Searching in Trees

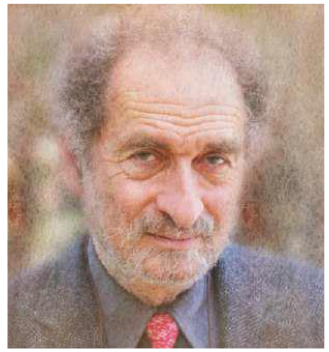
Gerth Stølting Brodal

MADALGO
CENTER FOR MASSIVE DATA ALGORITHMICS

Aarhus University

Workshop on *The Art of Data Structures* in honor of Prof. Athanasios Tsakalidis, Patras, Greece, November 7, 2017

Kurt Mehlhorn
PhD, Cornell 1974



Athanasios K. Tsakalidis
PhD, Saarbrücken 1983

2 papers

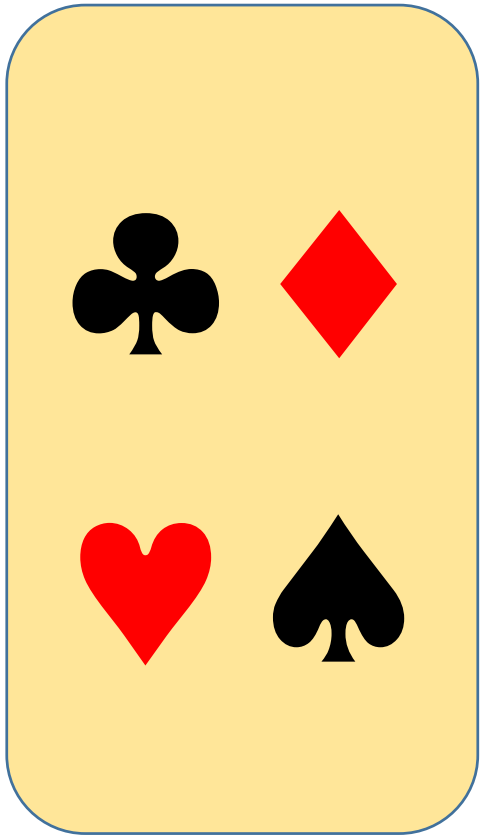
Erik Meineche Schmidt
PhD, Cornell 1977

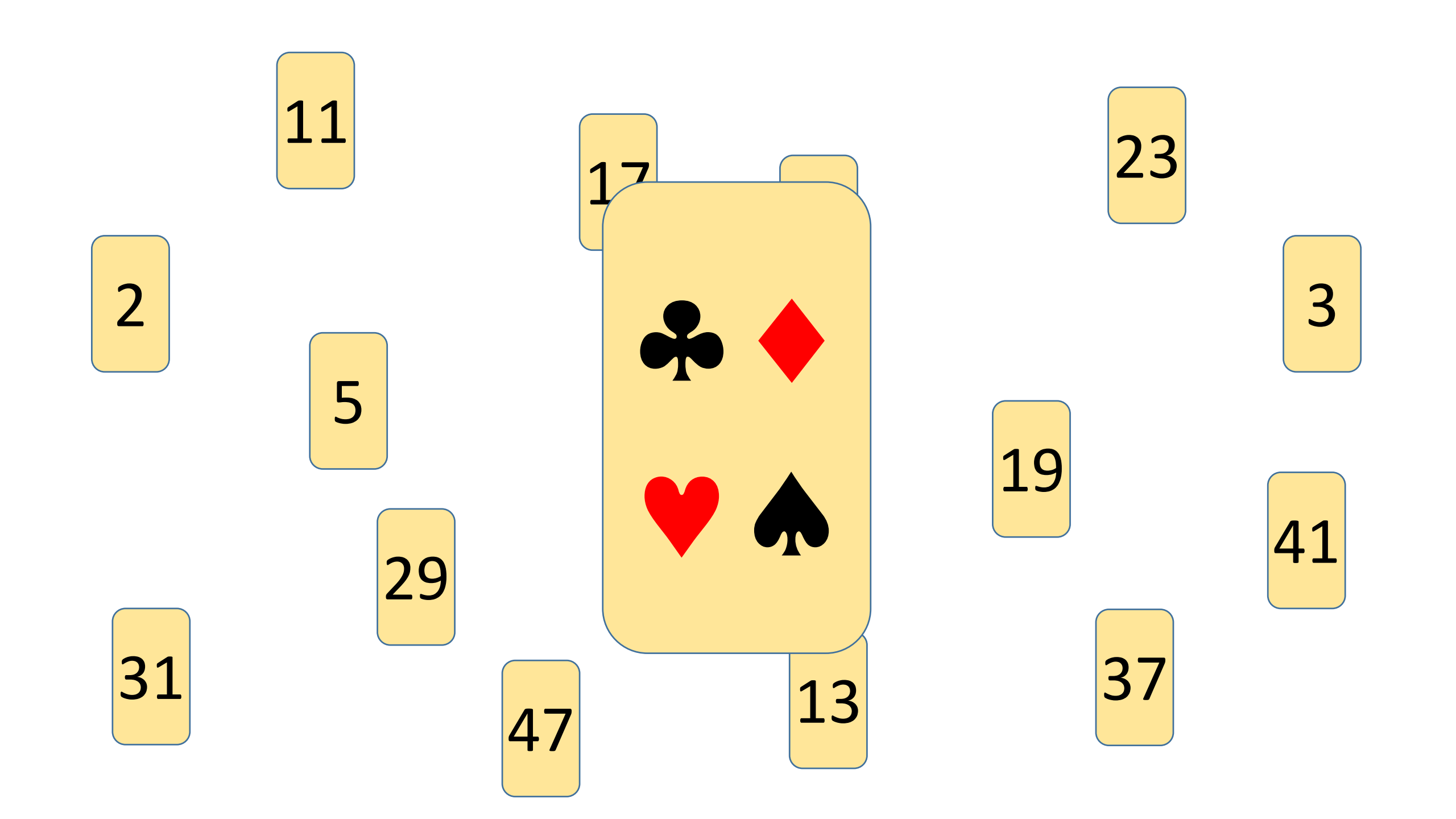
Gerth Stølting Brodal
PhD, Aarhus 1997

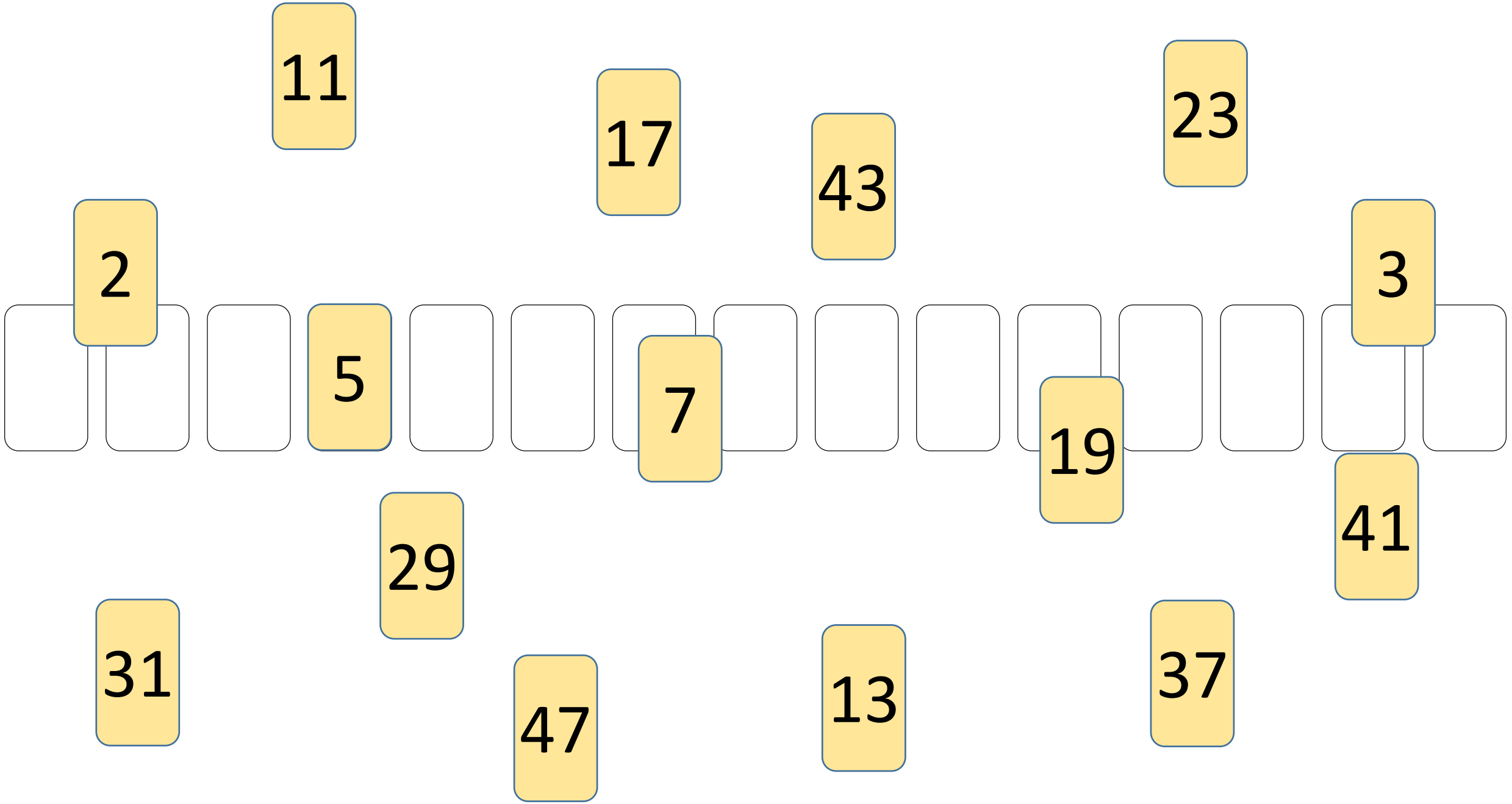
Konstantinos Tsakalidis
PhD, Aarhus 2011

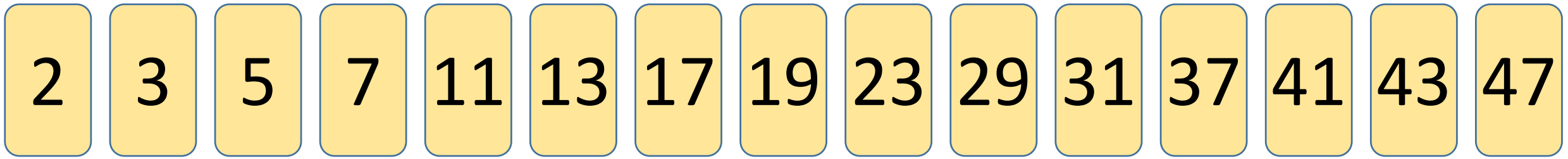
Konstantinos Tsichlas
PhD, Patras 2003

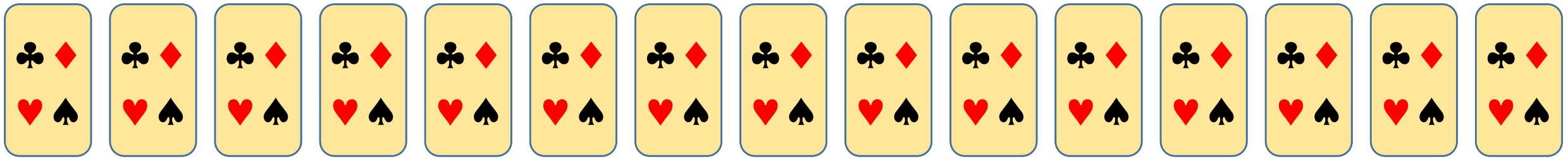
Konstantinos Mampentzidis
Diploma, Thessaloniki 2013
PhD, Aarhus exp. 2019



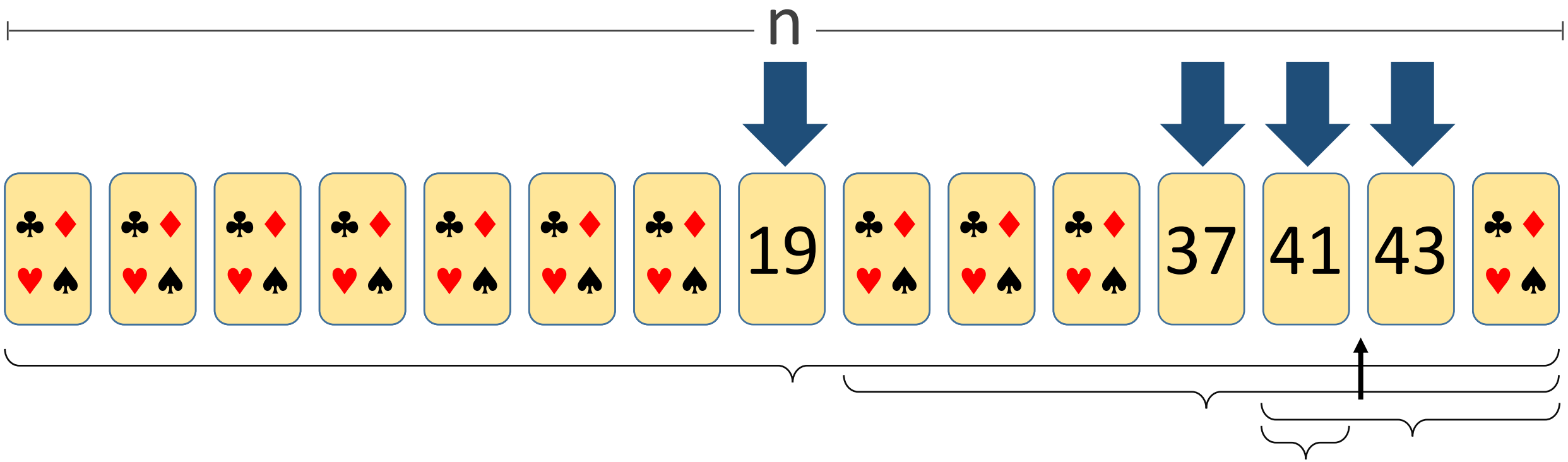

11
17
23
2
3
5
19
41
29
31
37
47
13


11
17
43
23
2
3
5
7
19
41
29
31
47
13
37


2 3 5 7 11 13 17 19 23 29 31 37 41 43 47

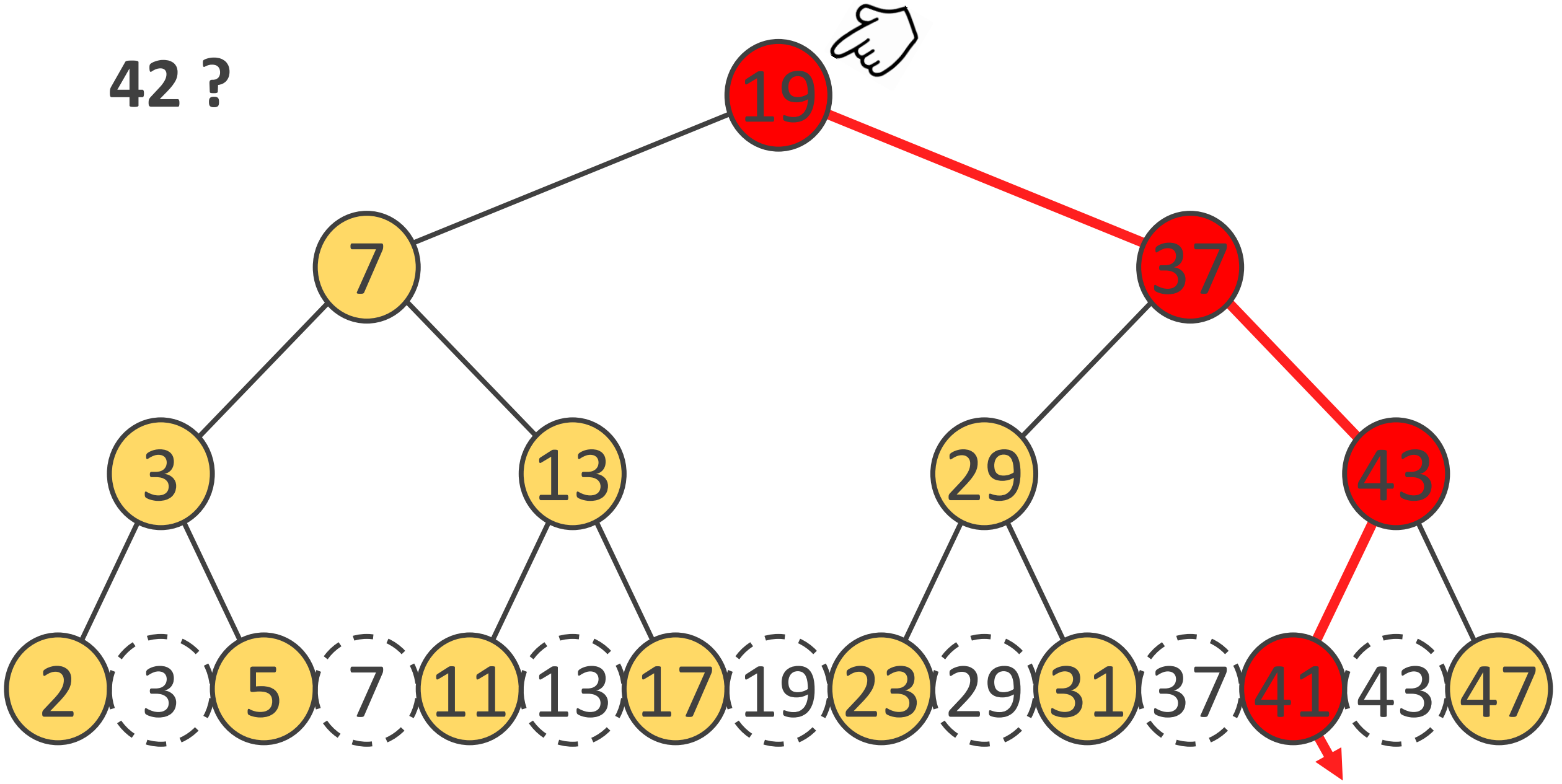


42 ?

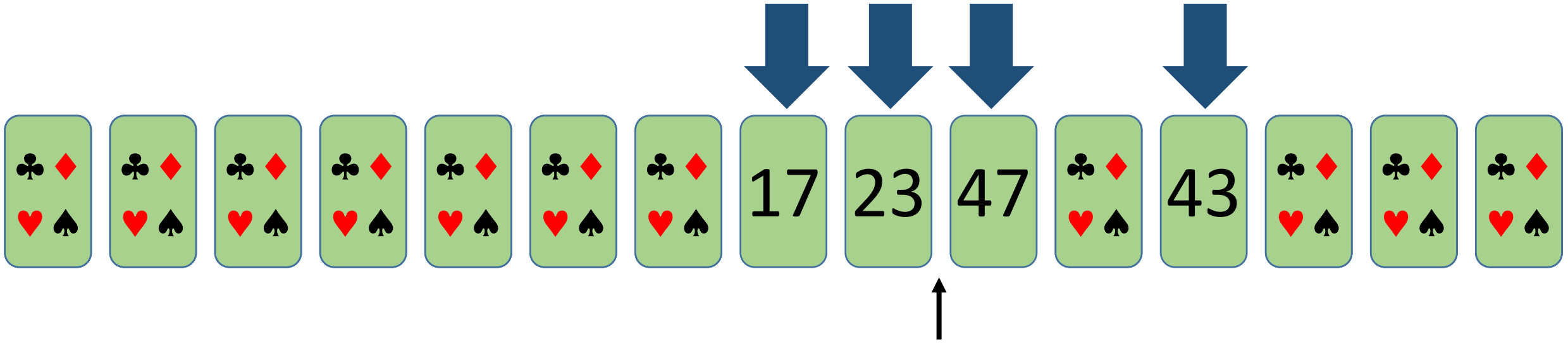


Binary search $1 + \lfloor \log_2 n \rfloor$ comparisons

42 ?

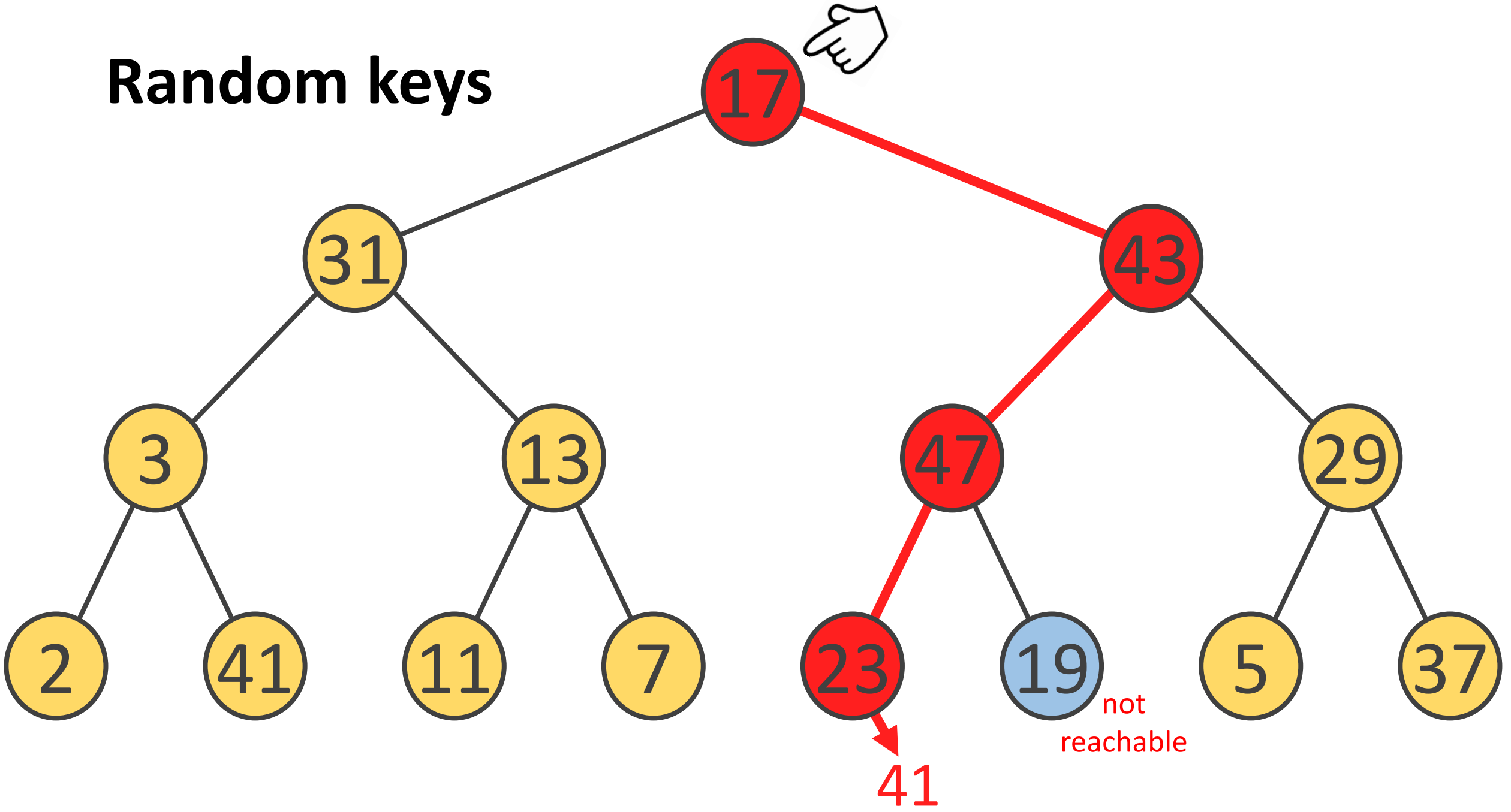


Binary search 41 ?

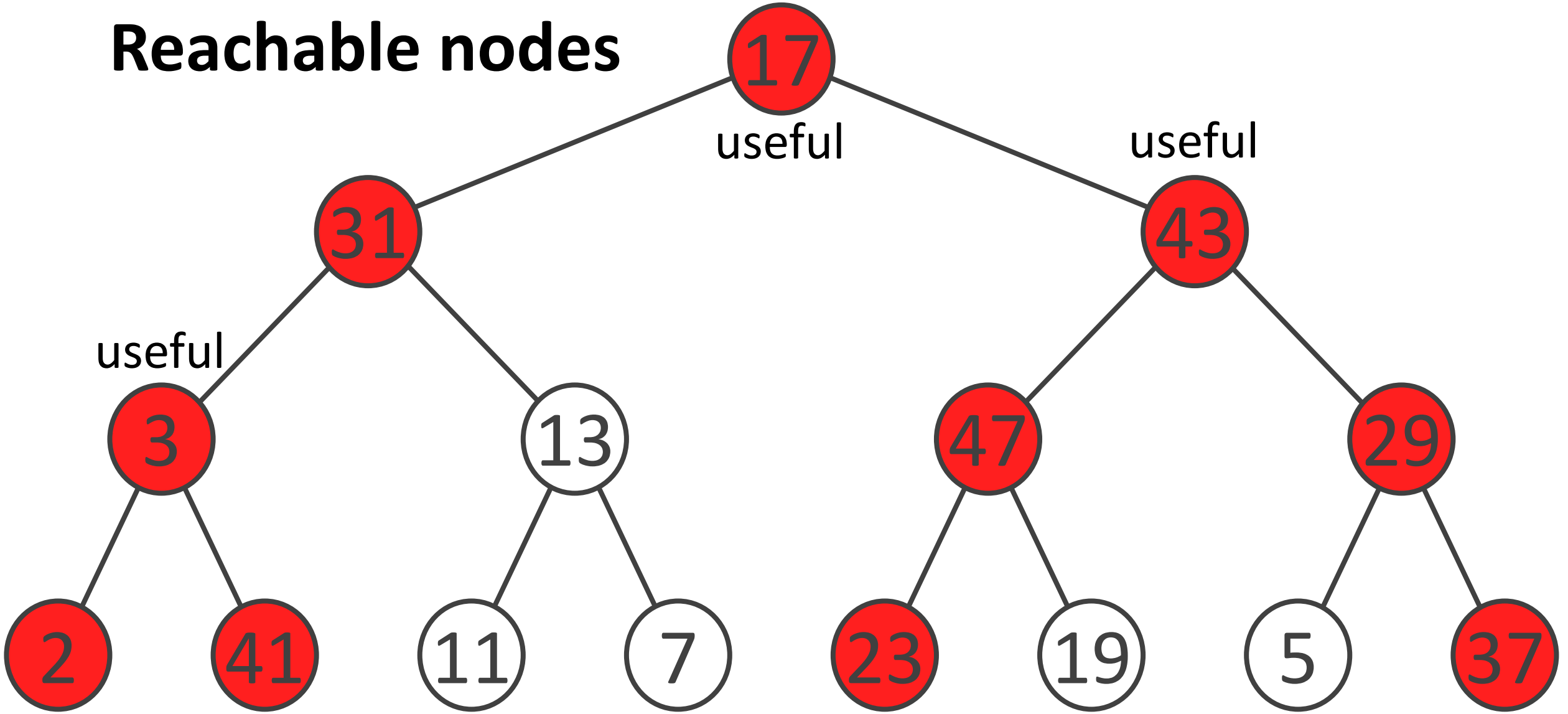


Forgot to sort...

Random keys



Reachable nodes



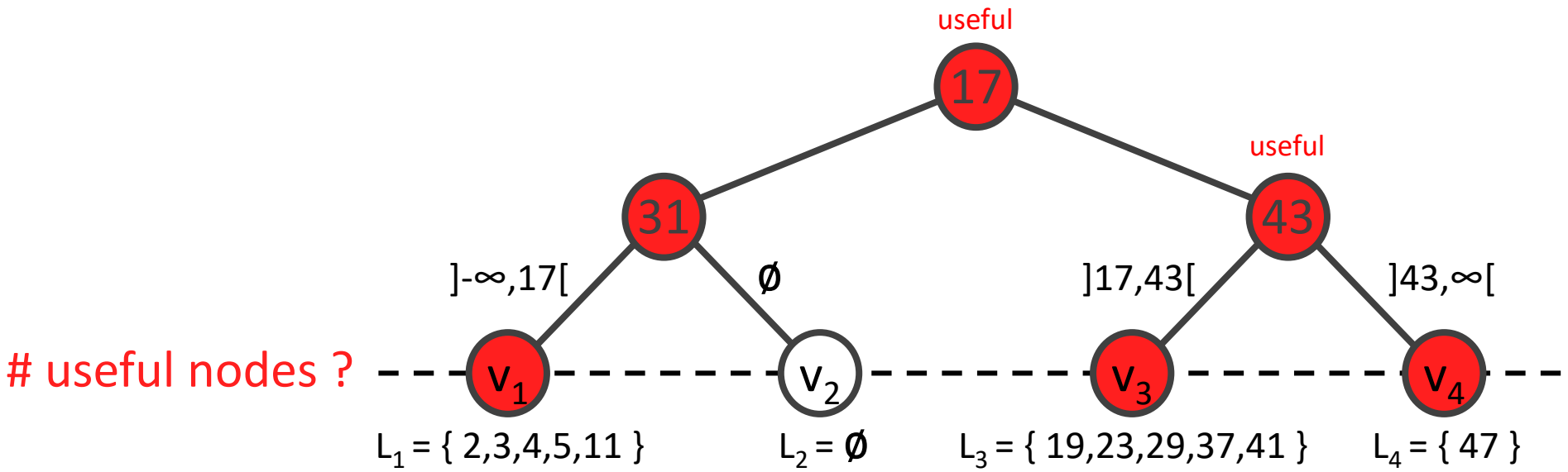
Reachable nodes – analysis

$\Pr[\ v_i$ useful $\ ] = |L_i| \ / \ \Sigma_j |L_j|$

E[# useful nodes at level] = $\Sigma_i(\ |L_i| \ / \ \Sigma_j |L_j| \ ) = 1$

E[# useful nodes in tree] = height - 1

E[# reachable nodes in tree] ≤ height^2 = **O(log^2 n)** □

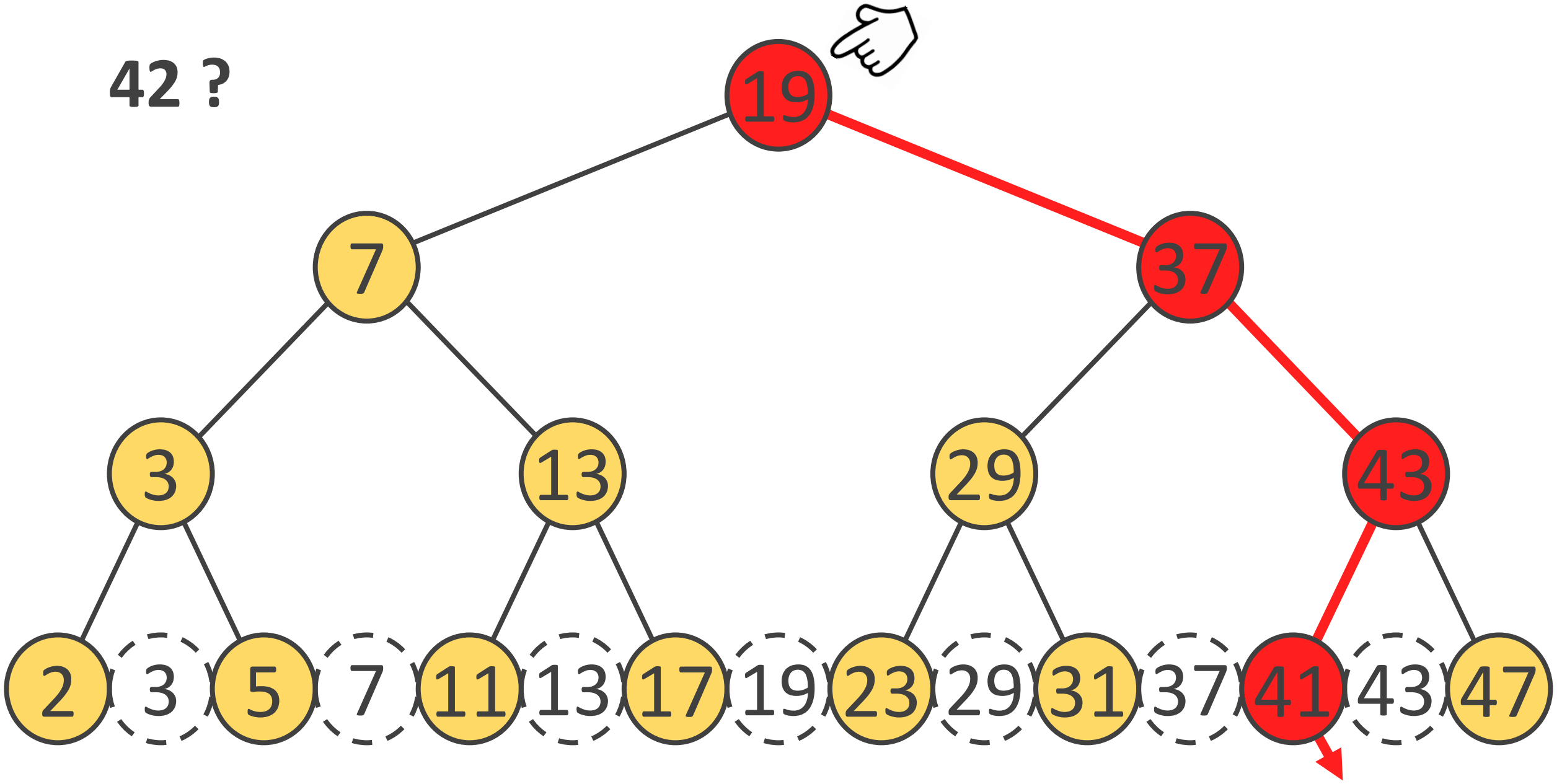


Fun-Sort

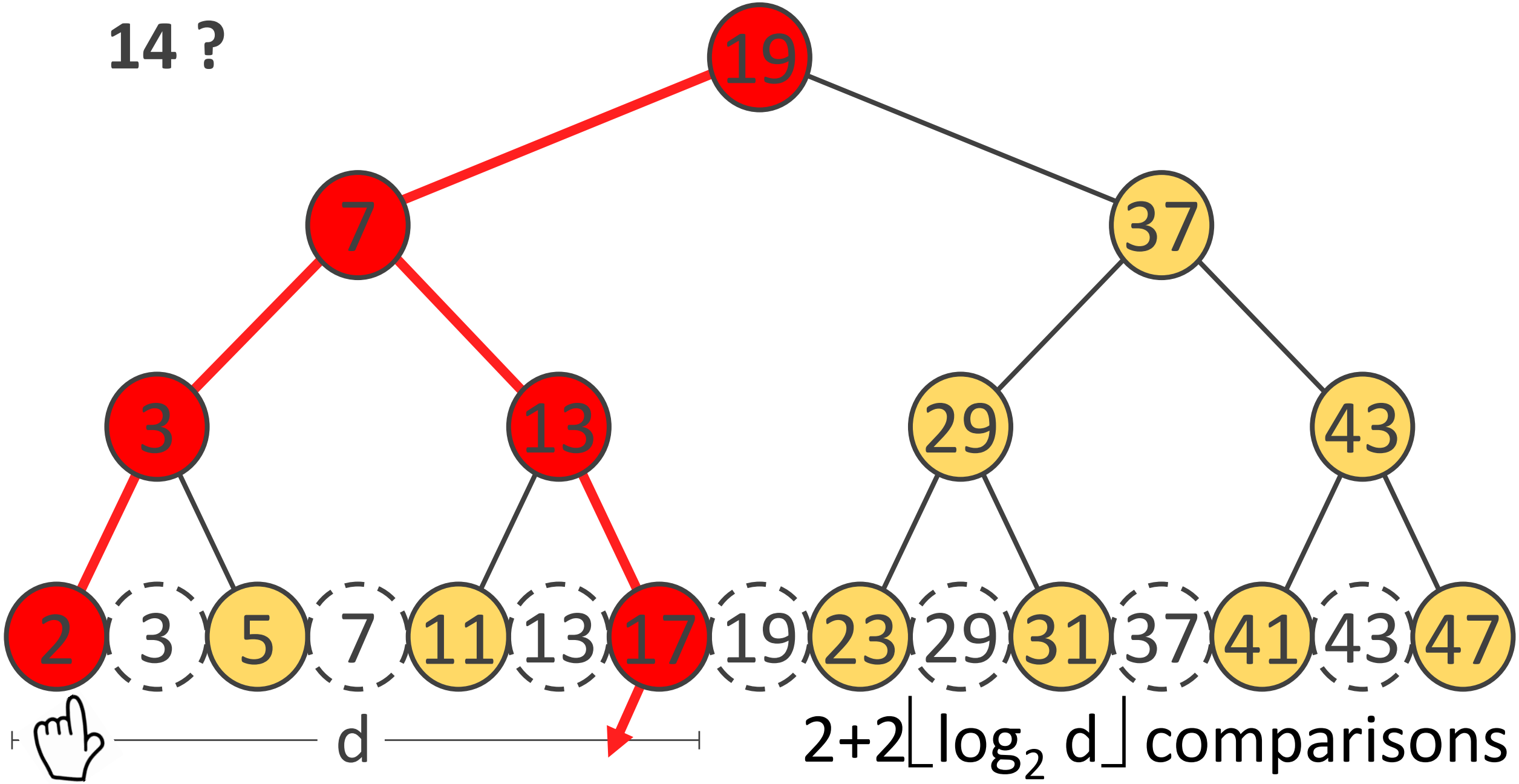
Uses repeated searches in an unsorted array to sort the array in $\Theta(n^2 \log n)$ time.

[Biedl, Chan, Demaine, Fleischer, Golin, King, Munro 2001]

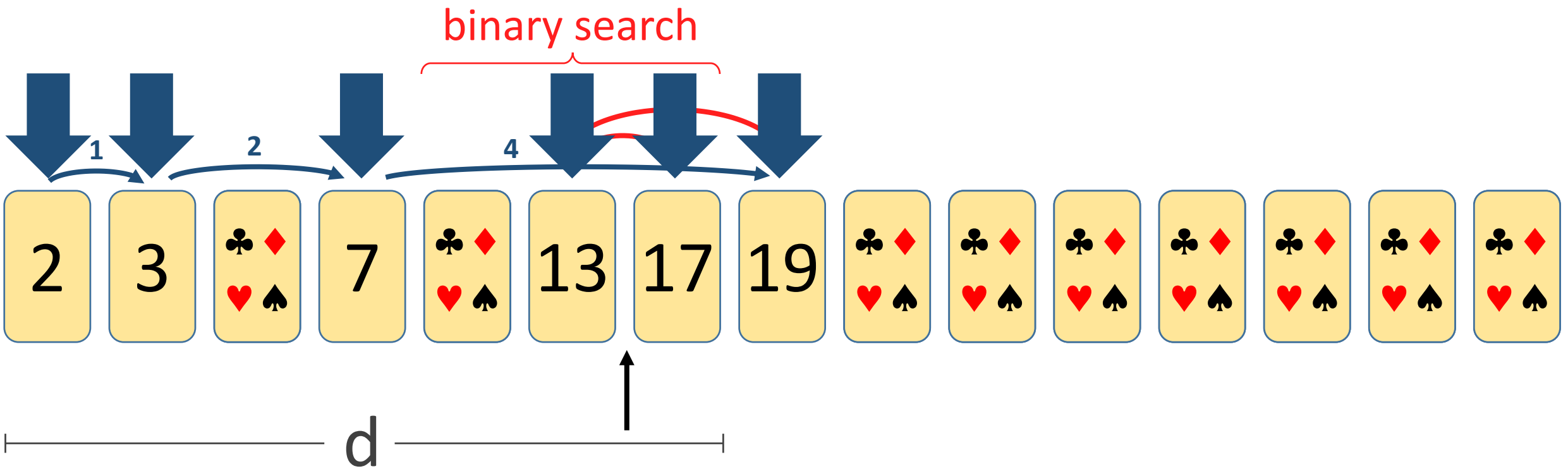
42 ?



14 ?

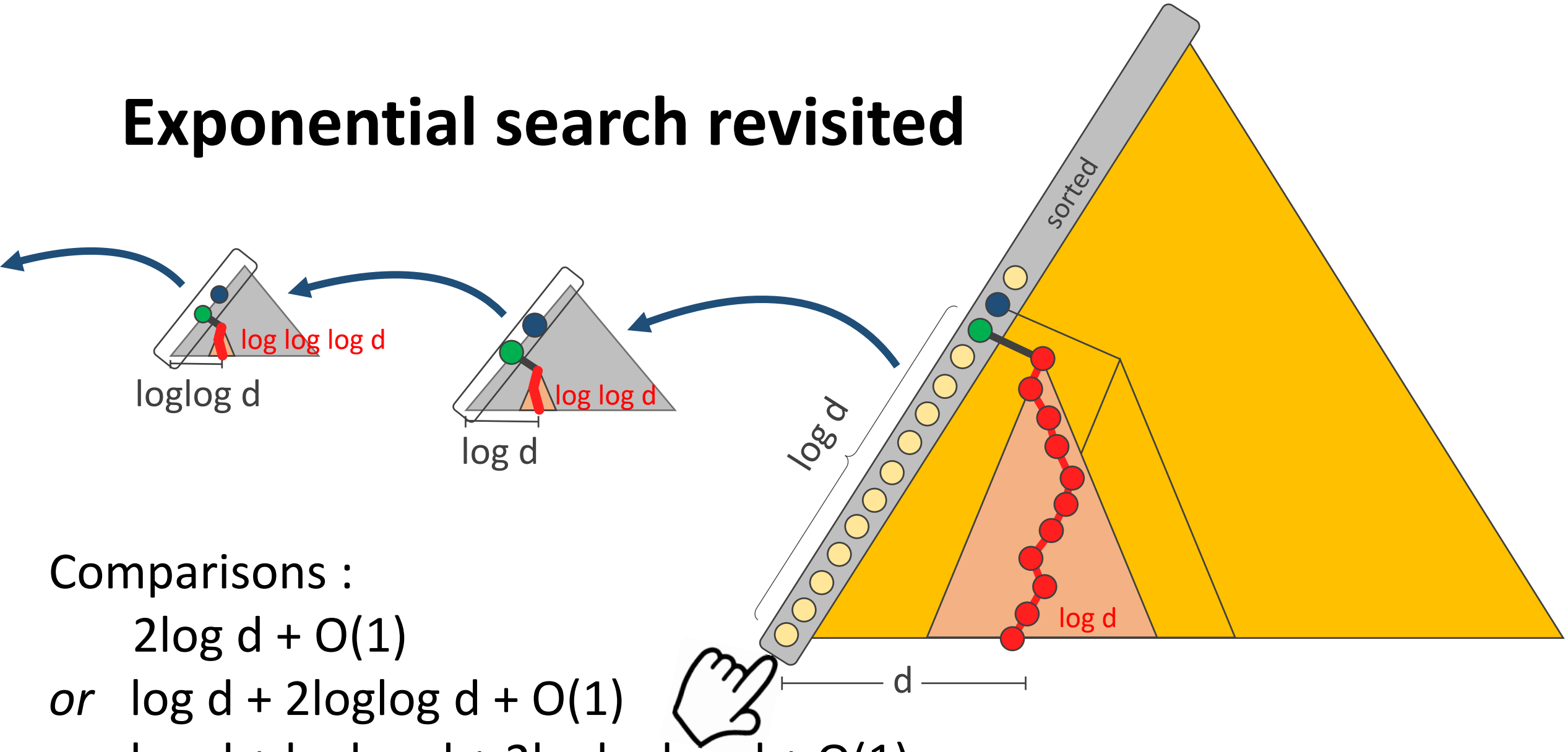

19
7
37
3
13
29
43
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
d
2+2⌊log2 d⌋ comparisons

14 ?



Exponential search $2+2\lfloor \log_2 d \rfloor$ comparisons

Exponential search revisited



Comparisons :

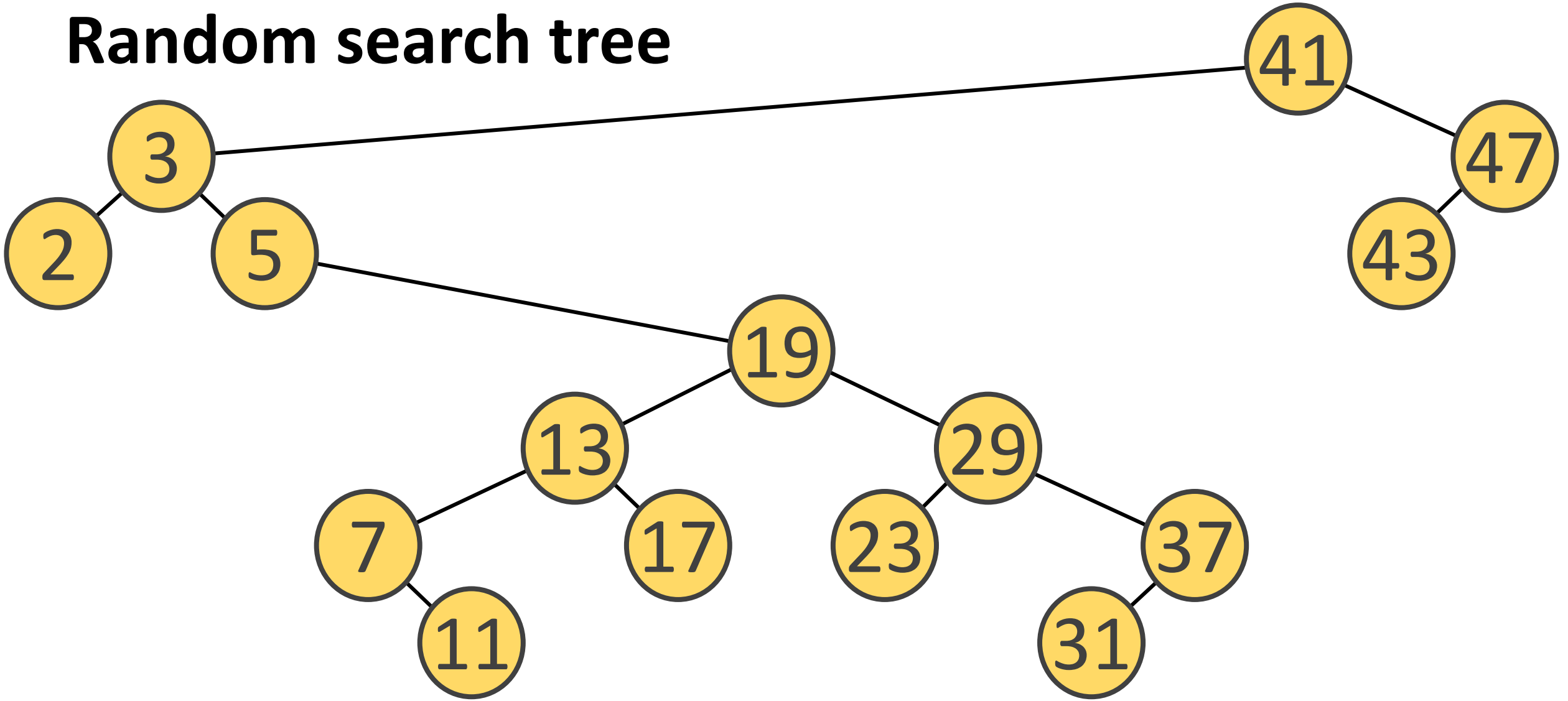
2log d + O(1)

or log d + 2loglog d + O(1)

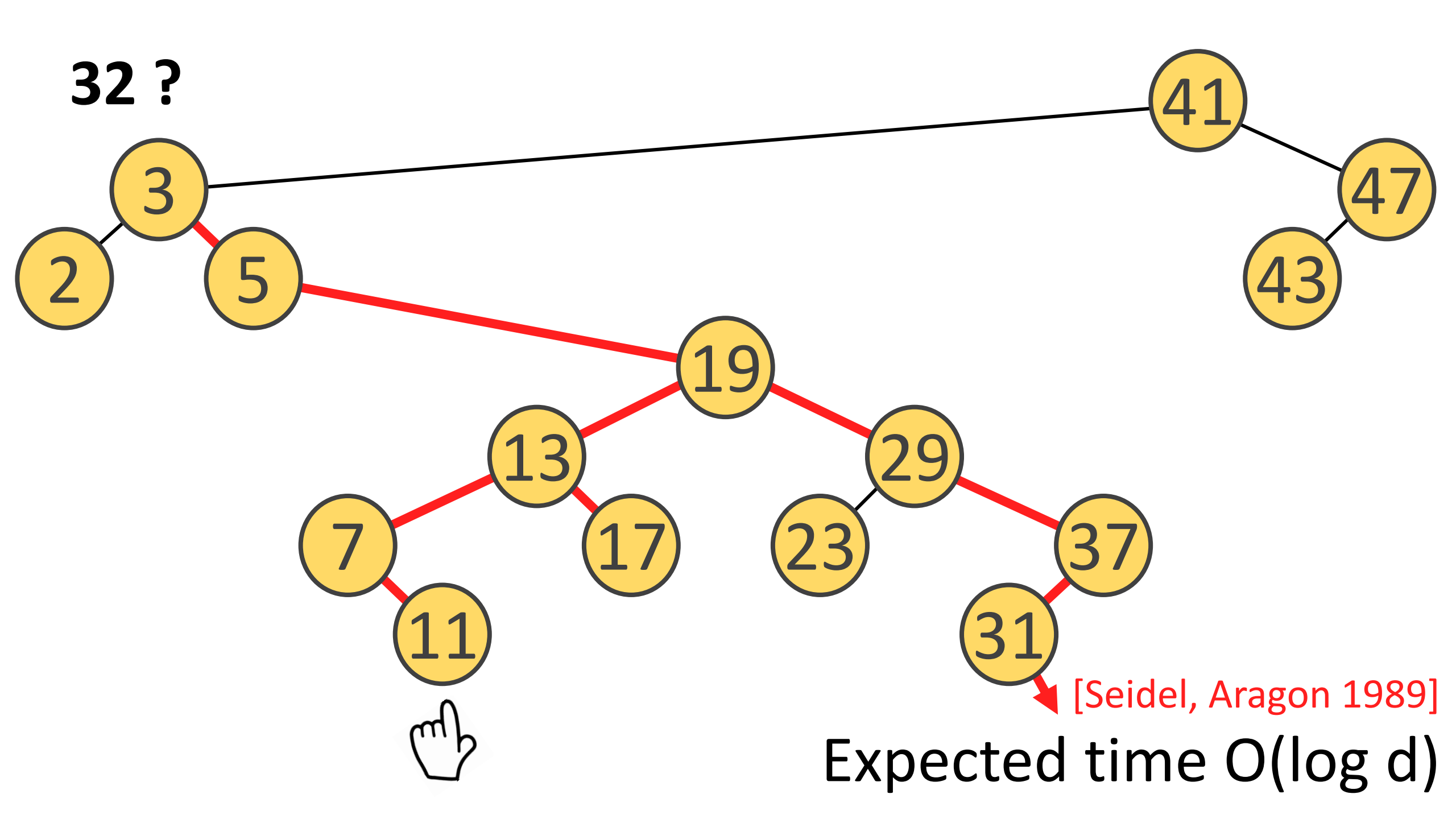
or log d + loglog d + 2logloglog d + O(1)

or $\sum_{i=1..\log^* d} \log^{(i)} d + O(\log^* d)$ [Bentley, Yao 1976]

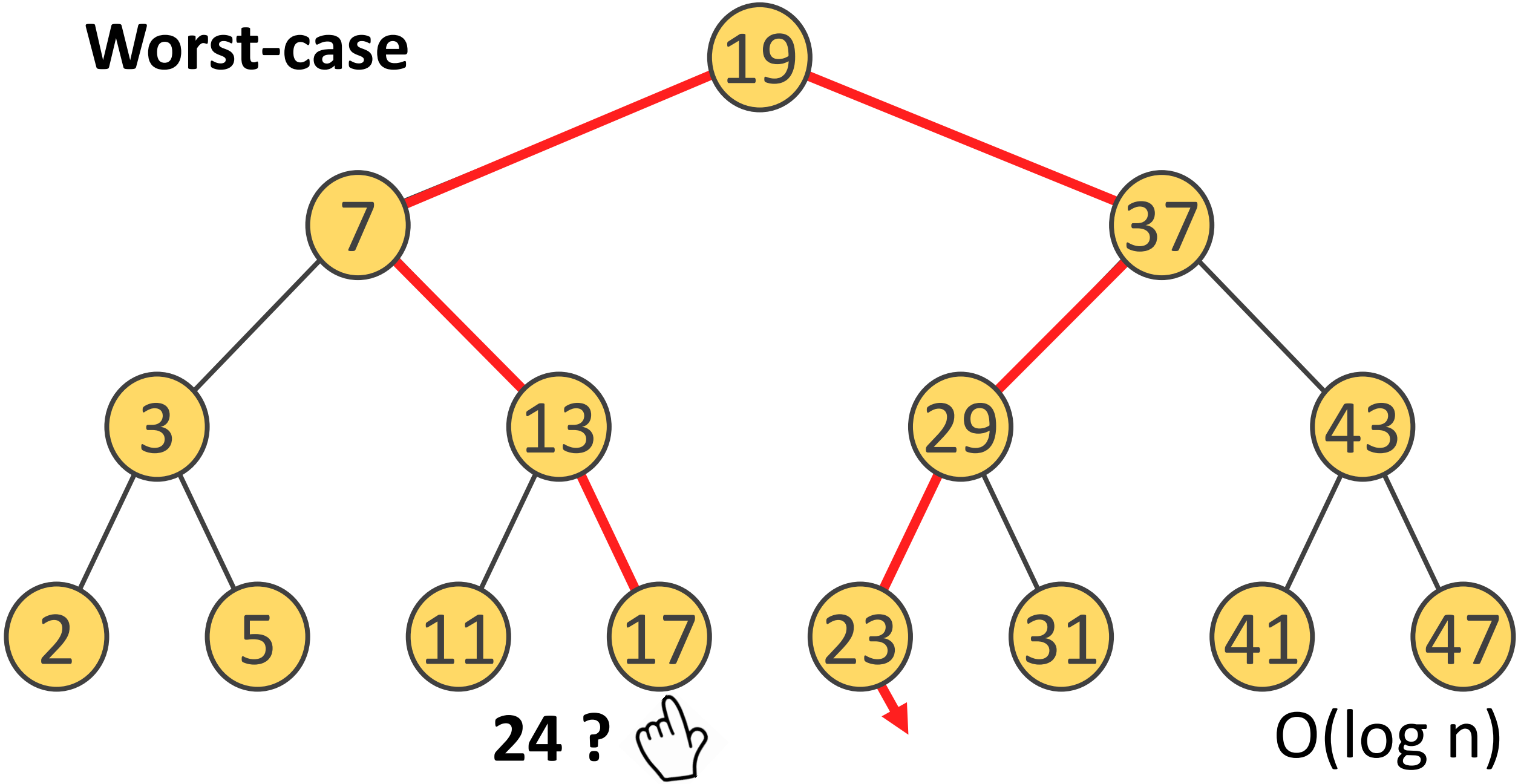
Random search tree



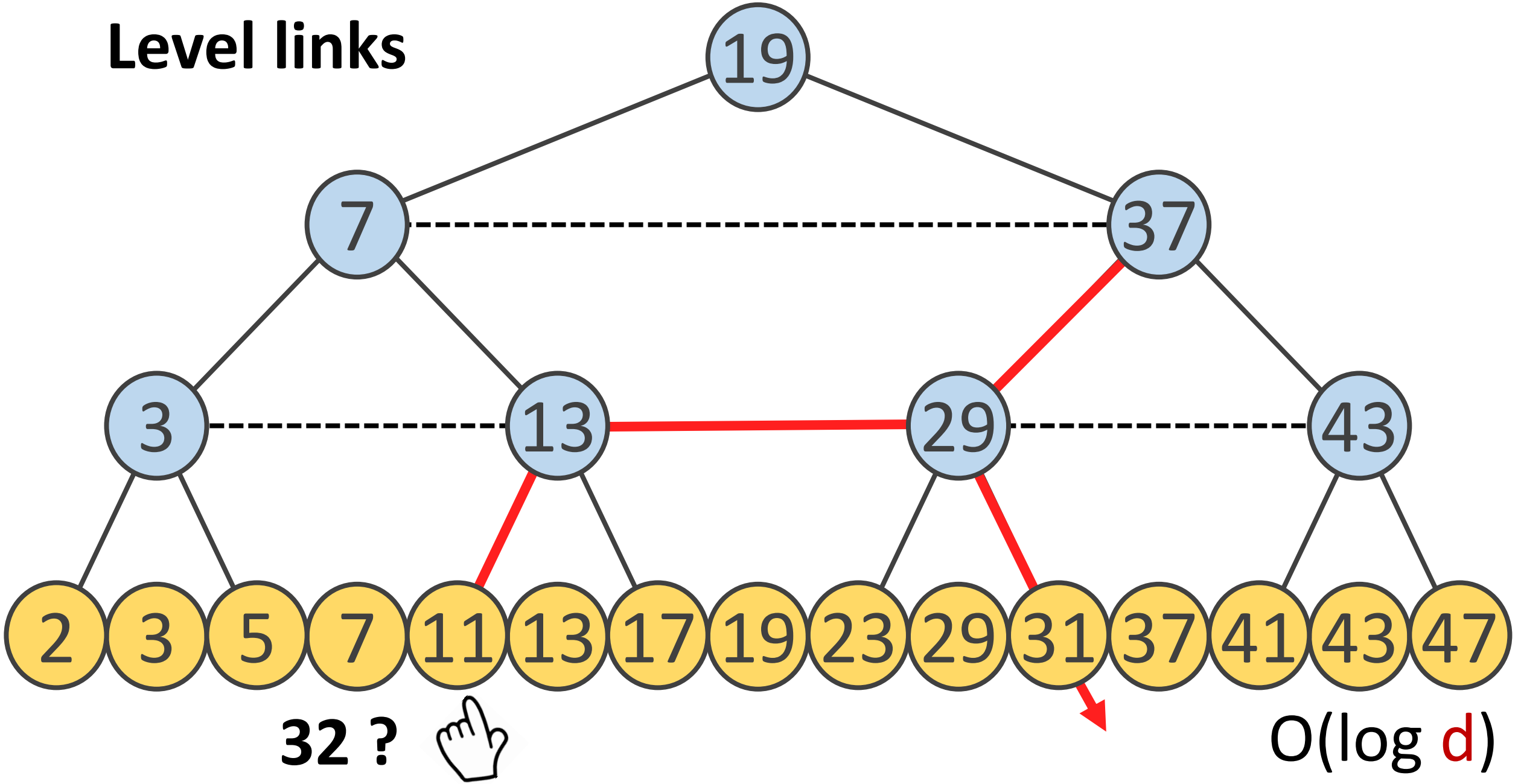
Expected depth O(log n)


32 ?
41
3
47
2
5
43
19
13
29
7
17
23
37
11
31
[Seidel, Aragon 1989]
Expected time O(log d)

Worst-case



Level links

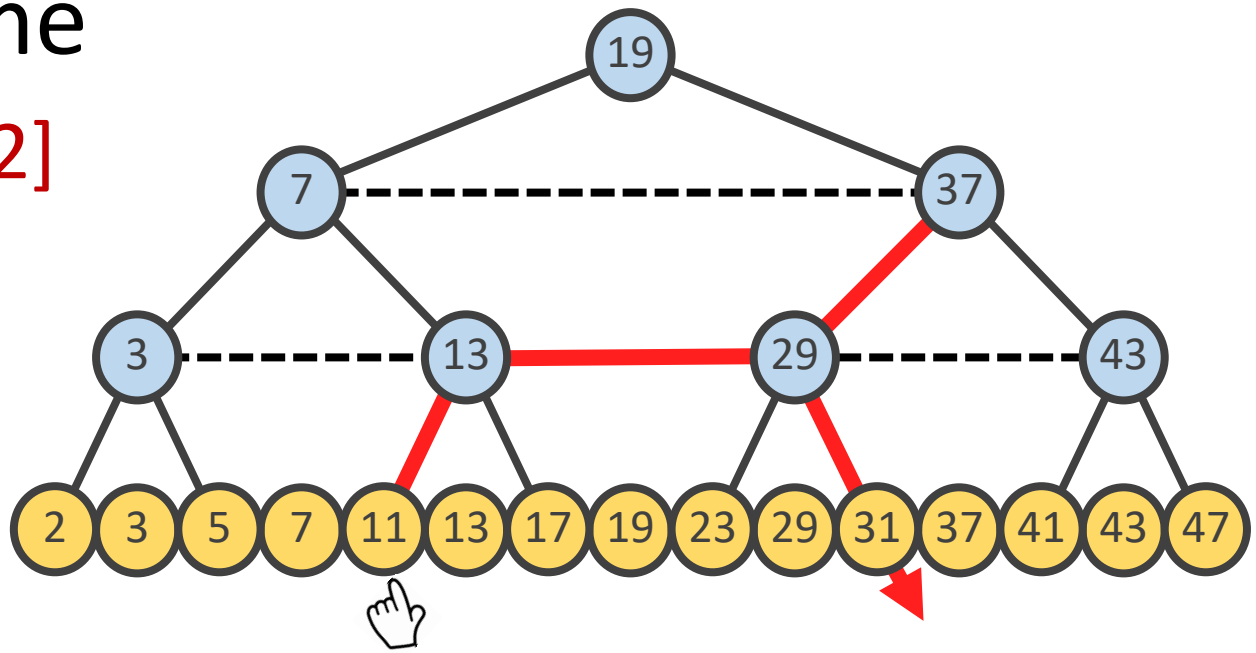


Dynamic finger search trees

Level-linked (2,4)-trees support

- Finger search in worst-case O(log d) time
- Insert/delete in amortized O(1) time but worst-case O(log n) time

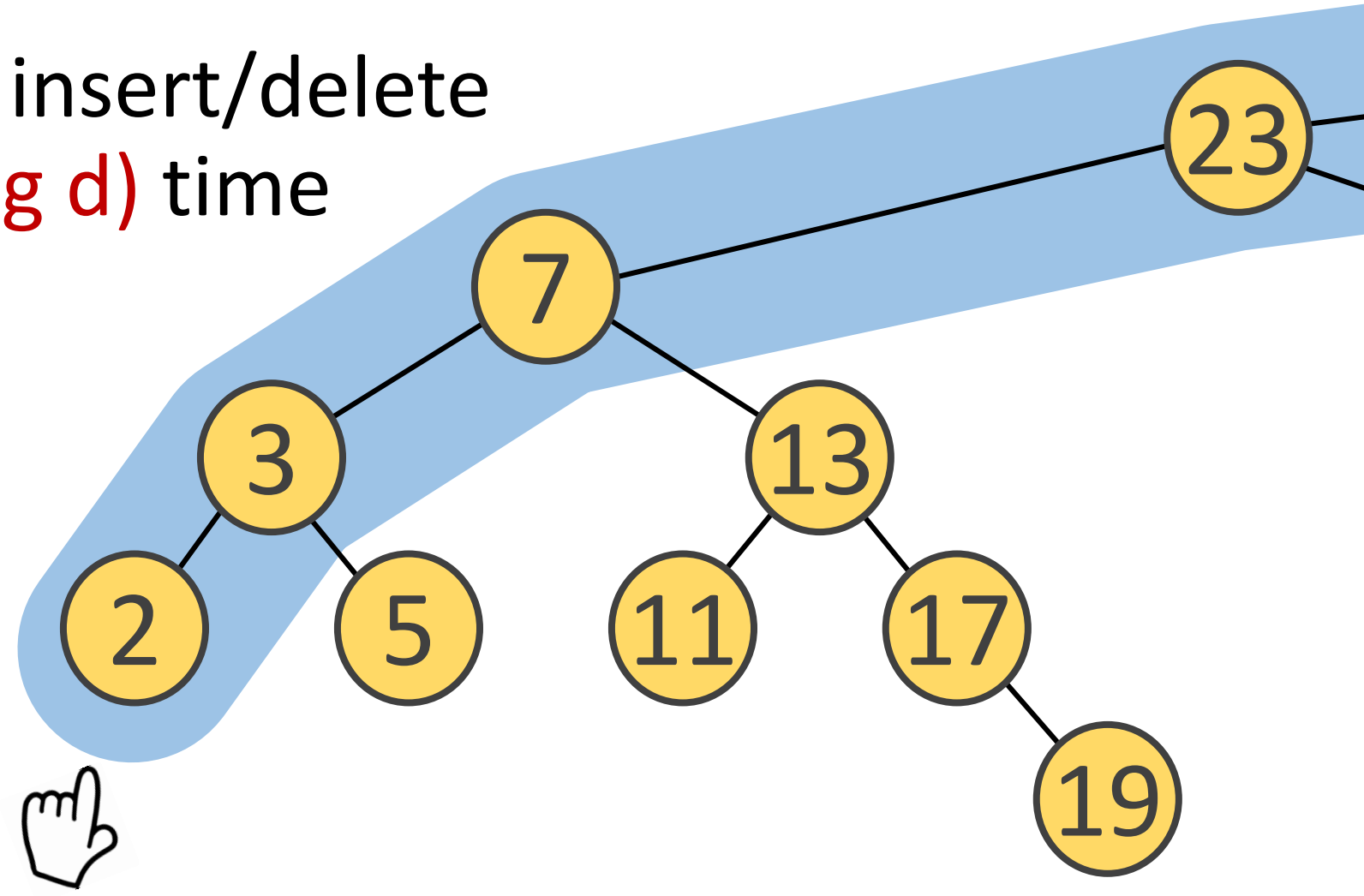
[Huddleston and Mehlhorn 1982]



AVL-trees with one finger

Finger search and insert/delete
in worst-case O(log d) time

[Taskalidis 1984]

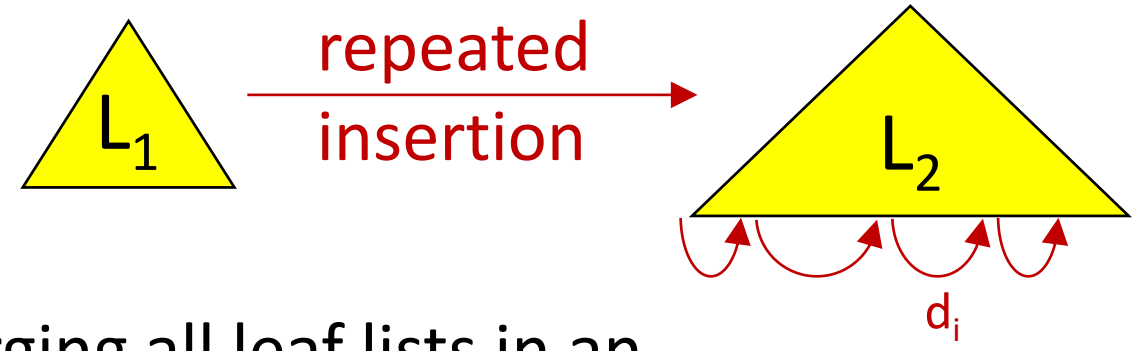


Dynamic finger search

		Search	Insert/Delete
Root finger	Red-black, AVL, 2-4-trees, ... Levcopolous, Overmars 1988	O(log n)	O(log n) O(1)
O(1)	Guibas et al. 1977, Tsakalidis 1984, ...	O(log **d**)	O(1)
Arbitrary	Huddleston and Mehlhorn 1982 (Level-linked (2,4)-trees)	O(log **d**)	O(log n) O(1) am.
	Treaps, Randomized Skip lists	O(log **d**) exp.	O(1) exp.
	Brodal, Lagogiannis, Makris, Tsakalidis, Tsichlas 2002	O(log **d**)	O(1)
RAM, Arbitrary	Dietz, Raman 1994 (comparison)	O(log **d**)	O(1)
	Andersson and Thorup 2000 (non-comparison) Sioutas, Panagis, Theodoridis, Tsakalidis 2005	$O(\sqrt{\log \mathbf{d}/\text{loglog } \mathbf{d}})$	O(1)
	Kaporis, Makris, Sioutas, Tsakalidis, Tsichlas, Zaroliagis 2003 (interpolation search, random distributions)	O(loglog **d**) exp.	O(1) exp.

Application : Binary merging [Huddleston, Mehlhorn 1982]

- Merging sorted lists L_1 and L_2 represented by finger search trees



$$\sum \log(d_i) = O\left(|L_1| \log\left(\frac{|L_2|+|L_1|}{|L_1|}\right)\right)$$

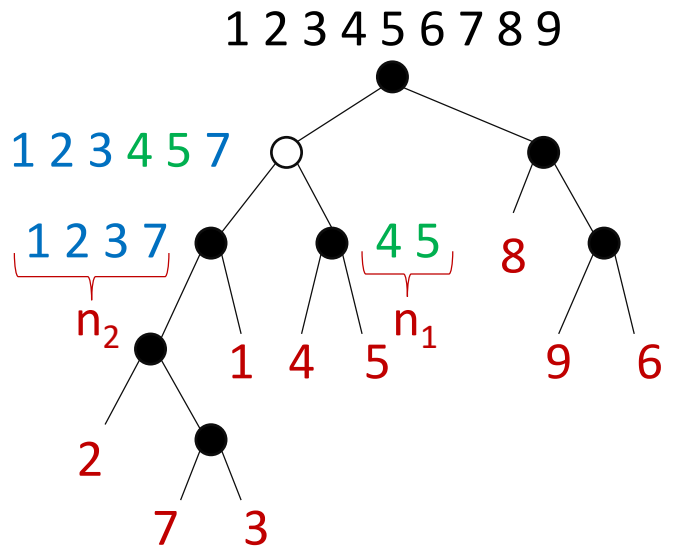
- Merging all leaf lists in an **arbitrary** binary tree $O(n \cdot \log n)$

Proof Induction $O(\log n!)$

$O(\log n_1! + \log n_2! + n_1 \cdot \log((n_1+n_2)/n_1))$

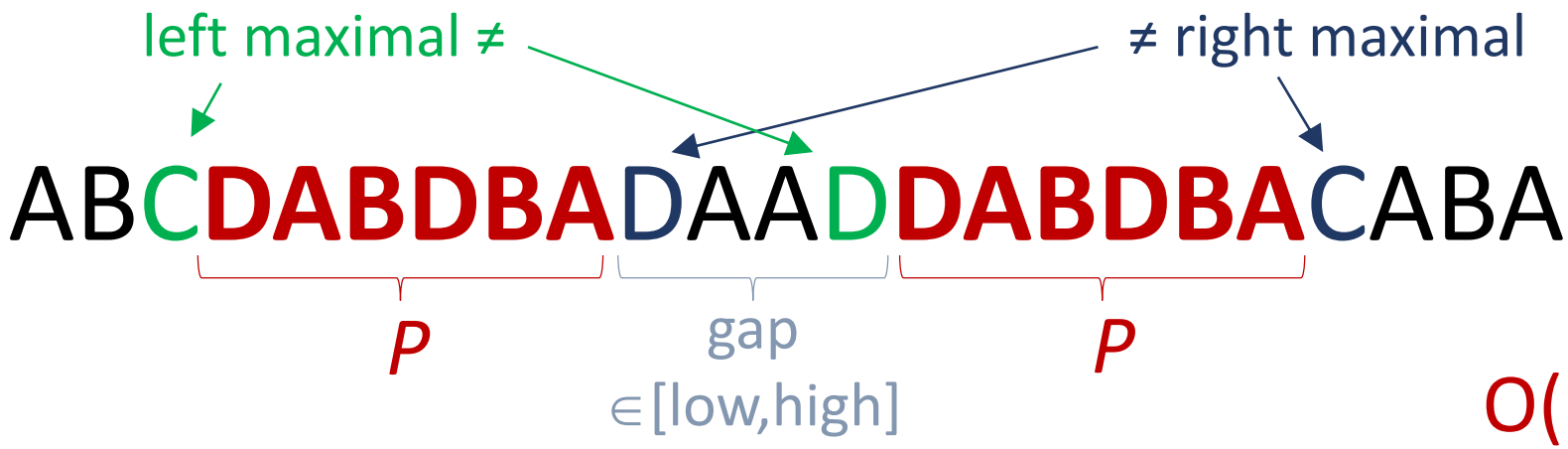
$= O(\log n_1! + \log n_2! + \log \binom{n_1+n_2}{n_1})$

$= O(\log (n_1! \cdot n_2! \cdot \binom{n_1+n_2}{n_1})) = O(\log (n_1+n_2)!)$ □



Application : Maximal pairs with bounded gap

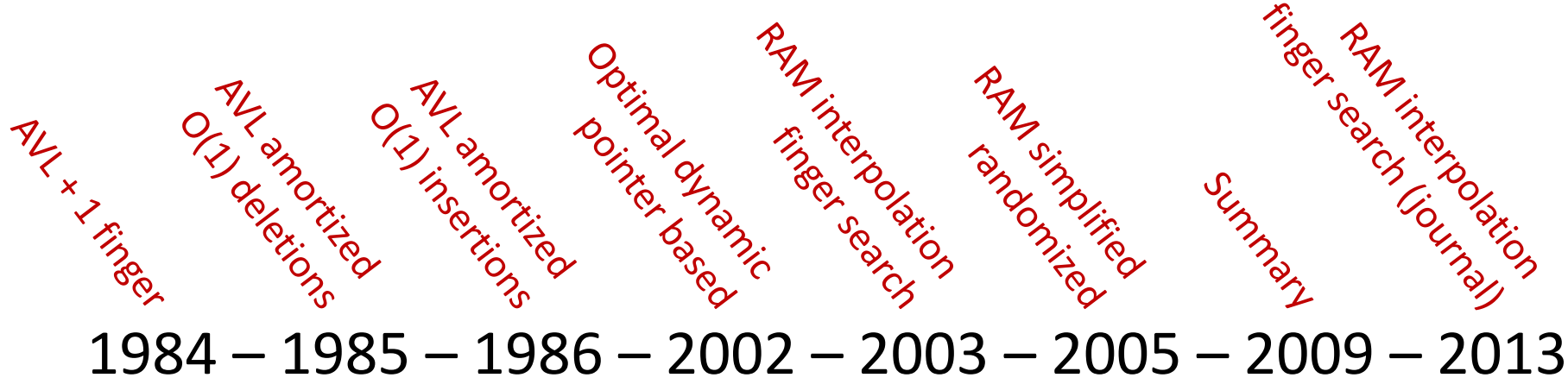
[Brodal, Lyngsø, Pedersen, Stoye 1999]



O(n·log n + k)

- Build suffix tree (ST) & make it binary
- Create leaf lists at each node
- Right-maximal pairs = ST nodes
- Find maximal pairs = finger search at ST nodes

Athanasios K. Tsakalidis & finger search



Thank you