CS6033 Lecture 5 Slides/Notes

Search Trees: B-Trees; Divide and Conquer: Binary Search, Merge Sort, Integer Multiplication, Matrix Multiplication, Deterministic QuickSelect (Ch 18, Handout & Notes, Ch 4, Sec. 9.3)

By Prof. Yi-Jen Chiang
CSE Dept., Tandon School of Engineering
New York University

1

Generalization of (2,4)-trees: B-trees

t: minimum degree of the B-tree. integer, $t \geq 2$

1. For an internal node $\neq$ root. # children : $t \sim 2t$
   (# keys stored: $t-1 \sim 2t-1$)
2. For the root. # children : $2 \sim 2t$
   (# keys stored: $1 \sim 2t-1$)
3. All leaves are at the same depth.

[(2,4)-tree is a B-tree with $t=2$]

Main Idea & Goal: B-tree is stored on disk. one I/O reads/writes 1 disk block (4k bytes). A B-tree node stores $(t-1) \sim (2t-1)$ keys & $t \sim 2t$ child ptrs

1 disk block fits B items

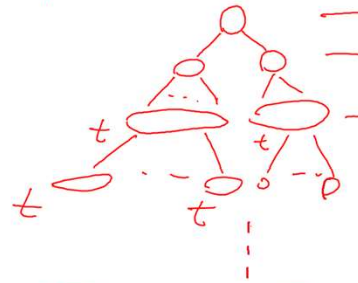
$\Rightarrow$ We want $t = \Theta(B)$. We want: minimize # I/O's in tree access

2

Thm: Height of a B-tree storing n keys is $O(\log_t n)$

Pf: min # keys in a B-tree with height h

m =



# keys	depth
1	0
$2(t-1)$	1
$2t(t-1)$	2
$2t^2(t-1) = 2t^{3-1}(t-1)$	3
⋮	⋮
$2t^{h-1}(t-1)$	h

* Each internal node ≠ root has ≥ t children and ≥ (t-1) keys

$$m = 1 + \left(2 + 2t + 2t^2 + \cdots + 2t^{h-1}\right)\cdot(t-1)$$

$$= 1 + 2\left(1 + t + t^2 + \cdots + t^{h-1}\right)(t-1)$$

$$= 1 + 2\,\frac{t^h - 1}{\cancel{(t-1)}}\cdot\cancel{(t-1)}$$

$$\Rightarrow n \ge m = 1 + 2(t^h - 1)$$

$$= 2\cdot t^h - 1.$$

$$2\cdot t^h \le (n+1) \Rightarrow h \le \log_t\left(\frac{n+1}{2}\right)$$

$$\Rightarrow h = O(\log_t n). \quad \#$$

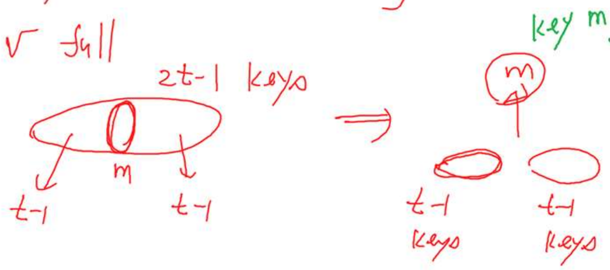
key: $t-1 \sim 2t-1$

One-Pass Insertion:

Invariant: When we visit a node v, v is non-full (# keys ≤ $2t-2$)

Alg: Before we visit a node v, if v is full (with $2t-1$ keys) then we split v before visiting it

e.g. v full



key m goes to the parent

$(t-1) + (t-1) + 1 = 2t-1$

* Note: Since the parent is nonfull, it can accommodate this extra key m with no problem.

One-Pass Deletion: Each node needs $\geq (t-1)$ keys by B-tree rules

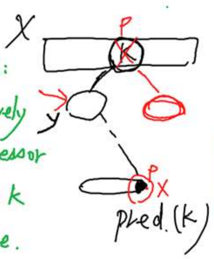
Idea: Before visiting a node v, make sure that v has $\geq t$ keys

Alg: Let k be the key to be deleted, x the current node being visited.

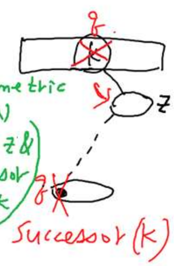
1. If node x contains k and x is a leaf,
   delete k from x. Done.

2. Else, If x contains k and x is an internal node

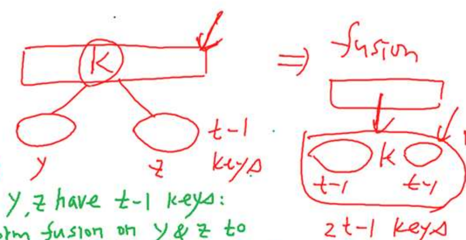
(a) if child y has $\geq t$ keys: Go to y. Recursively delete the predecessor p of k. replace k by p in x. Done.



(b) symmetric to (a) (child z & successor q of k)

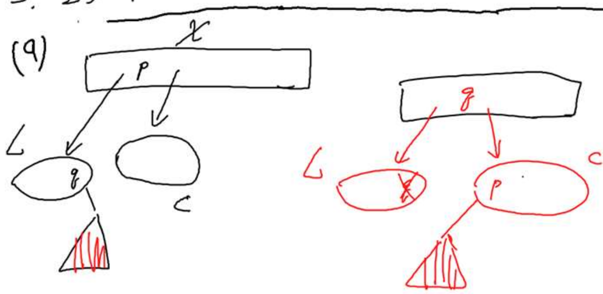


(c) Both y, z have $t-1$ keys: perform fusion on y & z to form node v'. Recurse on v'.



3. If k is not in the current node x, Let c be the child where the search for k should go.

(a)

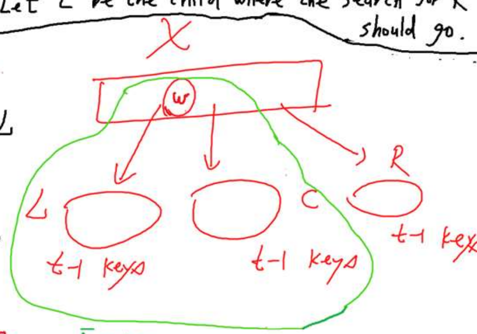


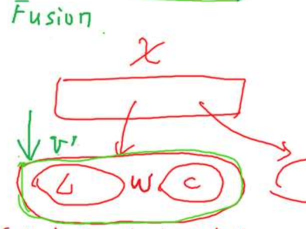
* If c has $\geq t$ keys, recurse on C.
Otherwise, c has $t-1$ keys

(a) The left sibling L of C has $\geq t$ keys:
Perform Transfer on L & C. (see fig. above). Recurse on C.

Else, if the right sibling R of C has $\geq t$ keys:
Perform Transfer on R & C (in a symmetric way).
Recurse on C.

(b) Both L and R have $t-1$ keys: Perform Fusion on L & C (or on C & R) to form node v'. Recurse on v'.





Q: What should we do when trying to visit the root?

A: Directly visit the root (i.e., apply this Alg. with current node x = root) even if the root has only 1 key.

* Then when trying to visit a **child** of the root in Case 2 or Case 3, a transfer or a fusion may occur.
If a fusion occurs and the root becomes empty (i.e., has no key), then the root is removed.

Comparison: In the one-pass **insertion algorithm**, before visiting the root, if the root is full, we **first split the root,** then visit the (new) root.

7

Divide & Conquer

2 Major Topics:

(1) Algorithm Design:

a. Subdivide the problem into subproblems.

b. Solve subproblems recursively.

c. Combine the solutions of subproblems into a solution for the current problem. [may need enhanced info from subproblems.]

(2) Solving Recurrences

Typically 4 main methods:

(a) Baby Master Theorem.
for some special form.

(b) Repeated Unfolding (e.g. We used it to derive Baby Master Thm)

(c) Recursion Tree

(d) Substitution.

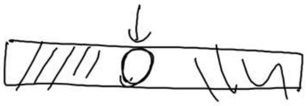
8

Divide & Conquer

* Solve Recurrences.

* Alg. Design

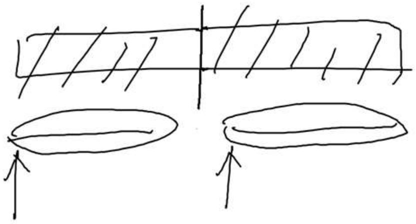
Ex.1. Binary Search.



$$\begin{cases} T(n) = T\left(\frac{n}{2}\right) + 1 \\ T(1) = \text{const.} \end{cases}$$

$a=1 \quad b=2.$ $\theta(n^d) = \theta(1)$ $d=0.$ $\frac{a}{b^d} = \frac{1}{2^0} = 1 \Rightarrow T(n) = \theta(n^d \log n) = \theta(\log n)$

Ex2. Merge Sort.



1 4 5 7 | 2 3 8 10.

1 2 3 4 5 7 8 10.

$$\begin{cases} T(n) = 2T\left(\frac{n}{2}\right) + \overset{\theta(n)}{cn} \\ T(1) = \text{const.} \end{cases}$$

$a=2. \quad b=2 \quad d=1$

$\frac{a}{b^d} = \frac{2}{2^1} = 1.$

$T(n) = \theta(n^d \log n) = \theta(n \log n)$

*** Discussed the handout "Baby-Master-Thm.pdf" for ``Baby'' Master Theorem.**

9

Ex3: Integer Multiplication

2 integers X, Y. each with n digits.

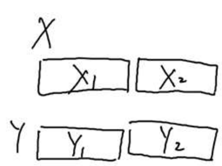
Compute $X \cdot Y$. $\boxed{\text{Assume: } n \text{ is a power of } 2.}$

Naively: $O(n^2)$ time.

Divide & Conquer:

Let $X = X_1 B^{\frac{n}{2}} + X_2$

$Y = Y_1 B^{\frac{n}{2}} + Y_2.$



X_1, X_2, Y_1, Y_2 : $\frac{n}{2}$ digits each.

B : base.

M1: $X \cdot Y = (X_1 Y_1) B + (X_1 Y_2 + X_2 Y_1) B^{\frac{n}{2}} + X_2 Y_2.$

$= Z_1 B + Z_2 B^{n/2} + Z_3.$

$Z_1 = \underbrace{X_1 Y_1}.$ $Z_2 = \underbrace{X_1 Y_2} + \underbrace{X_2 Y_1}.$ $Z_3 = \underbrace{X_2 Y_2}$

$T(n)$: time for multiplying 2 integers of n digits each.

$T(\frac{n}{2})$: 〃 〃 $\frac{n}{2}$ digits each.

$T(n)$: time for $X \cdot Y$.

$T(\frac{n}{2})$: 〃 〃 $\underline{X_1 Y_1}$. $\underline{X_1 Y_2}$. $\underline{X_2 Y_1}$. $\underline{X_2 Y_2}$

$T(n) = 4T(\frac{n}{2}) + cn$ $\quad c$: const

10

$T(n) = 4\,T\left(\frac{n}{2}\right) + \theta(n)$.

$a = 4.\ b = 2.\ d = 1$

$\frac{a}{b^d} = \frac{4}{2^1} > 1$

$\Rightarrow T(n) = \theta\left(n^{\log_b a}\right)$

$= \theta\left(n^{\log_2 4}\right) = \theta(n^2)$

M2: $X \cdot Y = (X_1 Y_1) B + (X_1 Y_2 + X_2 Y_1) B^{\frac{n}{2}} + X_2 Y_2$

$= Z_1 B + Z_2 B^{\frac{n}{2}} + Z_3$.

$Z_1 = X_1 Y_1$. $Z_2 = X_1 Y_2 + X_2 Y_1$ $Z_3 = X_2 Y_2$.

$X = X_1 B^{\frac{n}{2}} + X_2$

$Y = Y_1 B^{\frac{n}{2}} + Y_2$.

Let $C = X_1 + X_2$. $D = Y_1 + Y_2$. $\frac{n}{2}$ digits each

$C \cdot D$: $T\left(\frac{n}{2}\right)$.

$C \cdot D = (X_1 + X_2)(Y_1 + Y_2)$

$= \boxed{X_1 Y_1} + X_2 Y_1 + X_1 Y_2 + \boxed{X_2 Y_2}$

$= Z_1 + \underbrace{(X_1 Y_2 + X_2 Y_1)}_{= Z_2} + Z_3$.

$= Z_1 + Z_2 + Z_3$. $\Rightarrow Z_2 = CD - Z_1 - Z_3$.

Compute: $Z_1 = X_1 Y_1$ $Z_3 = X_2 Y_2$. CD

$\Rightarrow Z_2 = CD - Z_1 - Z_3$.

11

$\Rightarrow T(n) = 3\,T\left(\frac{n}{2}\right) + \theta(n)$.

$a = 3.\ b = 2.\ d = 1$.

$\frac{a}{b^d} = \frac{3}{2^1} > 1$

$T(n) = \theta\left(n^{\log_b a}\right)$

$= \theta\left(n^{\log_2 3}\right)$

Better than $\theta(n^2)$

Ex 4: Matrix Multiplication:

$A \cdot B$ each of A, B is an $n \times n$ matrix

$A \cdot B = C$ $\quad C_{ij} = \sum_{k=1}^{n} A_{ik} \cdot B_{kj}$ $\quad$ n time. n^2 items of C_{ij}

$\Rightarrow \theta(n^3)$ time.

Divide & Conquer: Assume n is a power of 2.

$A = \left[\begin{array}{c|c} A_1 & A_2 \\ \hline A_3 & A_4 \end{array}\right]$. $\quad B = \left[\begin{array}{c|c} B_1 & B_2 \\ \hline B_3 & B_4 \end{array}\right]$ $\quad A_i, B_j : \frac{n}{2} \times \frac{n}{2}$ matrix.

$A \cdot B = \begin{bmatrix} A_1 B_1 + A_2 B_3 & A_1 B_2 + A_2 B_4 \\ A_3 B_1 + A_4 B_3 & A_3 B_2 + A_4 B_4 \end{bmatrix}$

$T(n) = 8\,T\left(\frac{n}{2}\right) + \theta(n^2)$.

$a = 8.\ b = 2.\ d = 2$.

$\frac{a}{b^d} = \frac{8}{2^2} = 8/4 > 1$

12

$$\Rightarrow T(n) = \Theta\left(n^{\log_b a}\right) = \Theta\left(n^{\log_2 8}\right) = \Theta\left(n^{3}\right)$$

No Better !!

Strassen's Alg. :

Only needs 7 subproblems of $\left(\frac{n}{2} \times \frac{n}{2}\right)$ matrix multiplications.

Given in the textbook.

(Highly non-trivial).

$$T(n) = 7\,T\left(\frac{n}{2}\right) + \Theta(n^2)$$

$a = 7.\ b = 2.\ d = 2.\quad \frac{a}{b^d} = \frac{7}{2^2} > 1.$

$$\Rightarrow T(n) = \Theta\left(n^{\log_b a}\right) = \Theta\left(n^{\log_2 7}\right) \quad \left(\text{Better than } \Theta(n^3)\right)$$

✓

13

Ex : Deterministic Linear-Time Selection Alg.

Given a sequence of n unsorted items and an integer $k \in [1, n]$, find the k-th smallest item in $O(n)$ worst-case time.

Alg : 0. If ($n \bmod 5 \neq 0$)

{ Find the min and take it out : $\quad$ min $\quad \overbrace{\qquad\qquad}^{S'}$ remaining $(n-1)$ items

If ($k == 1$) return min

else { $k \leftarrow k-1$, $n \leftarrow n-1$ $\quad$ Recurse on S' }

}

At most $O(n) \cdot 4 = O(n)$ time to make ($n \bmod 5 == 0$)

// Now ($n \bmod 5 == 0$)

$O(n)$ $\quad$ 1. Partition the current set of n items into $\frac{n}{5}$ groups of 5 items each.

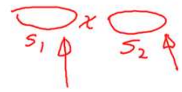
14

$O(n)$ 2. Sort each group of 5 items, take the median item from it. ($\frac{n}{5}$ medians)

$T(\frac{n}{5})$ 3. Apply the alg. recursively on the $\frac{n}{5}$ medians and find the median x among them. (x is the median among the medians)

$O(n)$ 4. Compare all items with x to get sets S_1 (items $< x$)
S_2 (items $> x$).

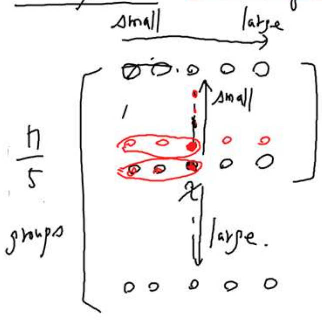
Count $|S_1|, |S_2|$



$T(|S_1|)$ or $T(|S_2|)$ 5. Recurse on either S_1 or S_2 (or report x and stop)

15

Analysis: size of S_2 (size of S_1 is symmetric)



Consider the # of items that are $\leq x$.

Call this # t.

$t \geq \left(\frac{n}{5} \cdot \frac{1}{2}\right) \cdot 3$

[For each of the $\frac{n}{5} \cdot \frac{1}{2}$ groups, x is $\geq$ 3 items (for the group containing x, this includes x)]

$$t \geq \frac{n}{5} \cdot \frac{1}{2} \cdot 3 = \frac{3n}{10}$$

S_2: items $> x$.

$$|S_2| = n - t \leq n - \frac{3n}{10} = \frac{7n}{10}$$

$$\boxed{T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + an \quad a: \text{const.}}$$

Solve by substitution:

Let $T(n) = cn$ for some const c.

$$cn \leq c\left(\frac{n}{5}\right) + c\left(\frac{7n}{10}\right) + an$$
$$\leq c \cdot \frac{9}{10} n + an$$

$\Rightarrow \frac{1}{10} c \leq a \Rightarrow$ Take $c = 10a$. $T(n) = cn = 10an = O(n)$ ※

16

Supplementary: Earlier/Standard Version
(slightly more complicated in analysis)

$T(n)$

Alg:

$O(n)$ 1. Partition the n items into $\lfloor \frac{n}{5} \rfloor$ groups of 5 items each, leaving at most 4 left-over items.

$O(n)$ 2. Sort each group of 5 items, take the median item from it. ($\lfloor \frac{n}{5} \rfloor$ medians)

$T(\lfloor \frac{n}{5} \rfloor)$ 3. Apply the alg. recursively on the $\lfloor \frac{n}{5} \rfloor$ medians and find the median x among them. (x is the median among the medians)

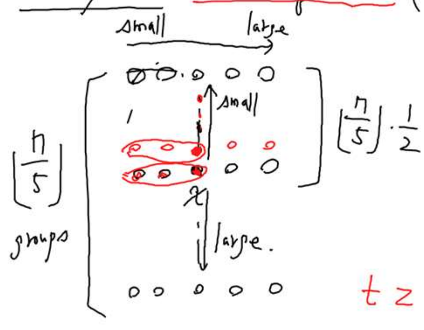
$O(n)$ 4. Compare all items with x to get sets S_1 (items $< x$), S_2 (items $> x$).

$T(|S_1|)$ or $T(|S_2|)$ 5. Recurse on either S_1 or S_2 (or report x & stop)

Count $|S_1|$, $|S_2|$

17

Analysis: size of S_2 (size of S_1 is symmetric)



Consider the # of items that are $\le x$.

Call this # t.

$$t \ge \left(\left\lfloor \frac{n}{5} \right\rfloor \cdot \frac{1}{2}\right) \cdot 3$$

[For each of the $\lfloor \frac{n}{5} \rfloor \cdot \frac{1}{2}$ groups, x is $\ge$ 3 items. (for the group containing x, this includes x)]

$$t \ge \left(\frac{n}{5} - 1\right) \cdot \frac{1}{2} \cdot 3 = \frac{3n}{10} - 1.5 \ge \frac{3n}{10} - 2.$$

S_2: items $> x$.

$$|S_2| = n - t \le n - \left(\frac{3n}{10} - 2\right) = \frac{7n}{10} + 2.$$

$$T(n) \le \underbrace{T\left(\left\lfloor \frac{n}{5} \right\rfloor\right)}_{\le T(n/5)} + \underbrace{T\left(\frac{7n}{10} + 2\right)}_{\le T(Dn)} + an \quad a: \text{const.}$$

Simplify $T(\frac{7n}{10} + 2)$.

Goal: Find some consts. D, E s.t. $T(\frac{7n}{10} + 2) \le T(Dn)$ for $n \ge E$.

and $\frac{n}{5} + Dn < n$

18

$\frac{n}{5} = 0.2 \cdot n$. $\frac{7n}{10} = 0.7n$ $\frac{7n}{10}+2 = Dn$ $D > 0.7$, $\underline{D + 0.2 < 1}$. Eg. Take $D = 0.75 = \frac{3}{4}$.

$\frac{7n}{10}+2 \le \frac{3}{4}n$. $\underline{2 \le} \; n\left(\frac{3}{4} - \frac{7}{10}\right) = n(0.75 - 0.7) = \underline{(0.05)n}$. True when $n \ge \frac{2}{0.05} = \underline{40}$.

$\frac{7}{10}n + 2 \le \frac{3}{4}n$ when $n \ge 40$.

$T(n) \le T\left(\left\lfloor\frac{n}{5}\right\rfloor\right) + T\left(\frac{7n}{10}+2\right) + an$

$$\boxed{T(n) \le T\left(\frac{n}{5}\right) + T\left(\frac{3n}{4}\right) + an \text{ when } n \ge 40.}$$

Solve it by substitution

Let $T(n) = cn$ for some const. c.

$\underline{cn} \le \frac{1}{5}(\underline{cn}) + \frac{3}{4}(\underline{cn}) + an$

$cn\left(1 - \frac{1}{5} - \frac{3}{4}\right) \le an$

$c(1 - 0.2 - 0.75) = c \cdot (1 - 0.95)$

$= c \cdot 0.05 = \frac{c}{20} \le a \Rightarrow$ Take $c = 20a$.

$\underline{T(n)} \le cn = 20an = \underline{O(n)}$ #

19