External Memory III

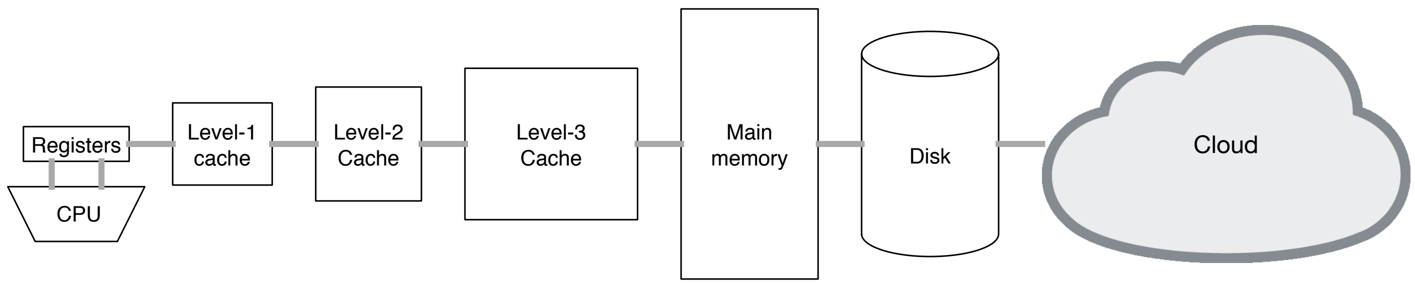
- Computational Models
- Scanning
- Double Array Traversal
- Searching

Philip Bille

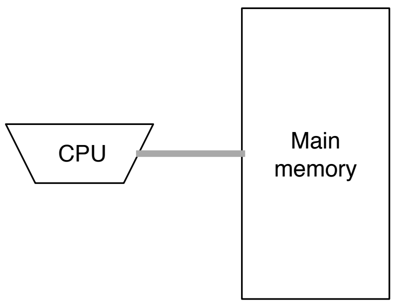
External Memory III

- Computational Models
- Scanning
- Double Array Traversal
- Searching

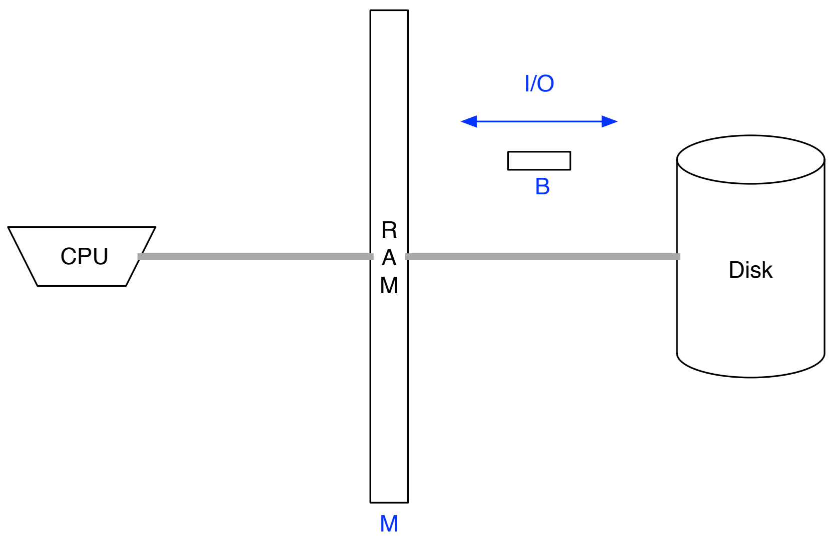
Computational Models



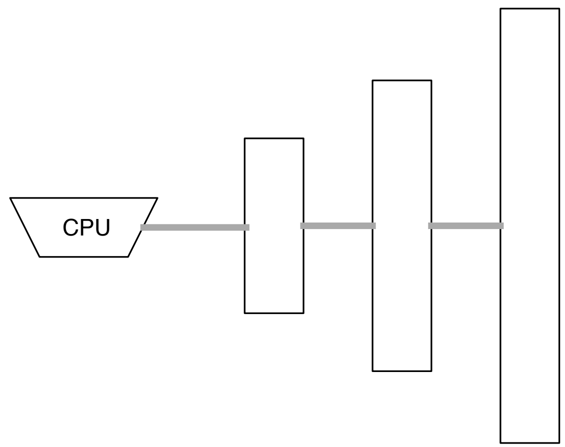
Real computer



Word RAM



I/O model



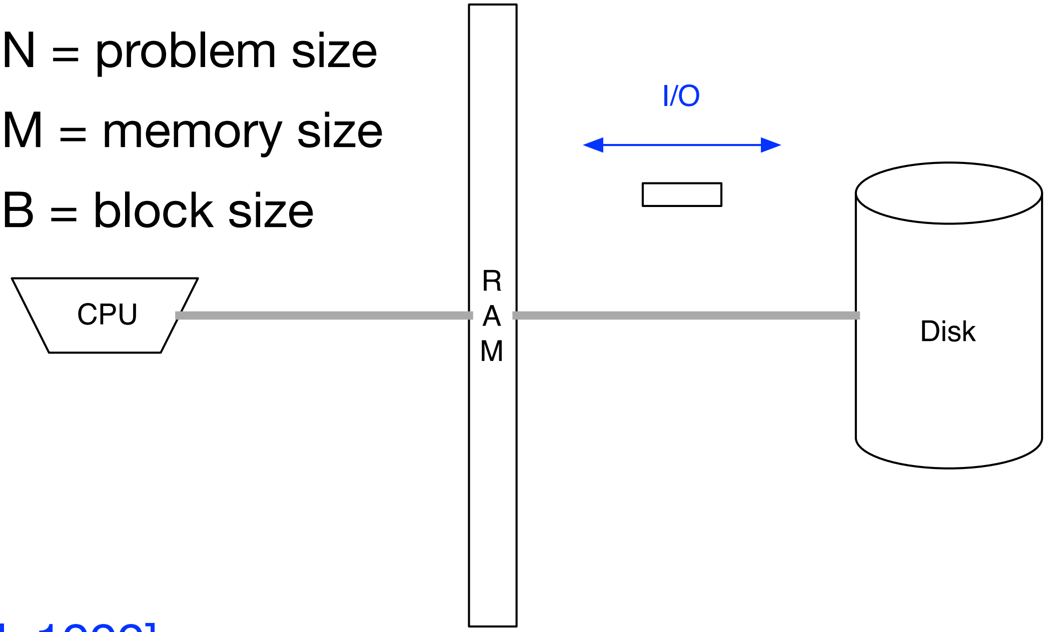
Multilevel models

Computational Models

N = problem size

M = memory size

B = block size

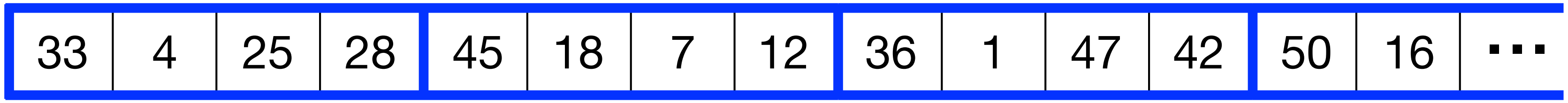


- Cache-oblivious model [Frigo et al. 1999].
    - Identical to I/O model except algorithms do not know B and M.
    - Program in the RAM model.
    - Analyze in the I/O model for arbitrary B and M.
    - Optimal off-line cache replacement strategy.
- Properties.
    - Efficient on one level of cache $\Longrightarrow$ efficient on all levels cache.
    - Portable + self-tunable + simple.

External Memory III

- Computational Models
- Scanning
- Double Array Traversal
- Searching

Scanning

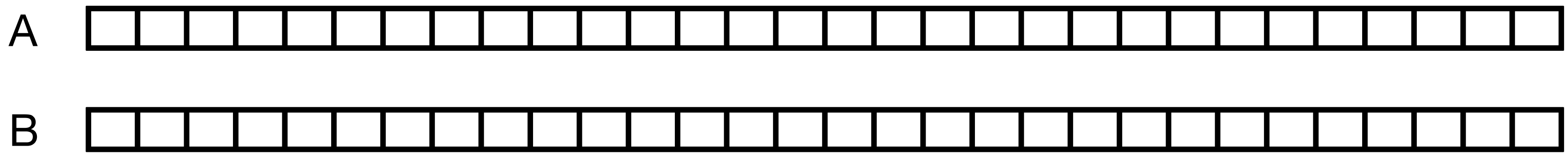


- Scanning. Given an array A of N values stored in N/B blocks and a key x, determine if x is in A.
- I/Os. O(N/B).

External Memory III

- Computational Models
- Scanning
- Double Array Traversal
- Searching

Double Array Traversal




```
for(int i = 0; i < n; i++) {
    for(int j = 0; j < n; j++) {
        f(A[i], B[j]);
    }
}
```

- Double array traversal. Given arrays A and B of length n (N = 2n) and a function f, compute f(A[i], B[j]) for all i, j.

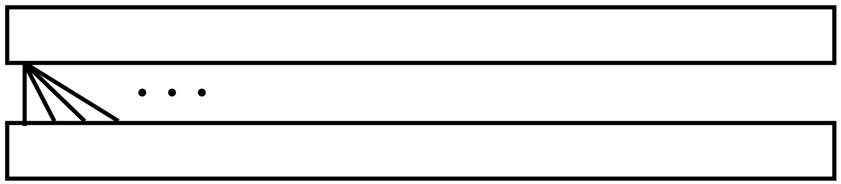
Double Array Traversal

- Applications.
    - Join in data bases.
    - Dynamic programming.
    - Matrix multiplication.
    - ...

Double Array Traversal

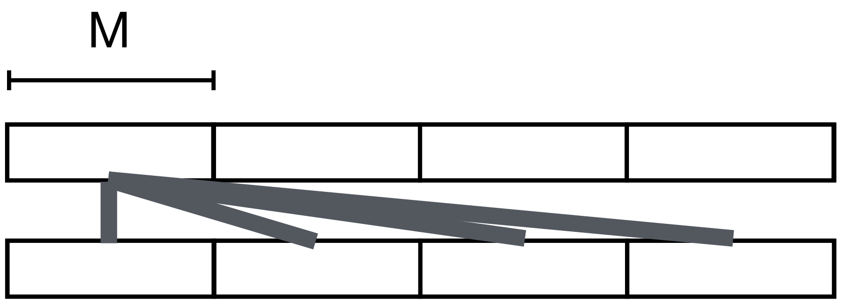
- Solution in 3 steps.
  - RAM algorithm.
  - Cache-conscious algorithm.
  - Cache-oblivious algorithm.

Double Array Traversal



- RAM algorithm.
- I/Os. $O(N^2/B)$.

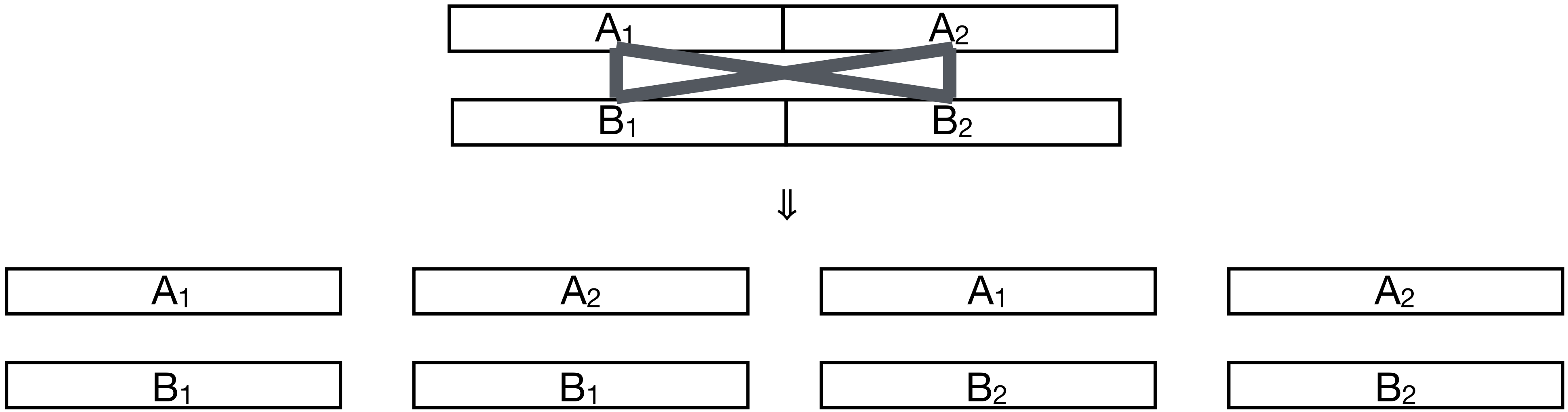
Double Array Traversal



- Cache-conscious algorithm.
  - Partition into N/M subarrays of size M
  - For each pair of subarray: read into memory and evaluate.
- I/Os.

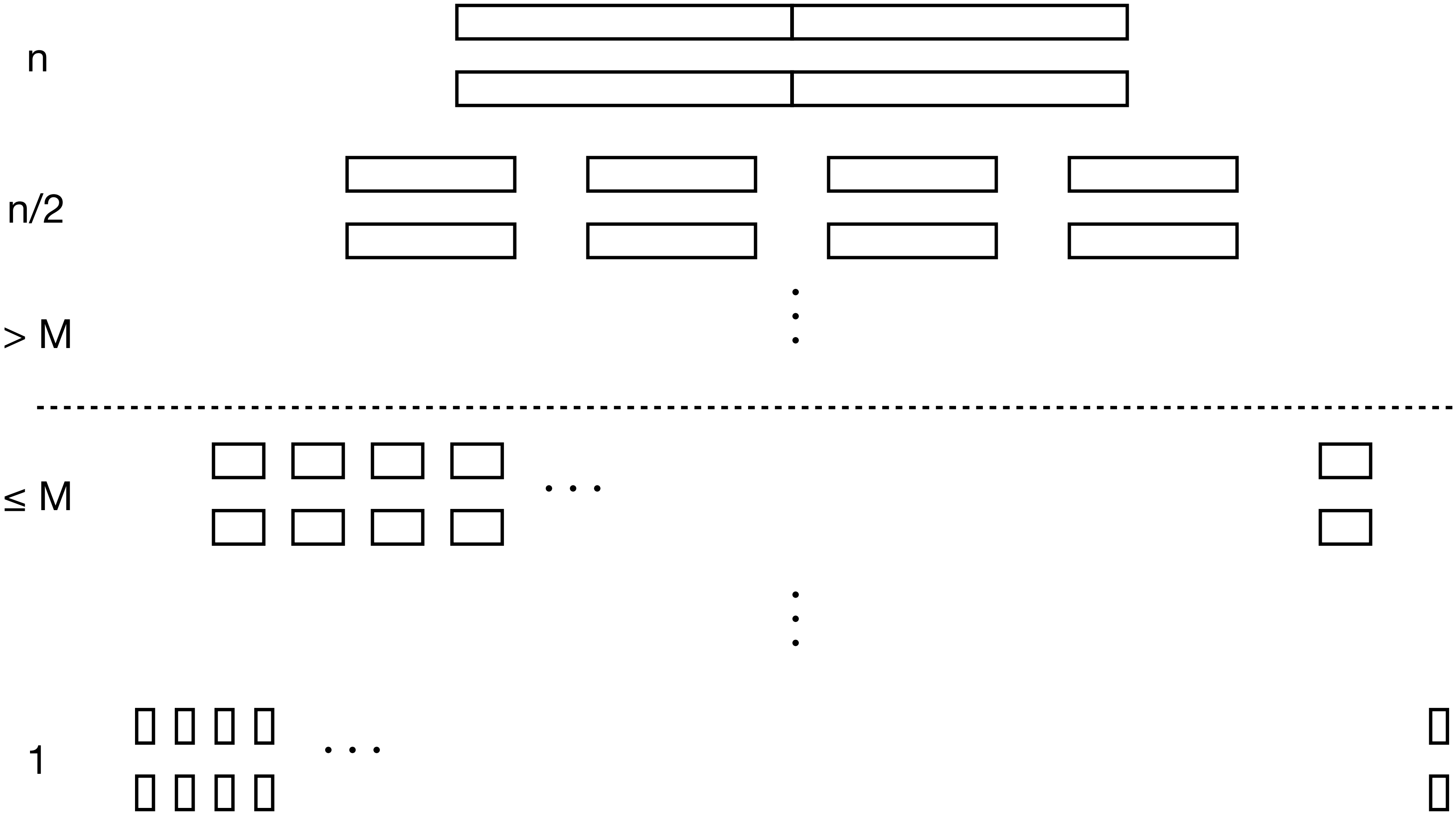
$$O\left(\frac{n}{M} \cdot \frac{n}{M} \cdot \frac{M}{B}\right) = O\left(\frac{n^2}{MB}\right) = O\left(\frac{N^2}{MB}\right)$$

Double Array Traversal



- Cache-oblivious algorithm.
    - Divide into n/2 sized subarrays and recurse.
    - Evaluate function when length is 1.
- I/Os?

Double Array Traversal



- I/Os

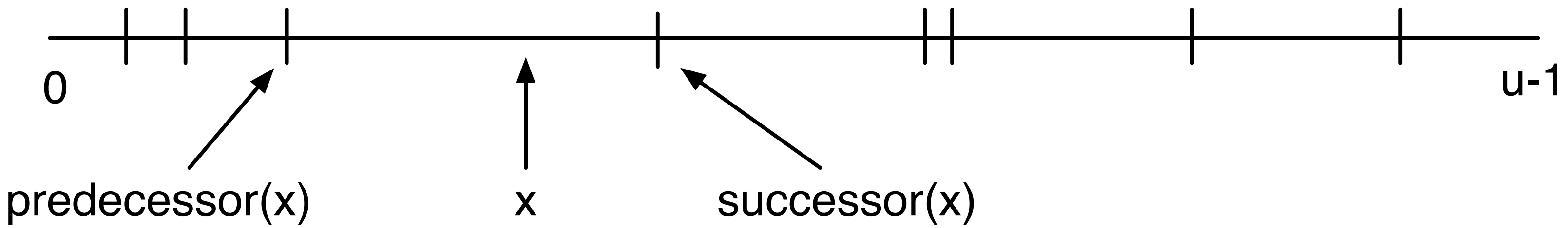
$$O\left(\frac{n}{M}\cdot\frac{n}{M}\cdot\frac{M}{B}\right) = O\left(\frac{n^2}{MB}\right) = O\left(\frac{N^2}{MB}\right)$$

External Memory III

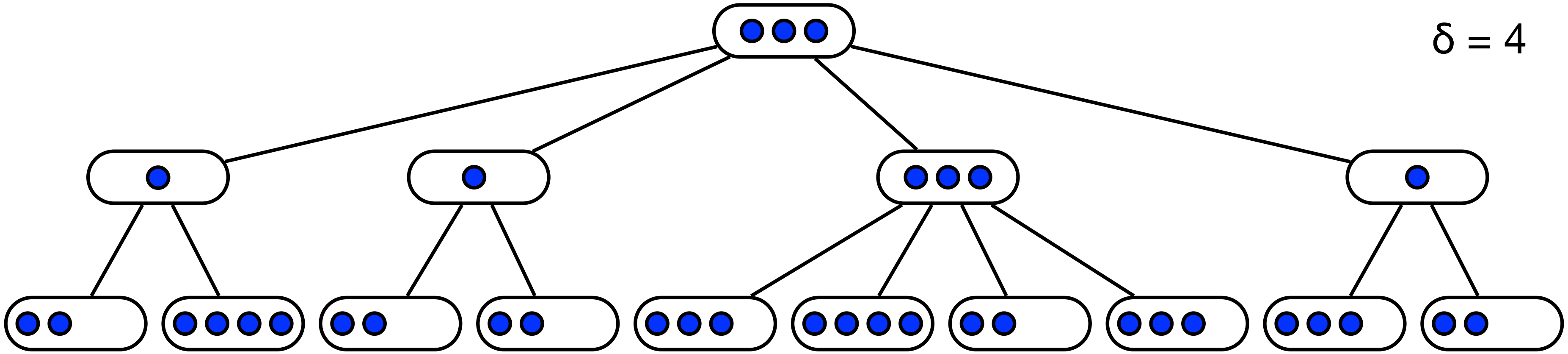
- Computational Models
- Scanning
- Double Array Traversal
- Searching

Searching

- Searching. Maintain a set $S \subseteq U = \{0, ..., u-1\}$ supporting
  - member(x): determine if $x \in S$
  - predecessor(x): return largest element in $S \leq x$.
  - successor(x): return smallest element in $S \geq x$.
  - insert(x): set $S = S \cup \{x\}$
  - delete(x): set $S = S - \{x\}$

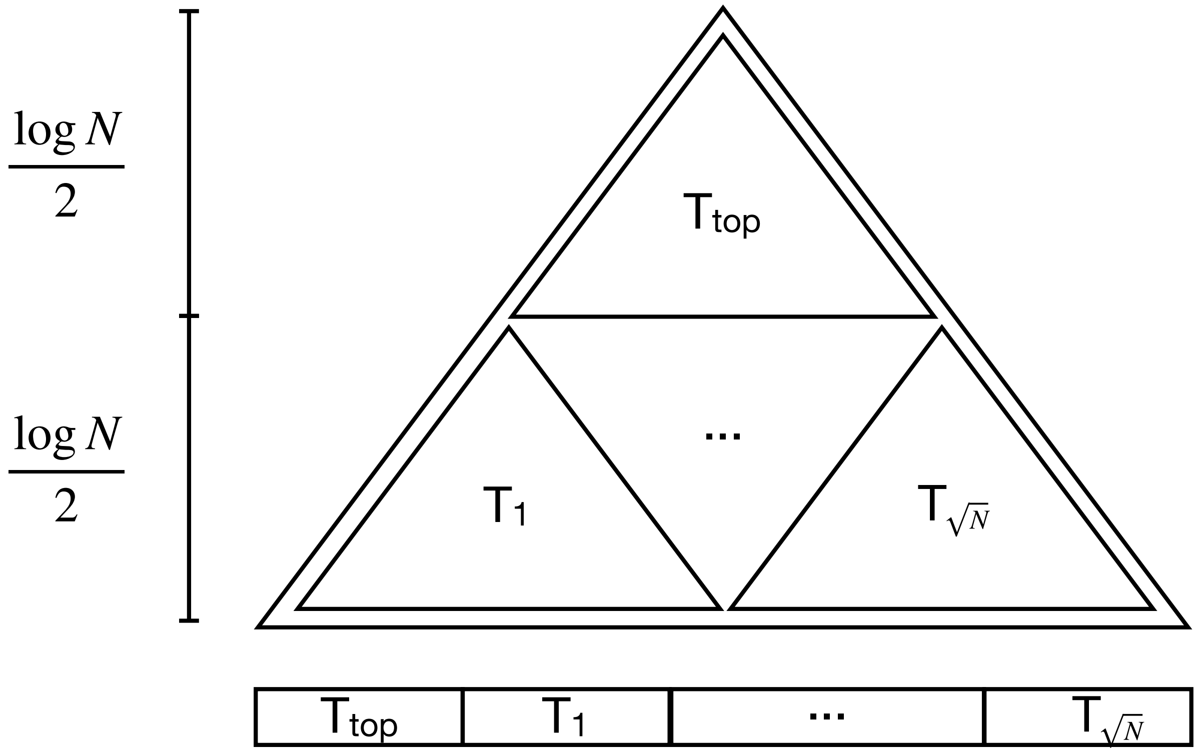


Searching



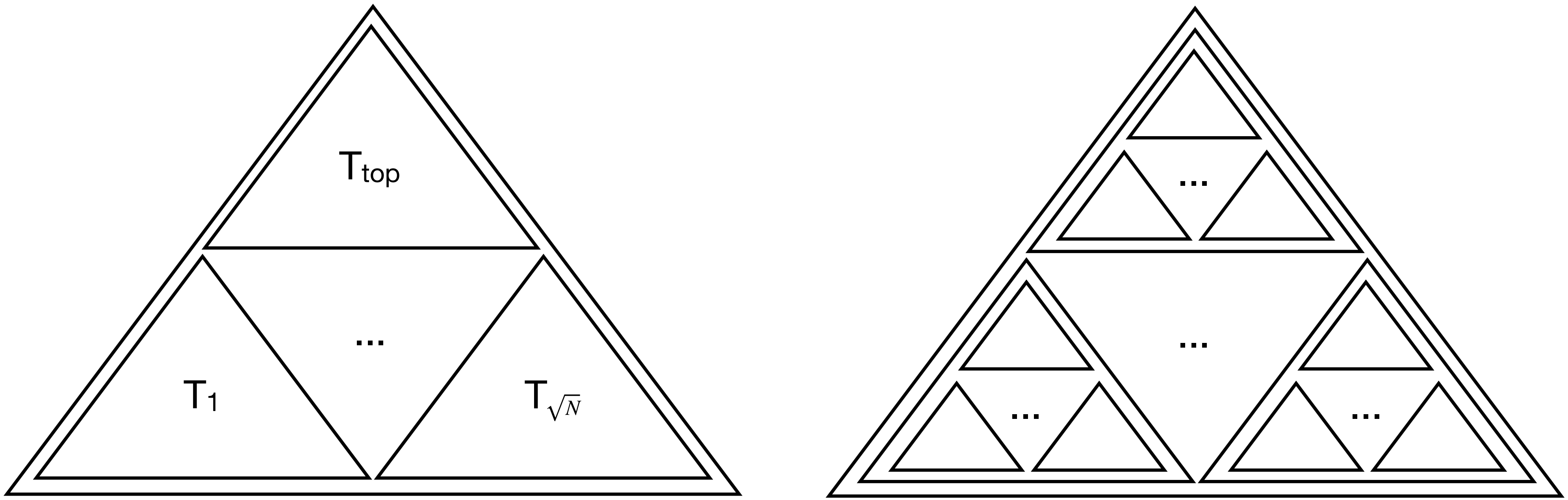
- B-trees.
  - Searching in $O(\log_B N)$ I/Os.
  - How can we get B-tree search bound in cache-oblivious model?

Searching



- Van Emde Boas layout.
  - Perfect balanced binary search tree T with N leaves.
  - Divide T into top tree T_{top} and bottom trees $T_1,..,T_{\sqrt{N}}$. by splitting at height (log N)/2.
  - Layout T_{top} followed by $T_1,..,T_{\sqrt{N}}$ consecutively in memory.
  - Recurse.

Searching



- Searching.
  - Consider first level where subtrees have size ≤ B.
  - Any search path intersects log N / log B subtrees.
  - ⇒ $O(\log_B N)$ I/Os.

Basic Bounds

	Internal	External (even cache-oblivious)
Scanning	O(N)	scan(N) = O(N/B)
Sorting	O(Nlog N)	sort(N) = $O((N/B)\log_{M/B} (N/B))$
Searching	O(log N)	search(N) = $O(\log_B N)$

External Memory III

- Computational Models
- Scanning
- Sorting
- Searching