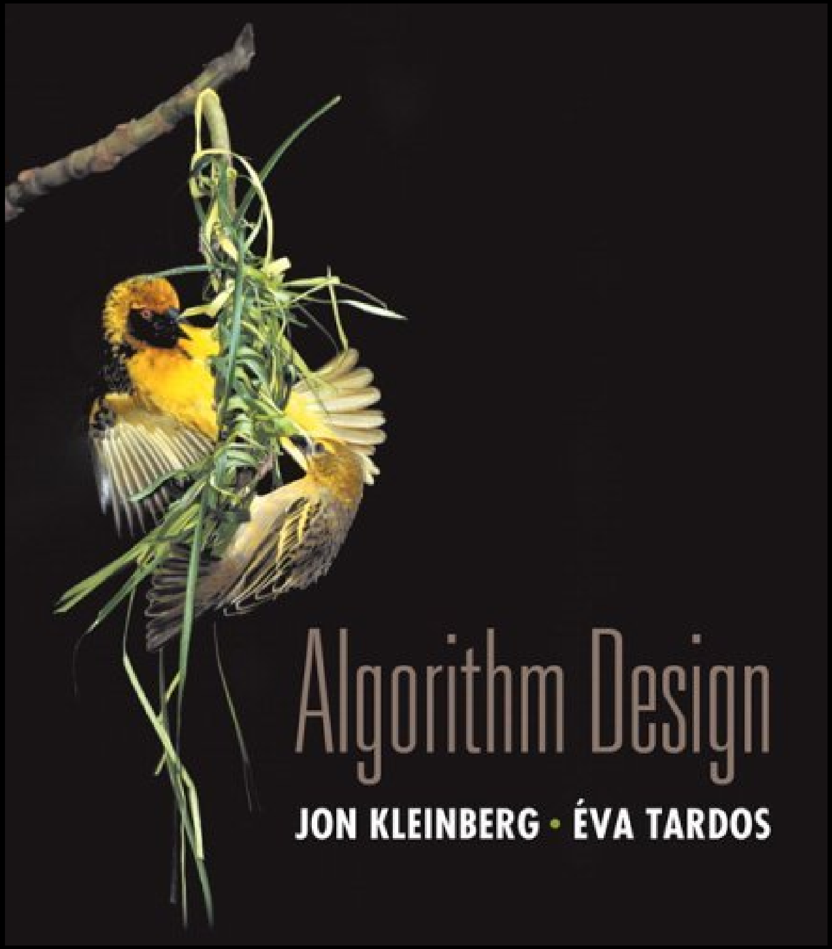


Additional Subject:

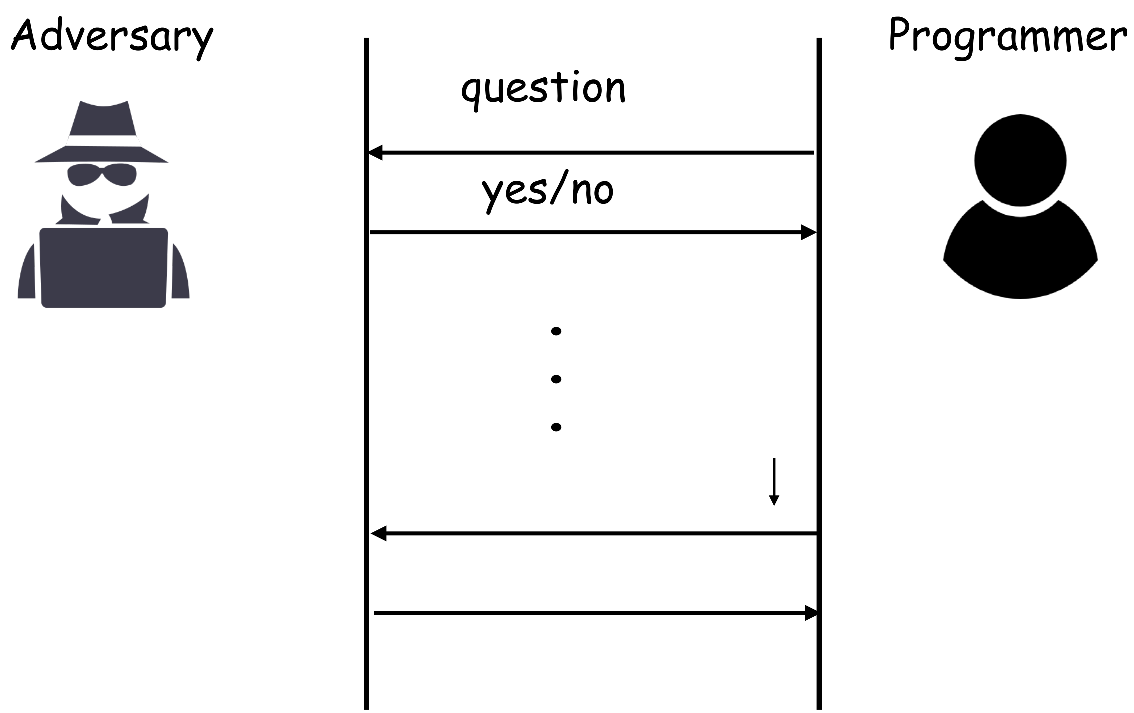
Adversary Argument

PEARSON
Addison Wesley

Adversary Arguments

Adversary: force the programmer to ask as many questions as possible

Constraint: adversary's answers have to be "consistent"



Adversary gradually construct a "bad" input for the programmer

Finding max and min

Problem: finding max and min for 2k keys

Solution: (1) compare k pairs, (2) find max (min) among winners (losers)
one win and one lose as one unit of information

Lower bound: 3n/2 – 2 (total information needed: 2n-2, n-1 wins and n-1 loses. n/2 comparisons of unseen keys following by n-2 operations)

Status of keys x and y compared by an algorithm	Adversary response	New status	Units of new information
N, N	$x > y$	W, L	2
W, N or WL, N	$x > y$	W, L or WL, L	1
L, N	$x < y$	L, W	1
W, W	$x > y$	W, WL	1
L, L	$x > y$	WL, L	1
W, L or WL, L or W, WL	$x > y$	No change	0
WL, WL	Consistent with assigned values	No change	0

Finding max and min: adversary in action

Interactions between the adversary and programmer

Comparison	x_1 Status	x_1 Value	x_2 Status	x_2 Value	x_3 Status	x_3 Value	x_4 Status	x_4 Value	x_5 Status	x_5 Value	x_6 Status	x_6 Value
x_1, x_2	W	20	L	10	N	*	N	*	N	*	N	*
x_1, x_5	W	20							L	5		
x_3, x_4					W	15	L	8				
x_3, x_6					W	15					L	12
x_3, x_1	WL	20			W	25						
x_2, x_4			WL	10			L	8				
x_5, x_6									WL	5	L	3
x_6, x_4							L	2			WL	3

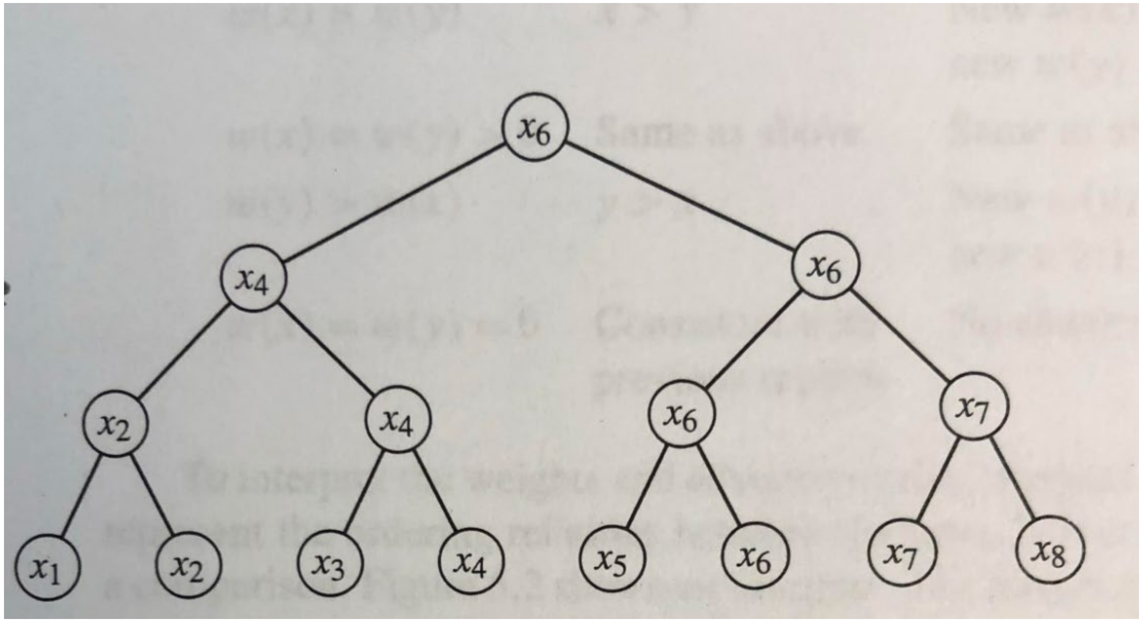
Finding the second-largest key

Problem: finding the second-largest key

Solution: (1) applies a knockout tournament

(2) uses the knockout again among the losers to the largest key

Lower bound: $n + \lceil \lg n \rceil + 1$



Finding the second-largest key: adversary

Case	Adversary reply	Updating of weights
$w(x) > w(y)$	$x > y$	New $w(x) =$ prior $(w(x) + w(y))$; new $w(y) = 0$.
$w(x) = w(y) > 0$	Same as above.	Same as above.
$w(y) > w(x)$	$y > x$	New $w(y) =$ prior $(w(x) + w(y))$; new $w(x) = 0$.
$w(x) = w(y) = 0$	Consistent with previous replies.	No change.

First knockout: n-1

Second knockout: ⌈lgn⌉ -1

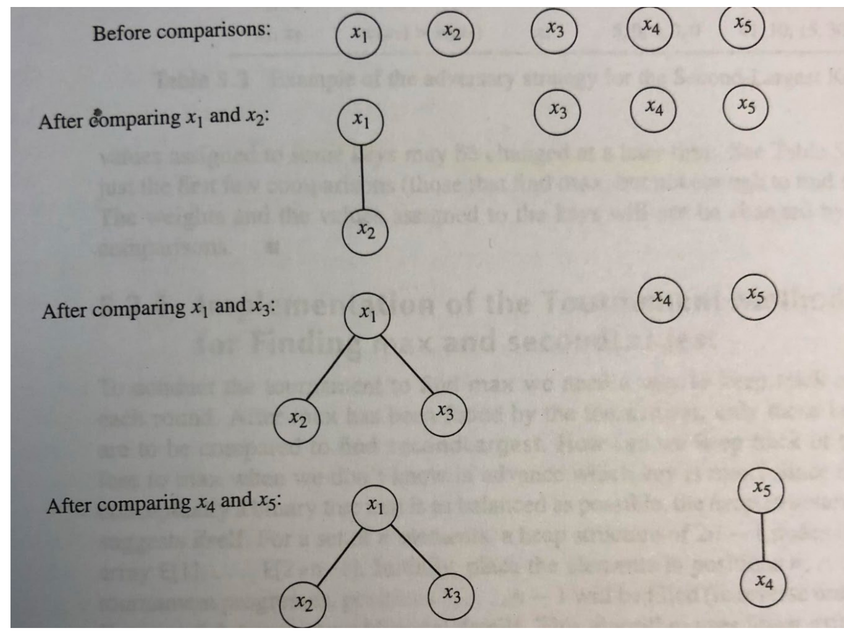
Force max to compare ⌈lgn⌉

A key has lost iff its weight is zero

The sum of he weights is always n

When it stops, only one key can have nonzero weight

(otherwise, there are two keys that never lost)



Finding the second-largest key: adversary in action

Comparands	Weights	Winner	New weights	Keys
x_1, x_2	$w(x_1) = w(x_2)$	x_1	2, 0, 1, 1, 1	20, 10, *, *, *
x_1, x_3	$w(x_1) > w(x_3)$	x_1	3, 0, 0, 1, 1	20, 10, 15, *, *
x_5, x_4	$w(x_5) = w(x_4)$	x_5	3, 0, 0, 0, 2	20, 10, 15, 30, 40
x_1, x_5	$w(x_1) > w(x_5)$	x_1	5, 0, 0, 0, 0	41, 10, 15, 30, 40

Finding the median

Problem: finding the median when n is odd, i.e., (n+1)/2-th element.

Naïve solution: (1) sort and (2) select the (n+1)/2-th element.

Complexity of the naïve solution: $O(n \lg n)$

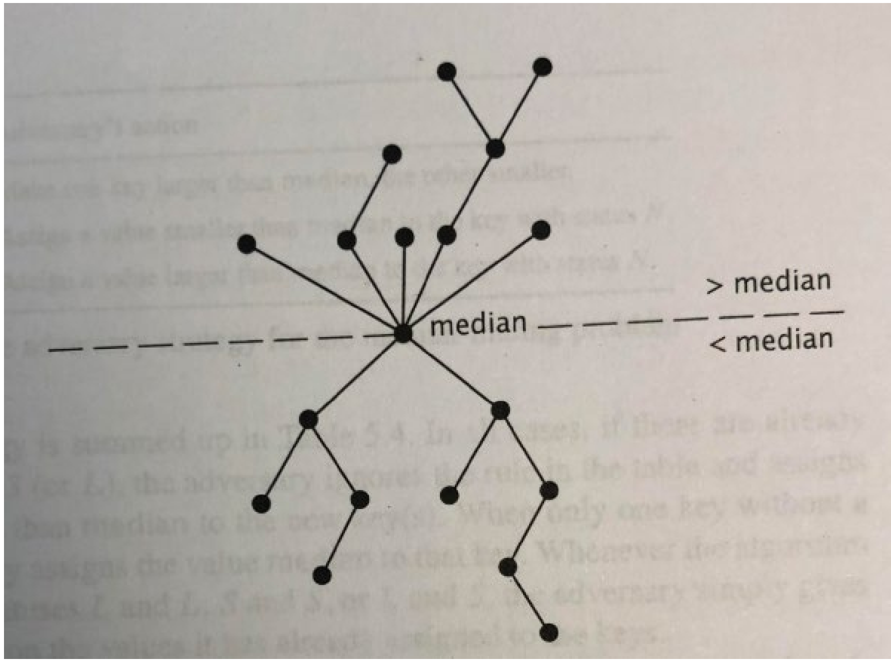
Lower bound: $3n/2-3/2$

(best lower bound so far: slightly > 2, but still has a gap)

Finding the median: adversary

Adversary: "floating" median

cannot assign values larger (smaller) than the median to more than (n-1)/2 keys.



Crucial comparison for x: if it is the first time where x > y, for y > median, or x < y for some y ≤ median.

Noncrucial: comparisons of x and y, where x > median and y < median

Finding the median: adversary

Adversary: forces the programmer to make noncritical comparisons
n-1 (crucial) + (n-1)/2 non-crucial = 3n/2-3/2

Each operation in the table creates at most one L-key and one S-key until there are (n-1)/2 L-keys or (n-1)/2 S-keys

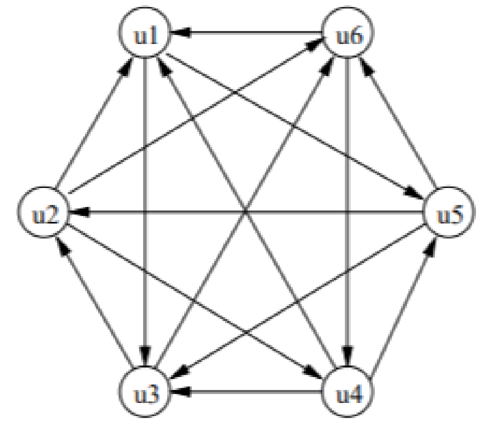
L	Has been assigned a value *L*arger than median.
S	Has been assigned a value *S*maller than median.
N	Has not yet been in a comparison.

Comparands	Adversary's action
N, *N*	Make one key larger than median, the other smaller.
L, *N* or *N*, *L*	Assign a value smaller than median to the key with status *N*.
S, *N* or *N*, *S*	Assign a value larger than median to the key with status *N*.

Kings and Sorted Sequence of Kings

Tournament: a complete directed graph such that for any u and v, either u -> v (u beats v) or v -> u, but not both.

King: u is a king if all other directly or indirectly through a third player in a tournament.



u_4 and u5 are kings

Sorted sequence of kings (Wu 2000): an ordered list of players in a tournament $u_1, u_2, \ldots, u_n$ such that

$u_i \to u_{i+1}$, and

u_i is a king in the sub-tounament induced by $\{u_j: i \le j \le n\}$.

$u_2 \to u_4 \to u_1 \to u_5 \to u_3 \to u_6$

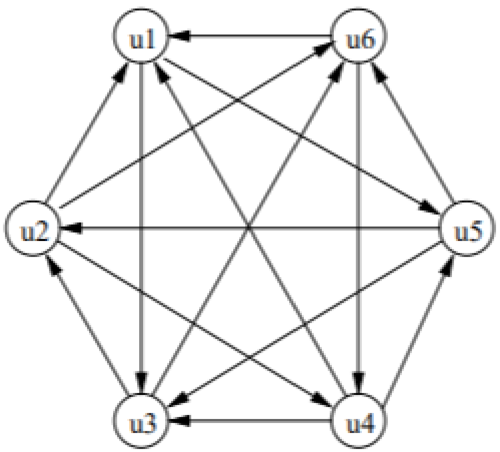
$u_2 \to u_6 \to u_4 \to u_1 \to u_5 \to u_3$

Kings and Sorted Sequence of Kings: adversary

King is legitimate: includes players with the maximum number of wins.

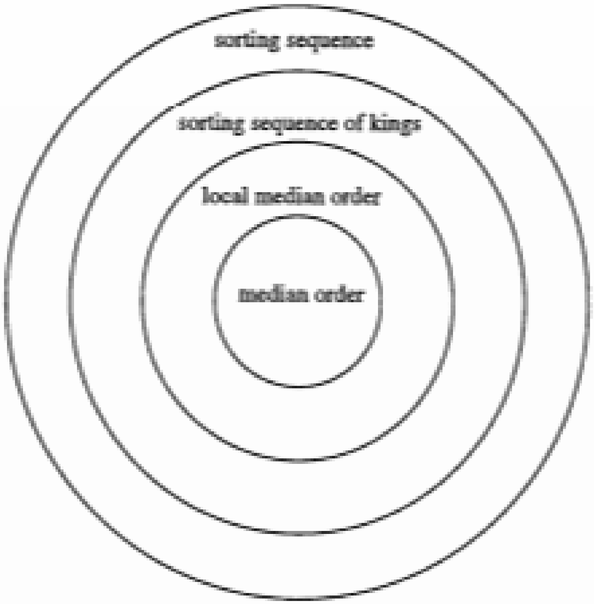
King (Sheng, Shen, and Wu 2003) (open problem): Ω ($n^{4/3}$) and $O(n^{3/2})$

Sorted sequence of kings (Sheng, Shen and Wu 2003): $\Theta(n^{3/2})$



Tournament ranking

Upset: $i < j$, but u_j beats u_i



Median order

- A order with minimum number of upsets
- NP-complete

Local median order

- Sub-tournament $N(i, j)$: $u_i, u_{i+1}, \ldots, u_j$
- # wins by u_i is greater than # loses in $N(i,j)$
- # loses by u_j is greater than # wins in $N(i,j)$

Nested relationships (Wu 2000)

- Median order
- Local median order
- Sorted sequence of kings
- Sorted sequence