**h**), allowing us to find an upper-bound on height.

BTree Analysis

The minimum number of **nodes** for a BTree of order m **at each level**:

root:

level 1:

level 2:

level 3:

…

level h:

BTree Analysis

$t = \lceil \frac{m}{2} \rceil$

The **total number of nodes** is the sum of all the levels:

$$1 + 2 \sum_{k=0}^{h-1} t^k$$

Summation Identity: $\sum_{i=0}^{n-1} x^i = \frac{x^n - 1}{x - 1}$

BTree Analysis

$$t = \lceil \frac{m}{2} \rceil$$

The **total number of nodes**:

$$1 + 2\frac{t^h - 1}{t - 1}$$

The **total number of keys**:

BTree Analysis

$t = \lceil \frac{m}{2} \rceil$

The **smallest total number of keys** is: $2t^h - 1$

So an inequality about **n**, the total number of keys:

Solving for **h**, since **h** is the max number of seek operations:

BTree Analysis

Given **m=101**, a tree of height **h=4** has:

Minimum Keys:

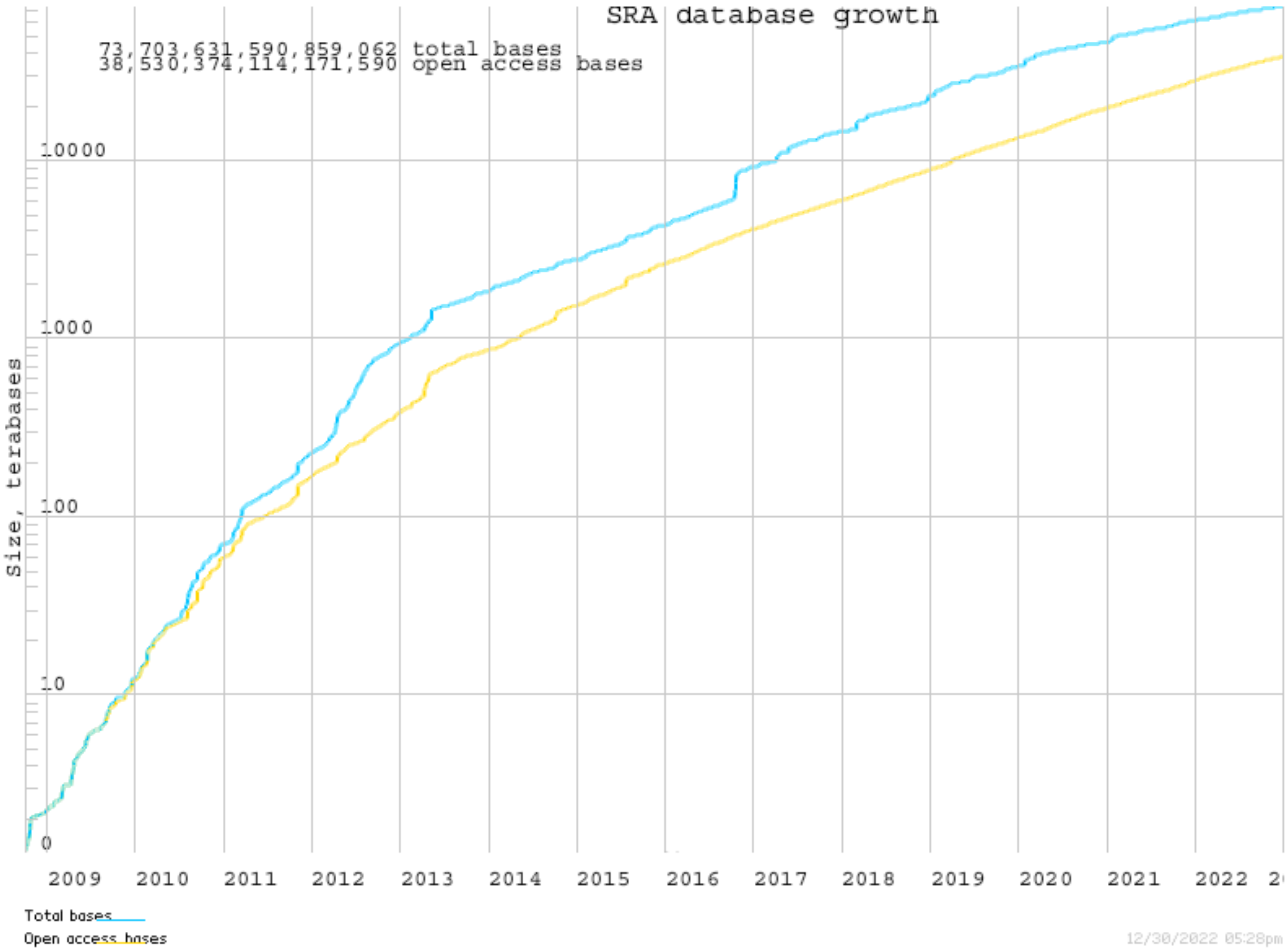
Maximum Keys:

BTree

The BTree is still used heavily today!

Improvements such as B+Tree and B*Tree exist far outside class scope

A story about BIG data



Thinking conceptually: Sorting a queue

How might we build a ‘queue’ in which our front element is the min?

Thinking conceptually: A tree without pointers

What class of (non-trivial) trees can we describe without pointers?