Comparison Based Dictionaries: Fault Tolerance versus I/O Efficiency

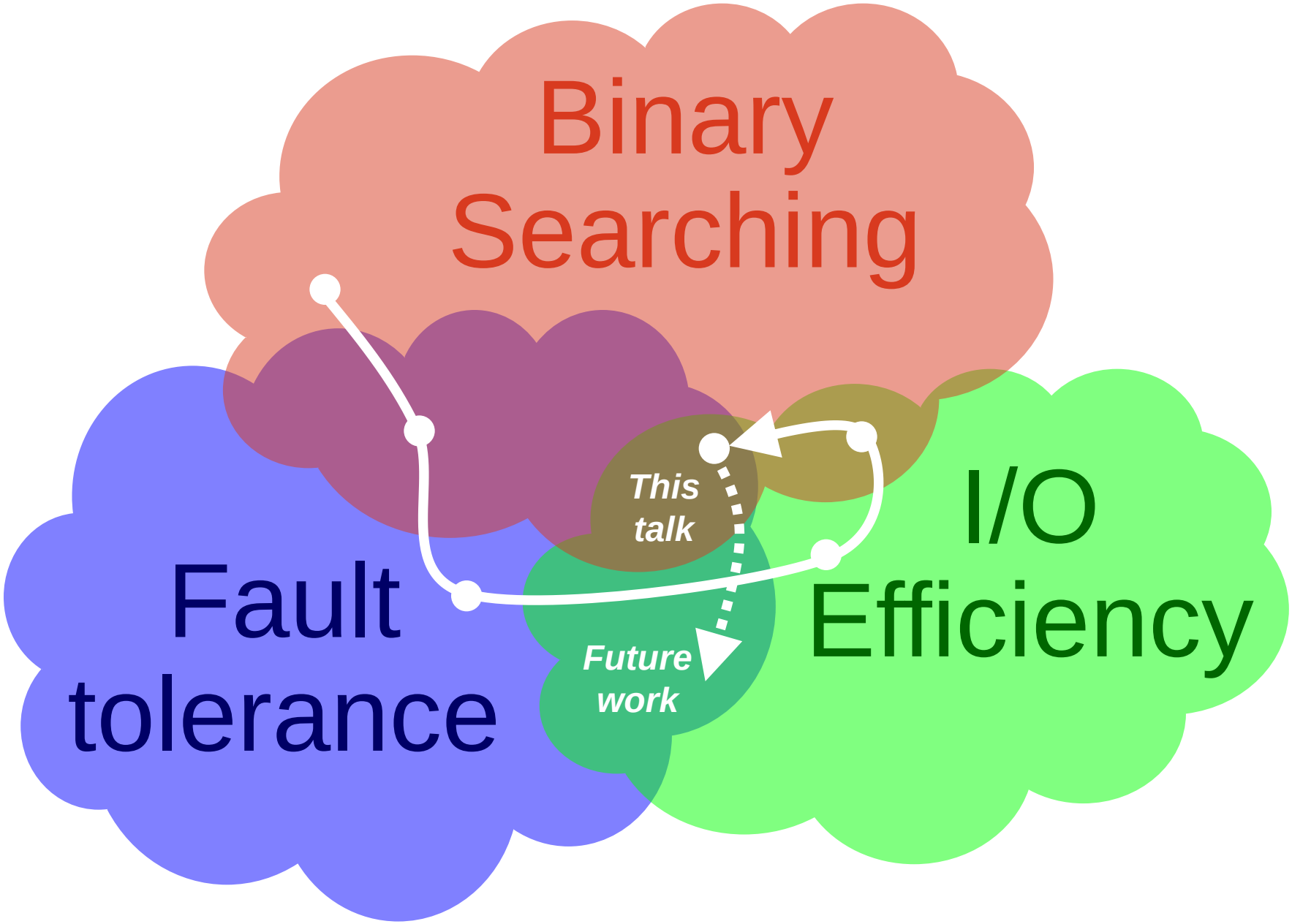
Gerth Stølting Brodal
Allan Grønlund Jørgensen
Thomas Mølhave

University of Aarhus

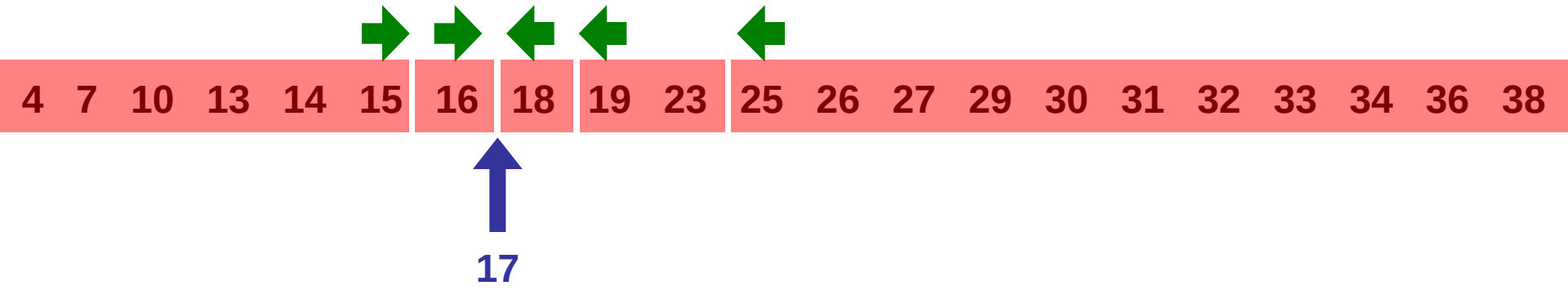
ADS 2007, 3rd Bertinoro Workshop on Algorithms and Data Structures
University Residential Centre of Bertinoro, Italy, September 30-October 5, 2007

MADALGO
CENTER FOR MASSIVE DATA ALGORITHMICS

AARHUS UNIVERSITET

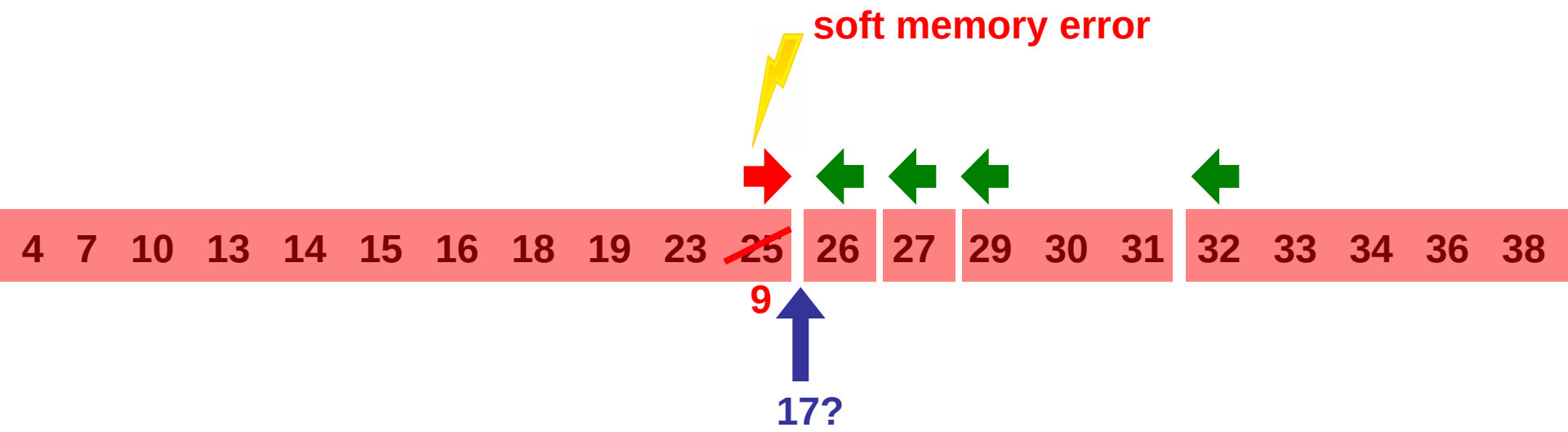

Binary Searching
Fault tolerance
I/O Efficiency
This talk
Future work

Search(17)



$O(\log N)$ **comparisons**

Search(17)


soft memory error
4 7 10 13 14 15 16 18 19 23 25 26 27 29 30 31 32 33 34 36 38
9
17?

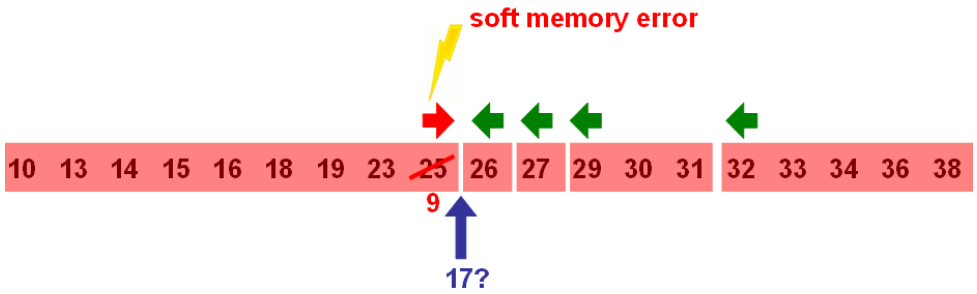
Faulty-Memory RAM Model

Finocchi and Italiano, STOC'04

- Content of memory cells can get corrupted
- Corrupted and uncorrupted content cannot be distinguished
- $O(1)$ safe registers
- **Assumption**: At most δ corruptions

- **Example**: Sorting requires time $\Theta(N \cdot \log N + \delta^2)$

Finocchi, Grandoni, Italiano, ICALP'06



Faulty-Memory RAM: Searching

$\Theta(\log N + \delta)$ **comparisons**

- Lower bound

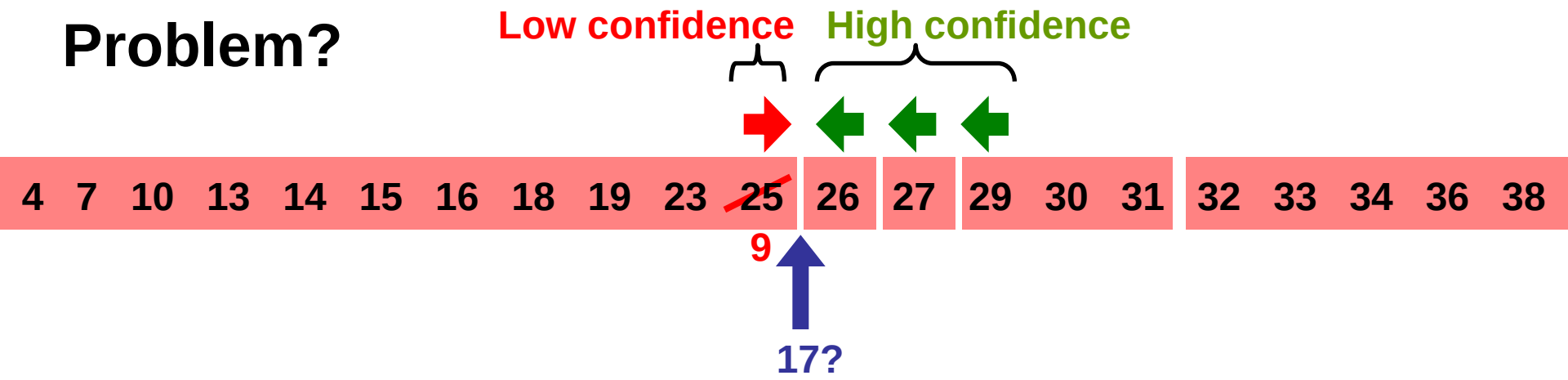
  Finocchi, Grandoni, Italiano, ICALP'06

- Upper bound

  Brodal, Fagerberg, Finocchi, Grandoni, Italiano, Jørgensen, Moruz, Mølhave, ESA'07

Faulty-Memory RAM: Searching

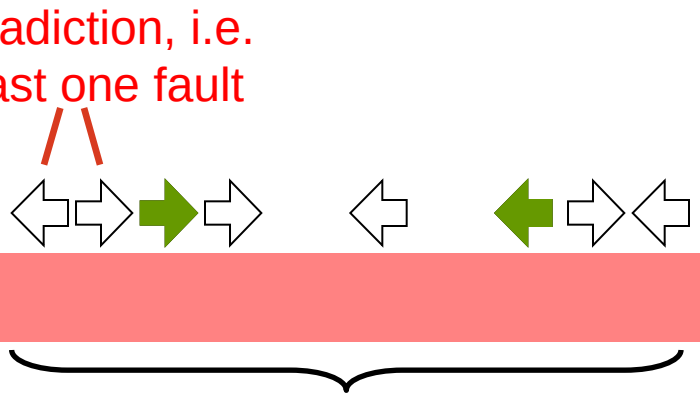
Problem?



Requirement: If there exists an uncorrupted element equal to the search key, we should find such an element

Faulty-Memory RAM: Searching

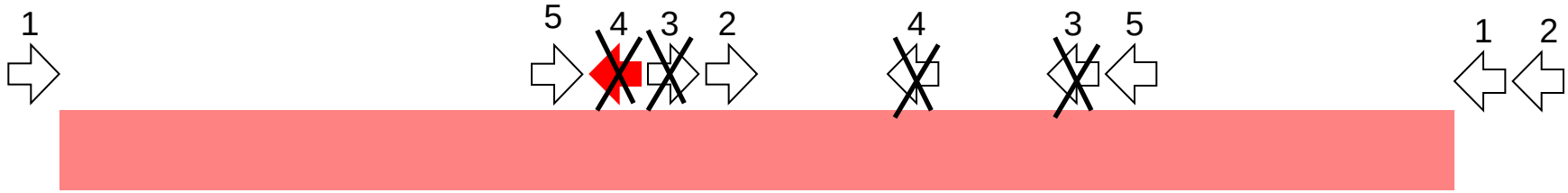
When are we done (δ=3)?



If range contains at least $\delta+1$ ⇨ and $\delta+1$ ⇦ then there is at least one uncorrupted ➡ and ⬅, i.e. x must be contained in the range

Faulty-Memory RAM: $\Theta(\log N + \delta)$ Searching

Brodal, Fagerberg, Finocchi, Grandoni, Italiano, Jørgensen, Moruz, Mølhave, ESA'07



If verification fails

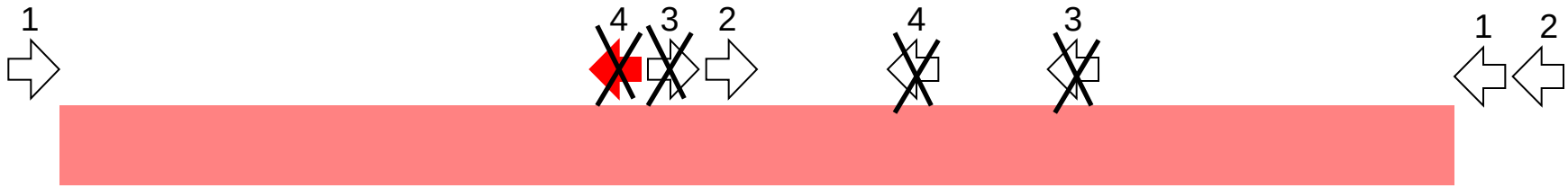
→ contradiction, i.e. ≥1 memory-fault

→ ignore 4 last comparisons

→ backtrack one level of search

Faulty-Memory RAM: $\Theta(\log N + \delta)$ Searching

Brodal, Fagerberg, Finocchi, Grandoni, Italiano, Jørgensen, Moruz, Mølhave, ESA'07

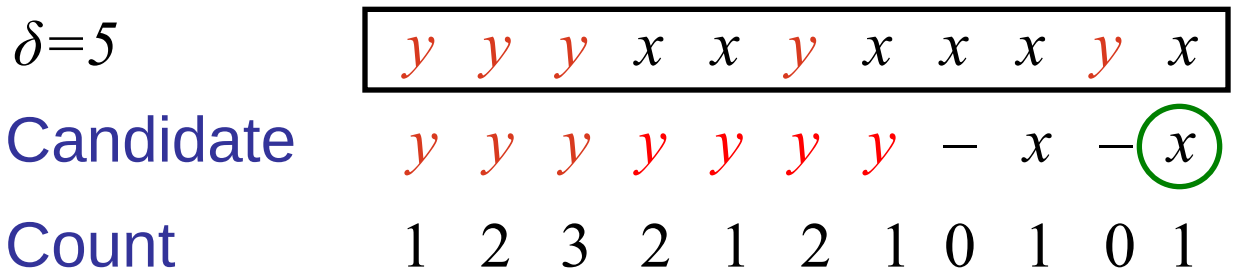


- Standard binary search + verification steps
- At most δ verification steps can fail/backtracking
- **Detail**: Avoid repeated comparison with the same (wrong) element by grouping elements into blocks of size $O(\delta)$

Faulty-Memory RAM: Reliable Values

- Store $2\delta+1$ copies of value x - at most δ copies uncorrupted
- x = majority
- Time $O(\delta)$ using two safe registers (candidate and count)

Boyer and Moore '91

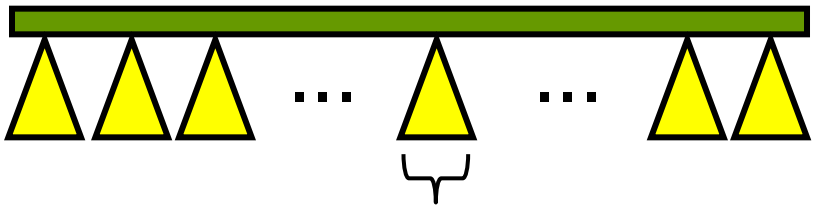


Faulty-Memory RAM: Dynamic Dictionaries

Brodal, Fagerberg, Finocchi, Grandoni, Italiano, Jørgensen, Moruz, Mølhave, ESA'07

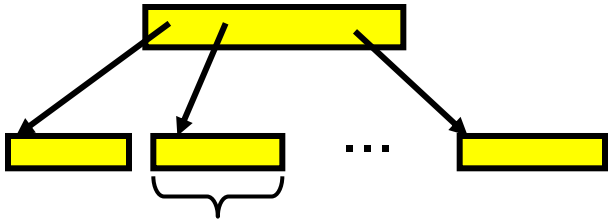
- Packed array
- Reliable pointers and keys
- Updates $O(\delta \cdot \log^2 N)$
- Searches = fault tolerant $O(\log N + \delta)$

Itai, Konheim, Rodeh, 1981



$\Theta(\delta \cdot \log N)$ elements

- 2-level buckets of size $O(\delta \cdot \log N)$
- Root: Reliable pointers and keys
- Bucket search/update amortized $O(\log N + \delta)$



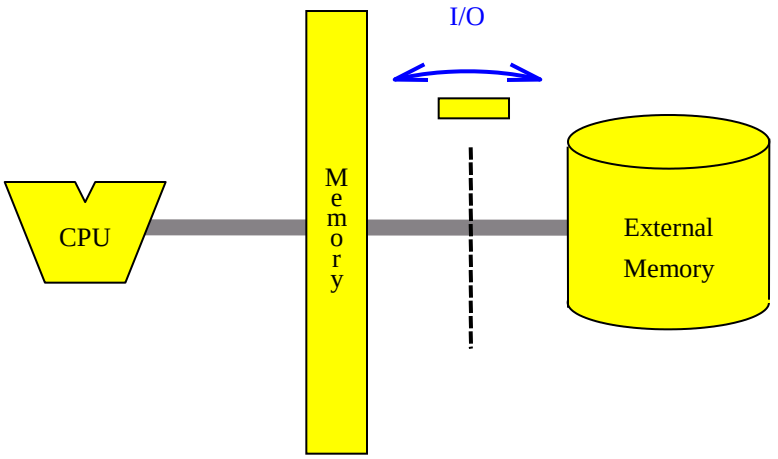
$\Theta(\delta)$ elements

- **Search and update amortized $O(\log N + \delta)$**

I/O Model

I/O Model

Aggarwal and Vitter 1988

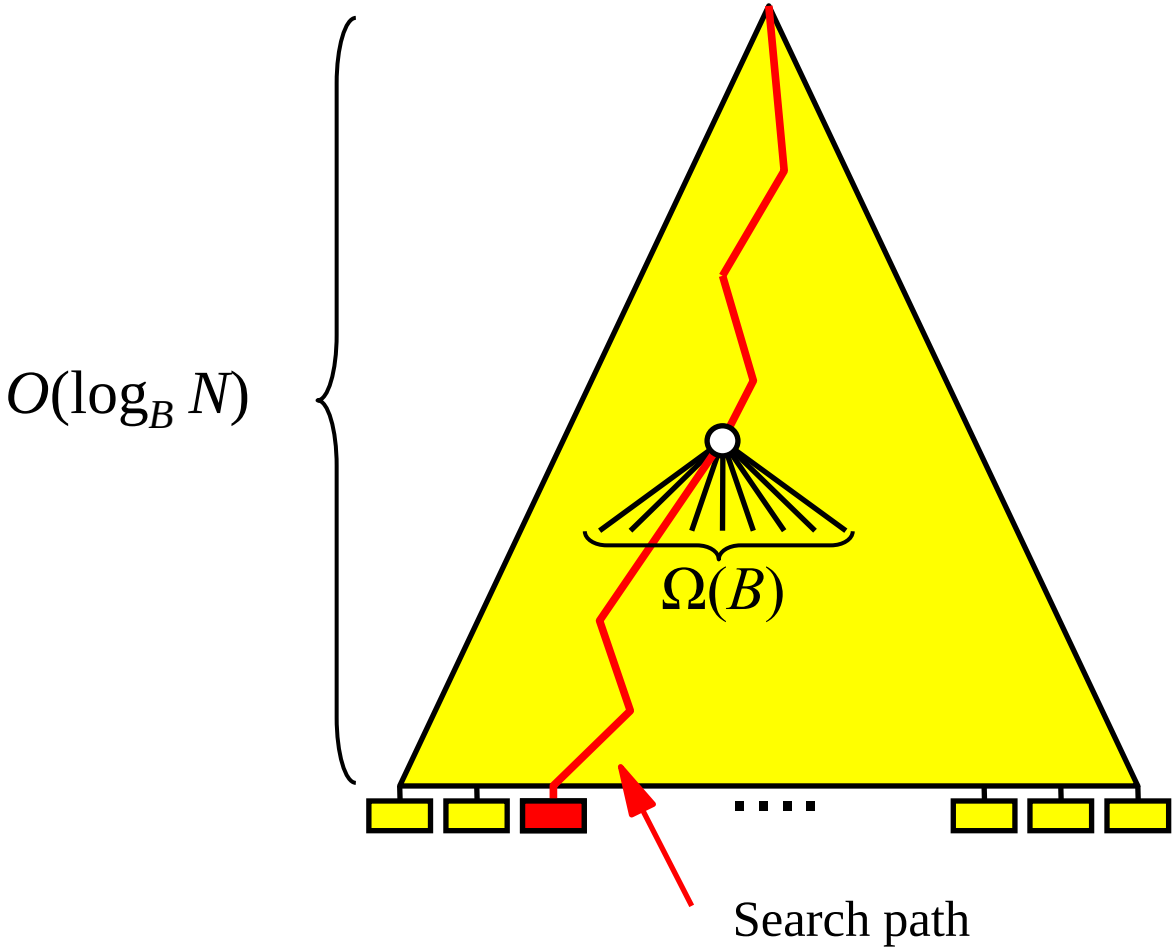


- N = problem size
- M = memory size
- B = I/O block size

- One I/O moves B consecutive records from to disk
- Complexity = number of I/Os

- **Example**: Sorting requires $\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$ I/Os

B-trees



- **Search and update $O(\log_B N)$**

Fault-Tolerance

versus

I/O Efficiency

Lower Bound for Fault-Tolerant External Searching

- Adversary argument

- If B^{ε} slabs per I/O $\rightarrow$ factor B^{ε} reduction and $B^{1-\varepsilon}$ faults
- After k I/Os $N/(B^{\varepsilon})^{k} - k \cdot B^{1-\varepsilon}$ elements remain

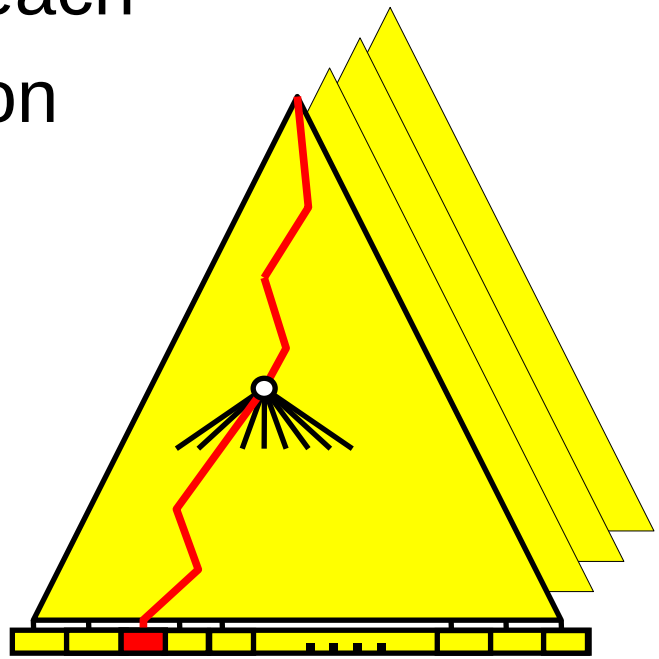
- $\Omega\left(\frac{1}{\varepsilon}\log_B N + \frac{\delta}{B^{1-\varepsilon}}\right)$ I/Os required [minimized wrt ε]

Randomized Upper Bound for Fault-Tolerant External Searching

- Sorted array + 2δ identical B-trees (over $N/(2\delta)$ elements, stored in BFS layout)
- Search: Select random tree for each node on search path + verification
- Probability no faults on path:

$$\prod_{i=1}^{\log_B N}\left(1-\frac{\beta_i}{2\delta}\right)\geq\frac{1}{2} \text{ where } \Sigma\beta_i\leq\delta$$

- **Search $O(\log_B N+\delta/B)$ expected**

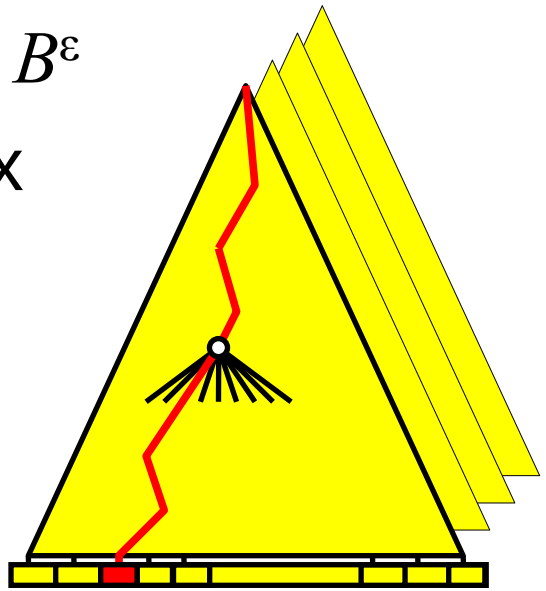


Deterministic Upper Bound for Fault-Tolerant External Searching

- Sorted array

  $+ 2\delta/B^{1-\varepsilon}$ identical B-trees of degree B^{ε}

  $+ B^{1-\varepsilon}$ copies of each key + min/max
- Search: Verify against min/max in each step – if fail, backtrack one level and advance to next copy



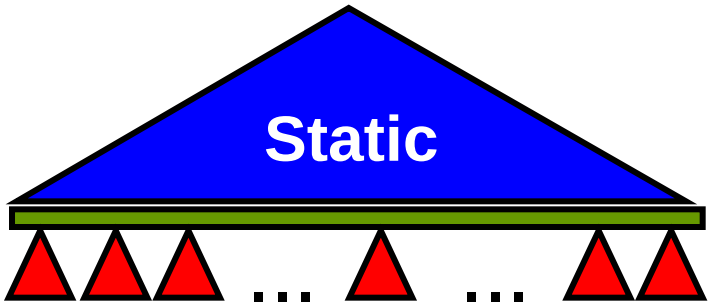
- **Search** $O\left(\frac{1}{\varepsilon}\log_B N + \frac{\alpha}{B^{1-\varepsilon}} + \frac{\delta}{B}\right)$ **I/Os**

Dynamic Fault-Tolerant External Dictionaries

Static structure

+ Packed arrays

+ Buckets of size $O(\delta \cdot \log^3 N)$



- **Deterministic**

  $O\left(\frac{1}{\varepsilon}\log_B N + \frac{\delta}{B^{1-\varepsilon}}\right)$ I/Os search and updates

- **Randomized**

  Expected $O(\log_B N + \delta/B)$ I/Os search and updates

Conclusion

- Fault-tolerant external memory searching

$$\Theta\left(\frac{1}{\varepsilon}\log_B N + \frac{\delta}{B^{1-\varepsilon}}\right) \text{ I/Os}$$

worst-case [minized wrt ε]

- Randomized $O(\log_B N + \delta/B)$ I/Os

Future Work

Fault Tolerance versus I/O Efficiency

- Randomized algorithms:

  Memory faults in internal memory?

- Sorting:

$$\Theta\left(\frac{N}{B}\log_{M/B}\frac{N}{B}+\frac{\delta^2}{B}\right) ?$$

- ...

THANKS