CSE 100:
B TREES (CONTD)
CAPES and End of year survey

• Have you filled out the CAPE form?
A. Yes
B. No

Deadline Mon Dec 15th
Insertion and properties of B-trees

Insert 15 into this B-tree

- Null, so insert 15 in this leaf node.
- This node is full, so it splits into 2 nodes, and the mid element of all keys (including 15) is promoted up.
Insertion and properties of B-trees

Insert 22 and 23
Insertion and properties of B-trees

Insert 16

null → 16 → promote up:
15 16 21 22 23
Insertion and properties of B-trees

Insert 16, after
Insertion and properties of B-trees

True or false: after the insert, the leaves will be at different levels
A. True
B. False
Insertion and properties of B-trees

Insert 62

Which key will be promoted up?
A. 61     B. 62     C. 68     D. 75     E. 80
Insertion and properties of B-trees

Insert 62

Which key will be promoted up?
A. 13  B. 25  C. 60  D. 85  E. 68
Insertion and properties of B-trees

Insert 62
Insertion and properties of B-trees

B-trees grow up! (Which is why all their leaves are always at the same level)
Insertion and properties of B-trees

B-trees grow up! (Which is why all their leaves are always at the same level)
What is the time to find an element in a B-tree of order \( M \) with \( N \) keys?

- What is the maximum height of the tree in terms of \( M \) and \( N \)?

  - Height is maximum, when the min no of keys are stored in each node.
  - The min no. of keys in a B-tree of order \( m \) is \( \lceil \frac{m}{2} \rceil - 1 \). This implies that the no. of children of a node is \( \lceil \frac{m}{2} \rceil \).
  - Calculate the no. of keys at each level of the B-tree using the above facts.

\[
\sum_{i=0}^{H-1} \left( \left\lceil \frac{m}{2} \right\rceil - 1 \right) \leq N
\]

\[
H \leq \log \left( \frac{\lceil \frac{m}{2} \rceil^H}{\left( \left\lceil \frac{m}{2} \right\rceil - 1 \right)^2} \right)
\]

(although the root can have one key, we will treat it as any other intermediate node)
Exact expression for the worst case time to find an element in a B-tree of order $M$ with $N$ keys?

- Maximum height: $\log_{M/2} N$

- Given
  - $T_m$ is the time to access an element in memory
  - $T_d$ is the time to access an element on disk
Exact expression for the worst case time to find an element in a B-tree of order M with N keys?

• Maximum height: \( \log_{M/2} N \)

• Given
  – \( T_m \) is the time to access an element in memory
  – \( T_d \) is the time to access an element on disk

• Time to find an element: \( T_m \log_M M \times \log_{M/2} N + T_d \log_{M/2} N \)

What is the worst case big O run time of find?

A. \( O(\log_2 M) \)
B. \( O(\log_2 N) \)
C. Neither

\[ O \left( (T_m \log_2 M + T_d) \log_{\frac{M}{2}}^2 N \right) = O \left( \log_{\frac{M}{2}} N \right) \]
B-Tree performance

- Worst case $O(\log_2 N)$ which doesn’t reflect their true benefit
- The time savings in a B-Tree comes from *efficiently reading lots of data from disk*
- When B-Trees are stored in memory they are typically comparable to other search trees
- When they have to access disk they are a big win
(2-3-4) B-trees
(2-3-4) B-trees == RBTs!
(2-3-4) B-trees == RBTs!

So which are asymptotically faster? RB-trees or 2-3-4 (B) trees?
A. RB trees
B. B-trees
C. They are the same
B-Tree performance

• Refer Paul Kube’s slides

http://cseweb.ucsd.edu/users/kube/cls/100/Lectures/lec17/lec17.pdf