Algorithms
Professor John Reif

ALG 1.3

Deterministic Selection and Sorting:

(a) Selection Algorithms and Lower Bounds
(b) Sorting Algorithms and Lower Bounds

Main Reading Selections:
CLR, Chapters 7, 9, 10
Auxillary Reading Selections:
AHU-Design, Chapters 2 and 3
AHU-Data, Chapter 8
BB, Sections 4.4, 4.6 and 10.1

Problem P size n

⇒ divide into *subproblems* size $n_1, \ldots, n_k$
solve these and "glue" together solutions

$$T(n) = \sum_{i=1}^{k} T(n_i) + g(n)$$

time to combine solutions

Examples:

1st lecture's *mult* $M(n) = 3\,M\left(\left\lceil \frac{n}{2} \right\rceil\right) + \theta(n)$

fast fourier *transform* $F(n) = 2\,F\left(\left\lceil \frac{n}{2} \right\rceil\right) + \theta(n)$

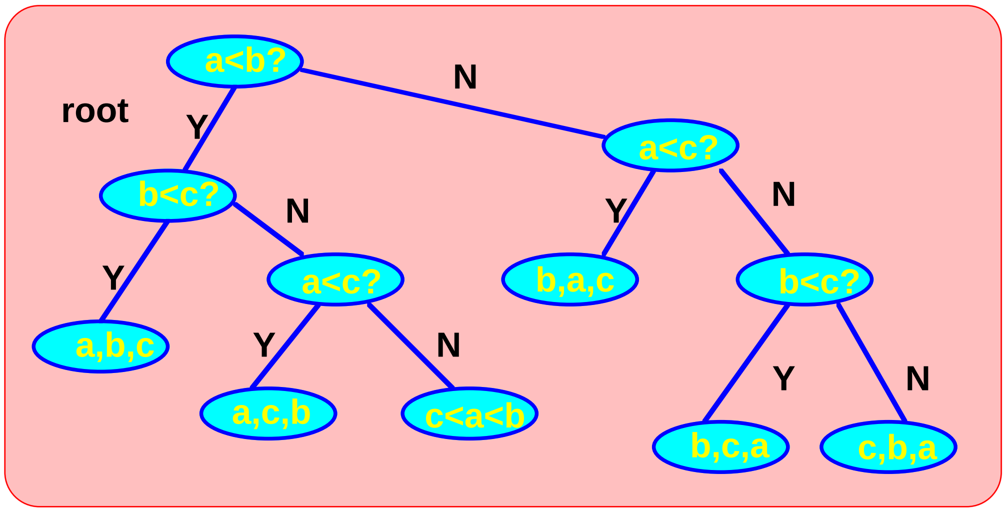
binary *search* $B(n) = B\left(\left\lceil \frac{n}{2} \right\rceil\right) + \theta(1)$

merge *sorting* $S(n) = 2 \cdot S\left(\left\lceil \frac{n}{2} \right\rceil\right) + \theta(n)$

Examples

Selection, and Sorting on *Decision Tree Model*

input a,b,c



$$Time = \left(\begin{array}{l} \text{\# comparisons} \\ \text{on longest path} \end{array} \right)$$

Binary tree

with L Leaves

facts: (1) has = L-1 internal nodes

(2) max height $\geq \lceil \log L \rceil$

Merging

2 lists with total of n keys

input $X_1 < X_2 < \ldots < X_k$

and

$Y_1 < Y_2 < \ldots < Y_{n-k}$

output

ordered merge of two key lists

goal

provably asymptotically optimal algorithm in

Decision Tree Model

use this Model because it

allows *simple proofs* of lower bounds

time = # comparisons so easy to bound time costs

Algorithm Insert

input $(X_1 < X_2 < \ldots < X_k), (Y_1)$

Case k=n-1

Algorithm : Binary Search

by Divide-and-Conquer

[1] Compare Y_1 with $X_{\lceil \frac{k}{2} \rceil}$

[2] *if* $Y_1 > X_{\lceil \frac{k}{2} \rceil}$ *insert* Y_1 into

$$\left(X_{\lceil \frac{k}{2} \rceil + 1} < \ldots < X_k \right)$$

else $Y_1 \leq X_{\lceil \frac{k}{2} \rceil}$ and *insert* Y_1 into

$$\left(X_1 < \ldots < X_{\lceil \frac{k}{2} \rceil} \right)$$

Total Comparison Cost:

$\leq \lceil \log(k+1) \rceil = \lceil \log(n) \rceil$

Since a binary tree with n=k+1 leaves has depth $> \lceil \log(n) \rceil$, this is ***optimal!***

Case: Merging equal length lists

Input $(X_1 < X_2 < \dots < X_k)$

$(Y_1 < Y_2 < \dots < Y_{n-k})$

where $k = \frac{n}{2}$

Algorithm

[1] $i \leftarrow 1$, $j \leftarrow 1$

[2] *while* $i \leq k$ and $j \leq k$ *do*

if $X_i < Y_j$ *then output* (X_i) and set $i \leftarrow i+1$

else output (Y_j) and set $j \leftarrow j+1$

[3] *output* remaining elements

Algorithm clearly uses 2k-1 = ***n-1 comparisons***

Lower bound:

consider case $X_1 < Y_1 < X_2 < Y_2 < \ldots < X_k < Y_k$
any merge algorithm must compare

claim:

(1) X_i with Y_i for i=1 ,..., k
(2) Y_i with X_{i+1} for j=1 ,..., k-1

(otherwise we could *flip* $Y_i < X_i$ with no change)
$\Rightarrow$ so requires $\geq$ 2k-1 = *n-1 comparisons!*

Sorting by Divide-and-Conquer

Algorithm Merge Sort

input set S of n keys

[1] *partition* S into set X of $\lceil \frac{n}{2} \rceil$ keys
and set Y of $\lfloor \frac{n}{2} \rfloor$ keys

[2] *Recursively* compute
Merge Sort (X) = $(X_1, X_2, \ldots, X_{\lceil \frac{n}{2} \rceil})$

Merge Sort (Y) = $(Y_1, Y_2, \ldots, Y_{\lfloor \frac{n}{2} \rfloor})$

[3] *merge* *above sequences*
using n-1 comparisons

[4] *output* merged sequence

Time Analysis

$$T(n) = T\left(\left\lceil \frac{n}{2} \right\rceil\right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n-1$$

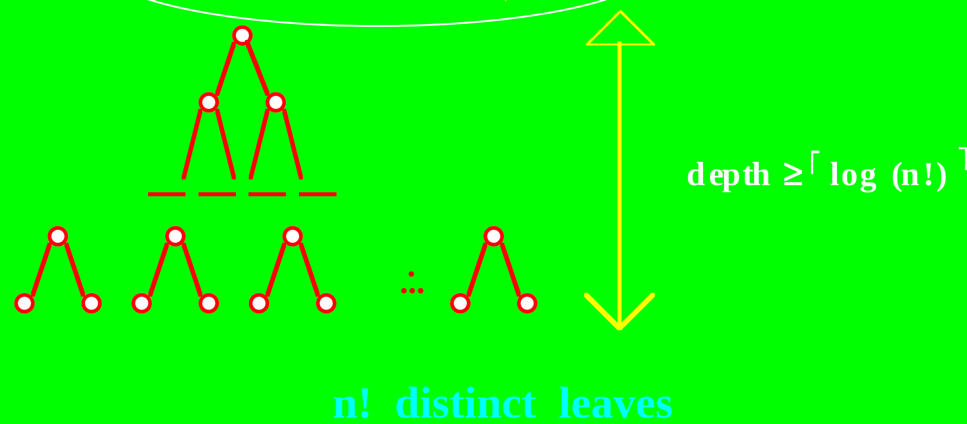
$$T(l) = 0$$

$$\Rightarrow T(n) = n \lceil \log n \rceil - 2^{\lceil \log n \rceil} + 1$$

$$= \theta(n \log n)$$

Lower Bounds on Sorting

(on decision tree model)



Easy Approximation (via Integration)

$$\log(n!) = \log(n) + \log(n-1) + \ . \ . \ . + \log(2) + \log(1)$$

$$\geq \int_{n-1}^{n} \log x\, dx + \ldots + \int_{1}^{2} \log x\, dx$$

$$\geq \int_{1}^{n} \log x\, dx \ \left(\text{Since } \log k \geq \int_{k-1}^{k} \log x\, dx\right)$$

$$\geq n \log n - n \log e + \log e$$

Better bound **using** ***Sterling Approximation***

$$n! \geq \sqrt{2\pi n} \left(\frac{n}{e}\right)^{n} \left(1 + \frac{1}{12}\, n\right)$$

$$\Rightarrow \log(n!) \geq n \log n - n \log e + \frac{1}{2} \log(2\pi n)$$

Selection Problems

input $X_1, X_2, \ldots, X_n$
and index $k \in \{1,\ldots,n\}$

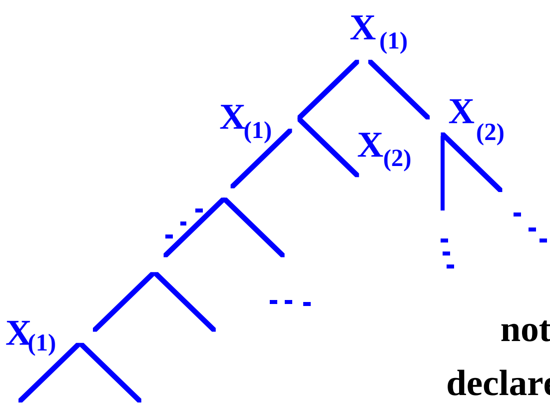
output $x_{(k)}$ = the k'th best

History:

Rev C.L. Dodge (Lewis Carol)
wrote article on lawn tennis
tournament in James Gazett, 1883

felt *prizes unjust* because:

- although winner $X_{(1)}$ always gets lst prize
- second $X_{(2)}$ may *not* get 2nd prize

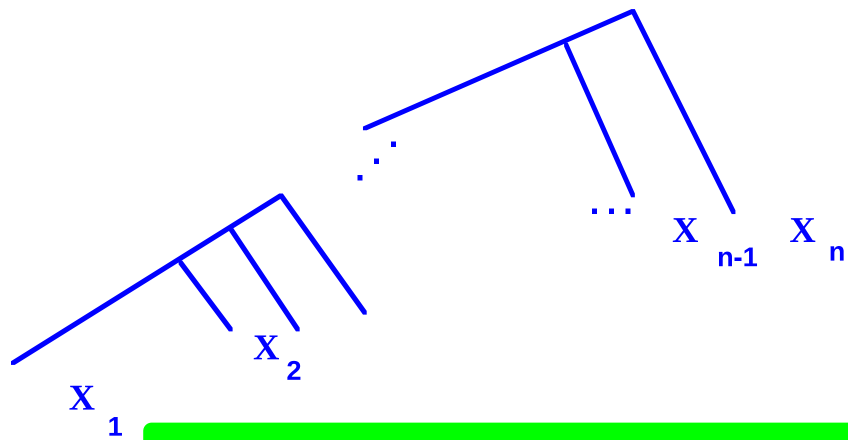


note: $X_{(2)}$ *not* declared 2nd best if it is *left branch*

Carol proposed his own (nonoptimal) tournament....

Selection of the champion $X_{(1)}$

- $X_{(1)}$ is easily determined in n-1 comparison



- $X_{(1)}$ requires *n-1 comparisons*

proof

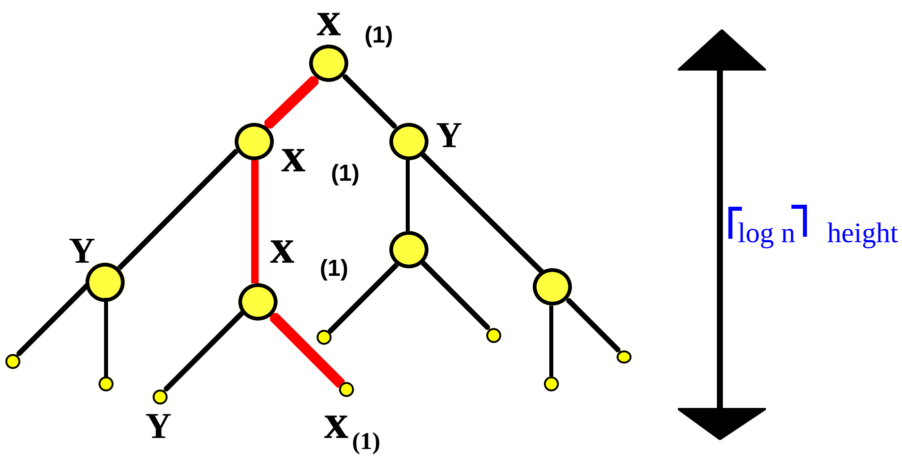
everyone except the champion $X_{(1)}$ must lose at least once!

Selection of the second best $X_{(2)}$

using ***n-2 +*** $\lceil$***logn***$\rceil$ ***comparisons***

Algorithm

[1] form a ***balanced binary tree***
for tournament to find $X_{(1)}$
using ***n-1 comparisons***



[2] Let Y be the set of players knocked out by champion $X_{(1)}$

$|Y| \leq \lceil \log n \rceil$

[3] Play a tournament among the Y's

[4] ***output*** $X_{(2)}$ **= champion of the Y's**

using $\lceil$**logn**$\rceil$ **-1 more comparisons**

Lower Bounds on finding $X_{(2)}$

requires $\geq$ ***n-2 +*** $\lceil \log n \rceil$ ***comparisons***

proof

#comparison $\geq m_1 + m_2 + \ldots$

where m_i **= #players who lost i or more matches**

Claim $m_1 \geq n-1$, **since at end we must know** $X_{(1)}$ **as well as** $X_{(2)}$

Claim $m_2 \geq (\#\text{who lost to } X_{(1)})-1$ **since everyone (except** $X_{(2)}$**) who lost to** $X_{(1)}$ **must also have lost one more time.**

lemma

(#who lost to $X_{(1)}$) $\geq \lceil \log n \rceil$ in worst case

proof

Use oracle who "fixes" results of games so that champion $X_{(1)}$ plays $\geq \lceil \log n \rceil$ matches

declare

$X_i > X_j$ if

(a) X_i previously undefeated and X_j lost at least once

(b) both undefeated but X_i played more matches

(c) otherwise, decide consistently with previous decisions

$\Rightarrow$ forces path from $X_{(1)}$ to root to have length $\geq \lceil \log n \rceil$

Selection by Divide-and-Conquer

Algorithm Select $_k$ (X)

input set X of n keys and index k

[1] *if* $n < c_o$ *then* output $X_{(k)}$ by sorting X and halt

[2] divide X into $\lfloor \frac{n}{d} \rfloor$ sequences of d elements each (with < d leftover), and *sort* each sequence

[3] let M be the *medians* of each of these sequences

[4] $m \leftarrow \text{Select}_{\frac{|M|}{2}}(M)$

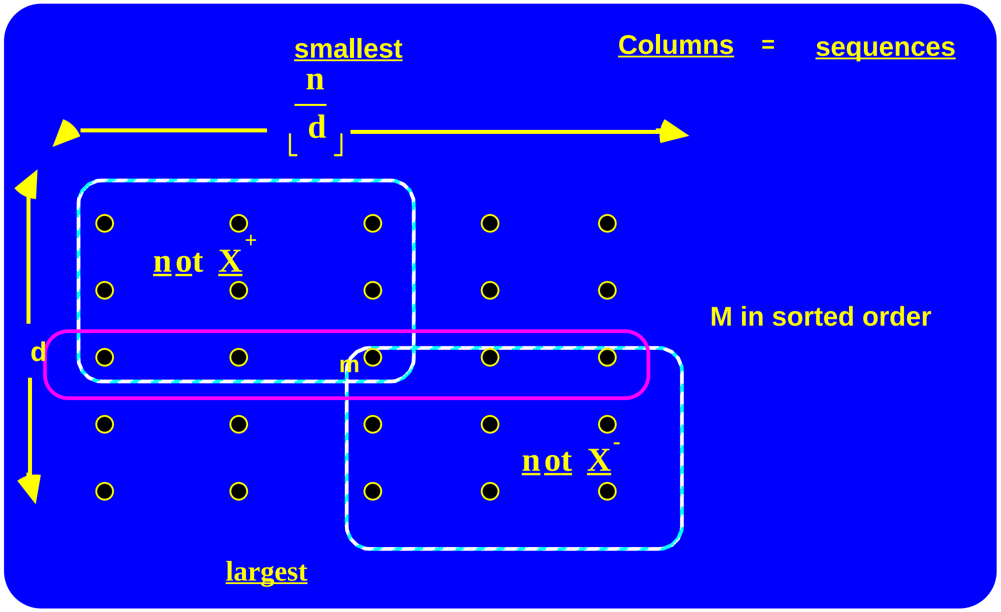
[5] let $X^- = \{x \;\varepsilon\; X \mid x < m\}$

let $X^+ = \{x \;\varepsilon\; X \mid x > m\}$

[6] if $|X^-| \geq k$ then *output* $\text{Select}_k(X^-)$

else if $n - |X^+| = k$ then *output* m

else *output* $\text{Select}_{k-(n - |X^+|)}(X^+)$



Proposition $|X^-|, |X^+|$ each $\leq n - \left\lfloor \frac{d+1}{2} \right\rfloor \cdot \left\lfloor \frac{n}{2d} \right\rfloor \leq \frac{3}{4} n$

$$T(n) \leq \begin{cases} c_1 & \text{if } n < c_o \\ T\left(\left\lfloor \frac{n}{d} \right\rfloor\right) + T\left(\frac{3}{4} n\right) + c_1 n \end{cases}$$

for a sufficiently large constant c_1

(assuming d is constant)

If say d=5, $T(n) \leq 20 c_1 n = O(n)$

Lower Bounds for Selecting $X_{(k)}$

input $X = \{x_1, ..., x_n\}$, index k

Theorem Every leaf of Decision Tree has depth $\geq$ n-1

proof

Fix a path p from root to leaf

The comparisons done on p define a *relation R_p*

Let R_p^+ = *transitive closure* of R_p

Lemma If path p determines $X_m = X_{(k)}$

then for all $i \neq m$ either $x_i \, R_p^+ \, x_m$ or $x_m \, R_p^+ \, x_i$

proof Suppose x_i is *un related to* x_m by R_p^+

Then can replace x_i in linear order either before or after x_m to violate $x_m = x_{(k)}$

Let the *"key" comparison* for x_i be when x_i is compared with x_j where either

(1) j=m

(2) $x_i \; R_p \; x_j$ and $x_j \; R_p^+ \; x_m$, or

(3) $x_j \; R_p \; x_i$ and $x_m \; R_p^+ \; x_j$

Fact x_i has unique "key" comparison determining either $x_i \; R_p^+ \; x_m$ or $x_m \; R_p^+ \; x_i$

$\Rightarrow$ So there are n-1 "key" comparisons, each distinct!

A hard to analyze sort:

SHELLSORT

input keys $X_1, \ldots, X_n$

(1) $\Delta \leftarrow \lfloor \frac{n}{2} \rfloor$

(2) *while* $\Delta > 0$ *do*
for $i = \Delta + 1$ to n *do*

increment sort

begin
$j \leftarrow i - \Delta$
while $j>0$ *do*
if $x_j > x_{j+\Delta}$ *then*
begin
SWAP ($x_j, x_{j+\Delta}$)
$j \leftarrow j - \Delta$
end
else $j \leftarrow 0$¨
end
$\Delta \leftarrow \lfloor \Delta / 2 \rfloor$

AHU Data Structures & Alg., pp. 290-291

passes of SHELLSORT:

1 increment sort $\left(X_k, X_{\frac{n}{2}+k}\right)$ for $k=1,\dots,\frac{n}{2}$

2 increment sort $\left(X_k, X_{\frac{n}{4}+k}, X_{\frac{n}{2}+k}, X_{\frac{3}{4}n+k}\right)$ for $k=1,\dots,\frac{n}{4}$

procedure

increment sort $(Y_1,\dots,Y_l)$

$\Big($ *for* i = z *by* 1 *until* i>n or $X_{i-1} < X_i$

do for j=1 by -1 until 1 do

if $X_{j-1} > X_j$ *then* swap $\left(X_{j-1}, X_j\right)$

facts (1) if $X_i, X_{\frac{n}{p^2}+1}$ sorted in pass p

$\Rightarrow$ they remain sorted in later passes

(2) distance between comparisons diminish

as $\frac{n}{2}, \frac{n}{4}, \dots, \frac{n}{p^2}, \dots$

(3) The best known time bound is $0\left(n^{1.5}\right)$

procedure RADIXSORT

input $X_1, \ldots, X_n$ ε {1 ,..., n}

[1] *for* j=1 ,..., n *do*
initialize B[j] to be the empty list

[2] *for* i=1 ,..., n *do*
add i to B[X_i]

[3] let L = (i_1, i_2 ,..., i_n) be the concatenation of B[1] ,..., B[n]

[4] *output* $X_{i_1} \leq X_{i_2} \leq \ldots \leq X_{i_n}$

-- *Costs O(n) time* on unit cost RAM

-- *avoids* Ω(nlogn) lower bound on SORT
by avoiding comparisons
instead uses indexing of RAM

-- generalizes (in c passes) to key
domains $\{1, \ldots, n^c\}$

open problems in sorting

(1) ***Complexity of SHELLSORT***

-- very good in practice claims Sedgewick

-- Is it $\theta(n^{1.5})$**?**

(2) ***Complexity of variable length***

-- ***sort*** **on multitape TM or RAM**

-- Is it $\Omega(n \log n)$**?**