Cache Oblivious Searching and Sorting

Gerth Stølting Brodal

BRICS

University of Aarhus

Joint work with Rolf Fagerberg (Århus), Riko Jacob (Munich), Michael A. Bender, Dongdong Ge, Simai He, Haodong Hu (SUNY Stony Brook), John Iacono (Polytechnic, NY), Alejandro López-Ortiz (Waterloo)

IT University of Copenhagen, April 30, 2003

Outline of Talk

- Hardware
- Computational models
  - RAM model (Random Access Machine)
  - IO model
  - Cache oblivious model
- Binary searching and dictionaries
- Sorting
- Priority queues
- Concluding remarks

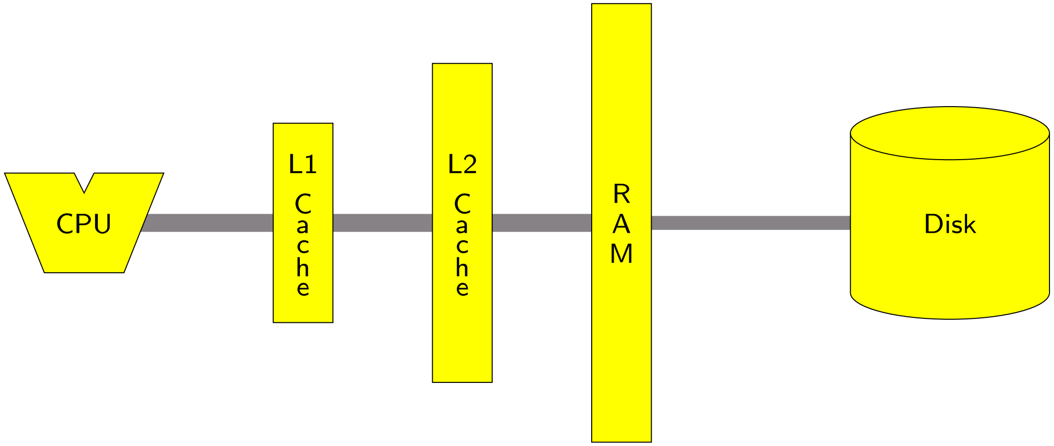
Hardware

Hardware

- Dell Latitude L400, 700Mhz (January 2002)
- Mobile Intel Pentium III
- Primary 16 Kb instruction cache and 16 Kb write-back data cache
- 256 Kb Level 2 Cache
- 256 Mb SDRAM
- 10 Gb disk

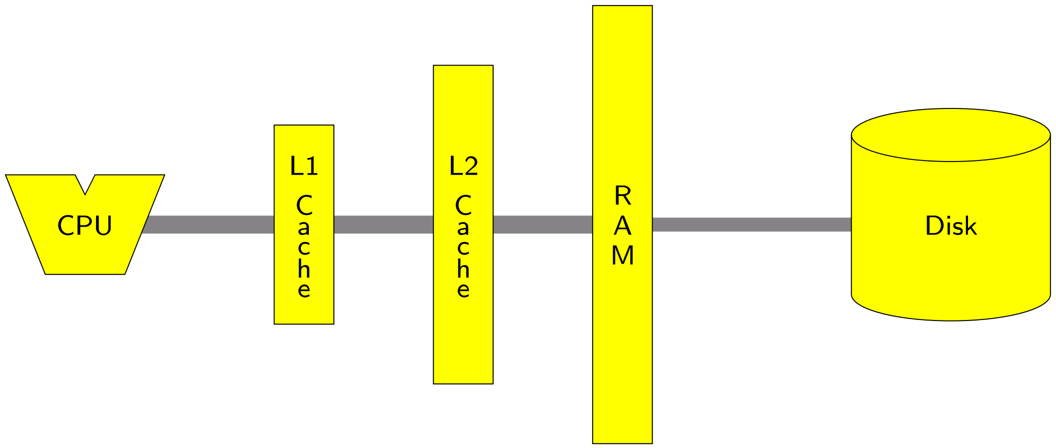
Hardware

- Dell Latitude L400, 700Mhz (January 2002)
- Mobile Intel Pentium III
- Primary 16 Kb instruction cache and 16 Kb write-back data cache
- 256 Kb Level 2 Cache
- 256 Mb SDRAM
- 10 Gb disk



Memory hierarchy

Trends in Implementation Technology



	L1 Cache	L2 Cache	Virtual memory
Block size	4 – 32 bytes	32 – 256 bytes	4 – 16 KB
Hit time (cycles)	1 – 2	6 – 15	10 – 100
Miss penalty (cycles)	8 – 66	30 – 200	700.000 – 6.000.000
Size	1 – 128 KB	256 KB – 16 MB	16 – 8192 MB

Source: *Computer Architecture – A Quantitative Approach*, Hennessy & Patterson, 2nd. Ed. 1996

The Unknown Machine

Algorithm

↓

C program

↓ gcc

Object code

↓ linux

Execution

Can be executed on machines with a specific class of CPUs

Algorithm

↓

Java program

↓ javac

Java bytecode

↓ java

Interpretation

Can be executed on any machine with a Java interpreter

The Unknown Machine

Algorithm	Algorithm
↓	↓
C program	Java program
↓ gcc	↓ javac
Object code	Java bytecode
↓ linux	↓ java
Execution	Interpretation

Can be executed on machines with a specific class of CPUs

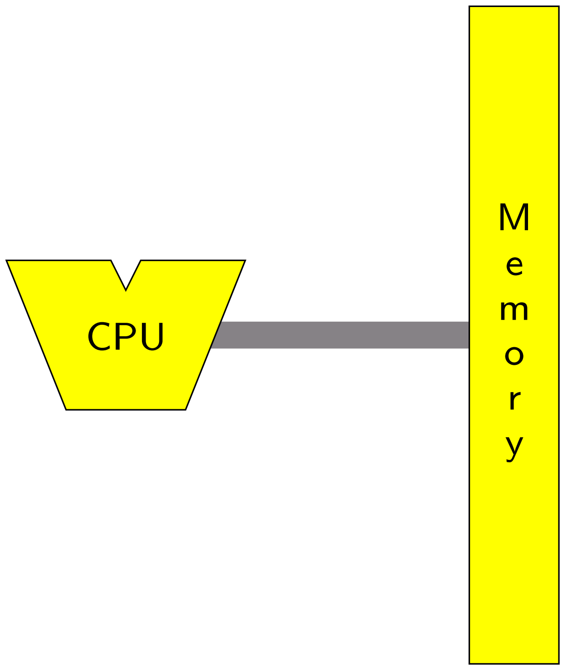
Can be executed on any machine with a Java interpreter

Goal Develop algorithms that are optimized w.r.t. memory hierarchies without knowing the parameters

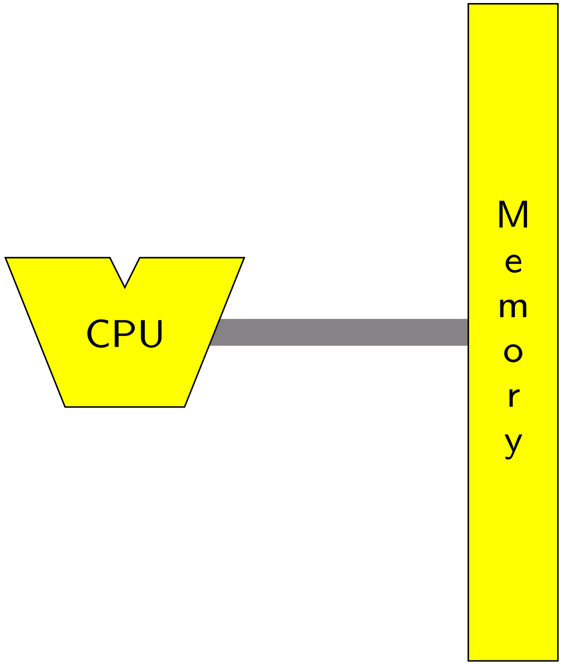
Outline of Talk

- Hardware
- Computational models
  - RAM model (Random Access Machine)
  - IO model
  - Cache oblivious model
- Binary searching and dictionaries
- Sorting
- Priority queues
- Concluding remarks

RAM Model
(Random Access Machine)



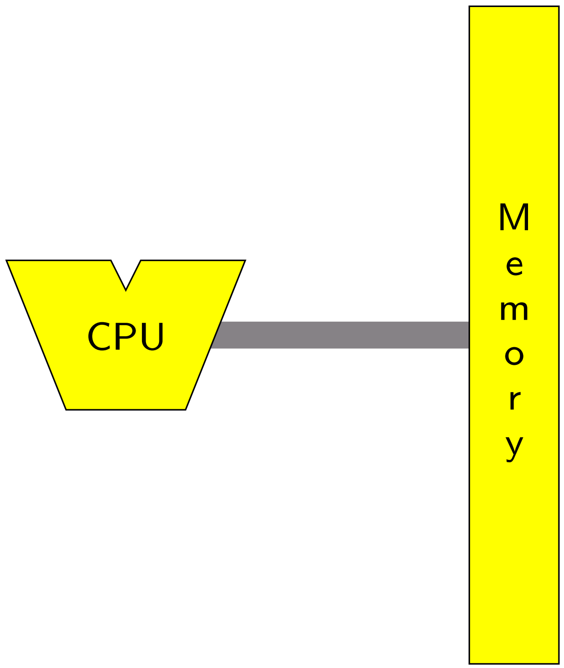
RAM Model
(Random Access Machine)



$+ - * / \vee \wedge \neq$ ... $O(1)$ time

Memory access $O(1)$ time

RAM Model
(Random Access Machine)



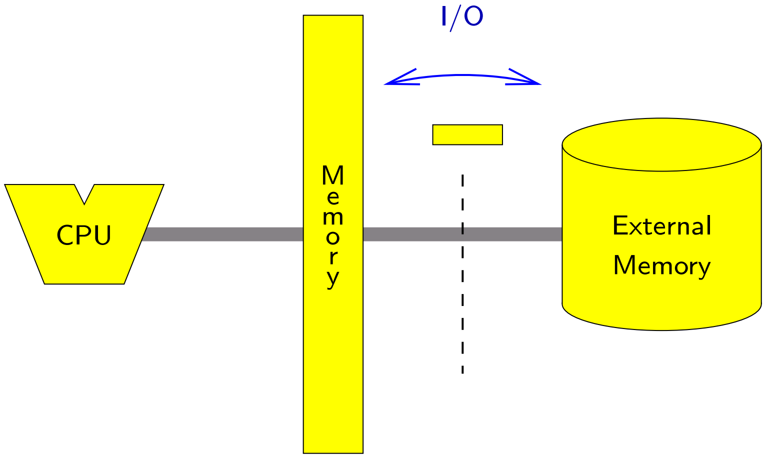
$+ - * \, / \, \vee \wedge \neq$ ... $O(1)$ time

Memory access ~~$O(1)$ time~~

Ignores the presence of memory hierarchies

I/O Model

Aggarwal and Vitter 1988



N = problem size

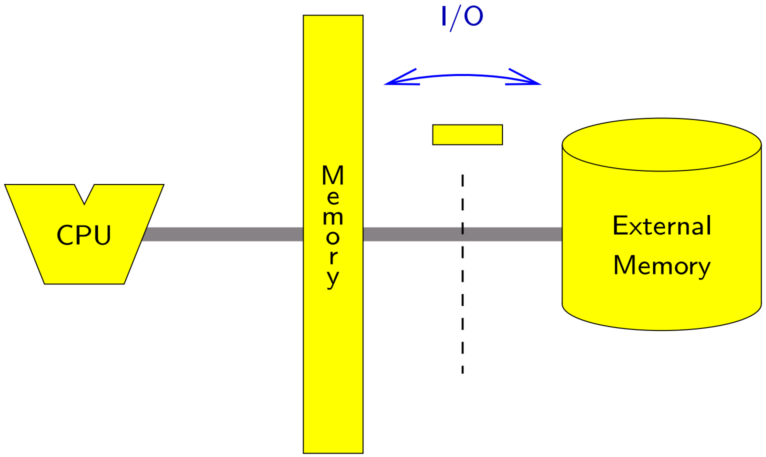
M = memory size

B = I/O block size

- One I/O moves B consecutive records from/to disk
- Cost: number of I/Os

I/O Model

Aggarwal and Vitter 1988



N = problem size

M = memory size

B = I/O block size

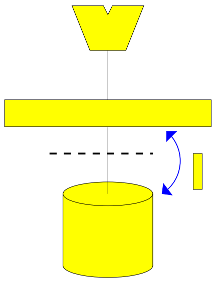
- One I/O moves B consecutive records from/to disk
- Cost: number of I/Os

Scan$(N) = O(N/B)$ Sort$(N) = O\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$

Cache Oblivious Model

Frigo, Leiserson, Prokop, Ramachandran 1999

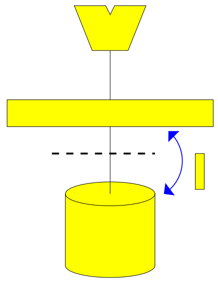
- Program in the RAM model
- Analyze in the I/O model (for arbitrary B and M)
- Optimal off-line cache replacement strategy



Cache Oblivious Model

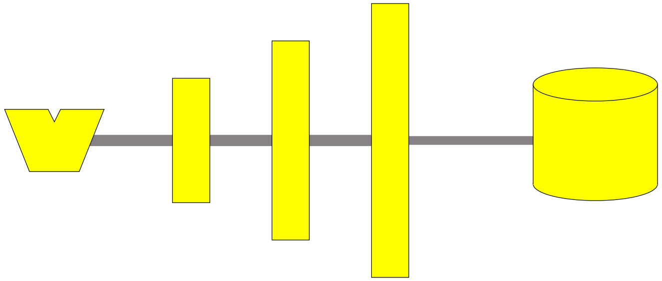
Frigo, Leiserson, Prokop, Ramachandran 1999

- Program in the RAM model
- Analyze in the I/O model (for arbitrary B and M)
- Optimal off-line cache replacement strategy



Advantages

- Optimal on arbitrary level $\Rightarrow$ optimal on all levels
- B and M not hard-wired into algorithm



Outline of Talk

- Hardware
- Computational models
  - RAM model (Random Access Machine)
  - IO model
  - Cache oblivious model
- ▶ Binary searching and dictionaries
- Sorting
- Priority queues
- Concluding remarks

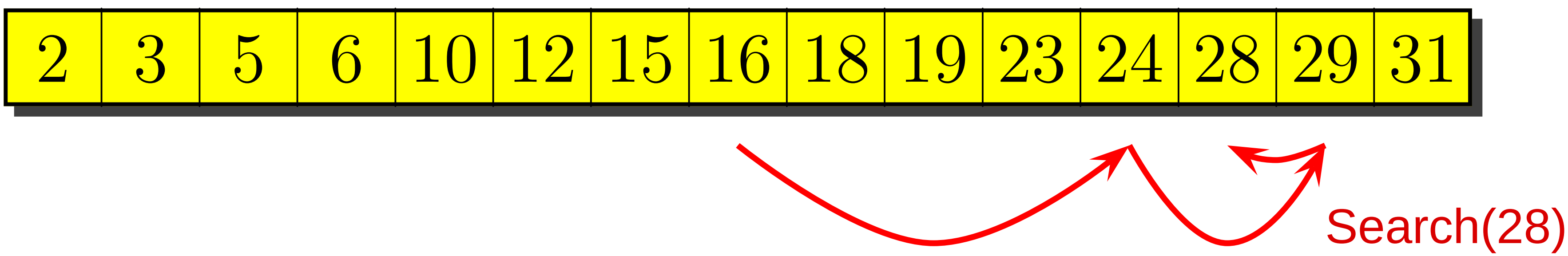
RAM model : Binary Searching

- Sorted array of n elements
  = static dictionary
- Binary search requires $O(\log_2 N)$ time

2	3	5	6	10	12	15	16	18	19	23	24	28	29	31

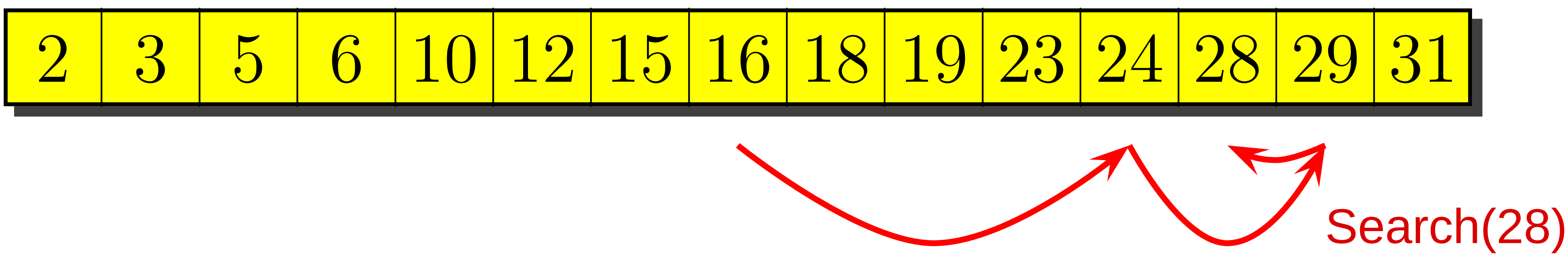
RAM model : Binary Searching

- Sorted array of n elements
  = static dictionary
- Binary search requires $O(\log_2 N)$ time



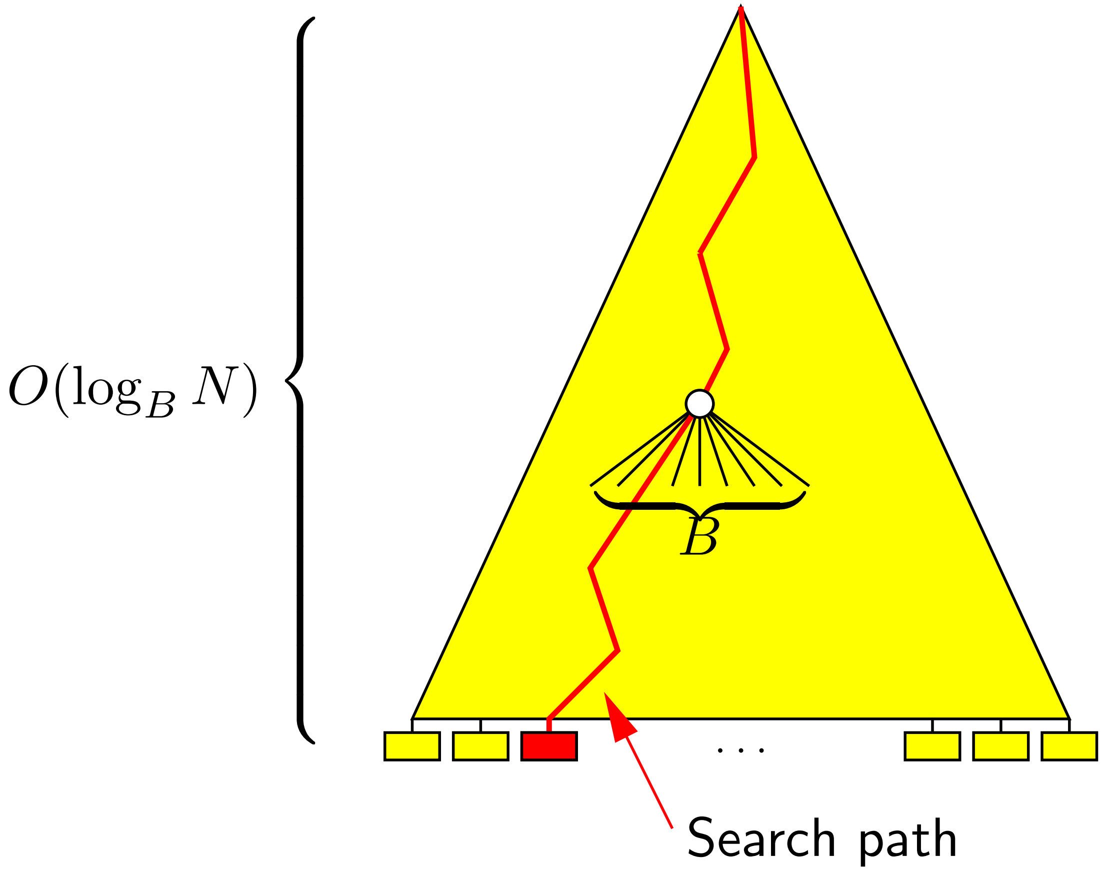
RAM model : Binary Searching

- Sorted array of n elements = static dictionary
- Binary search requires $O(\log_2 N)$ time



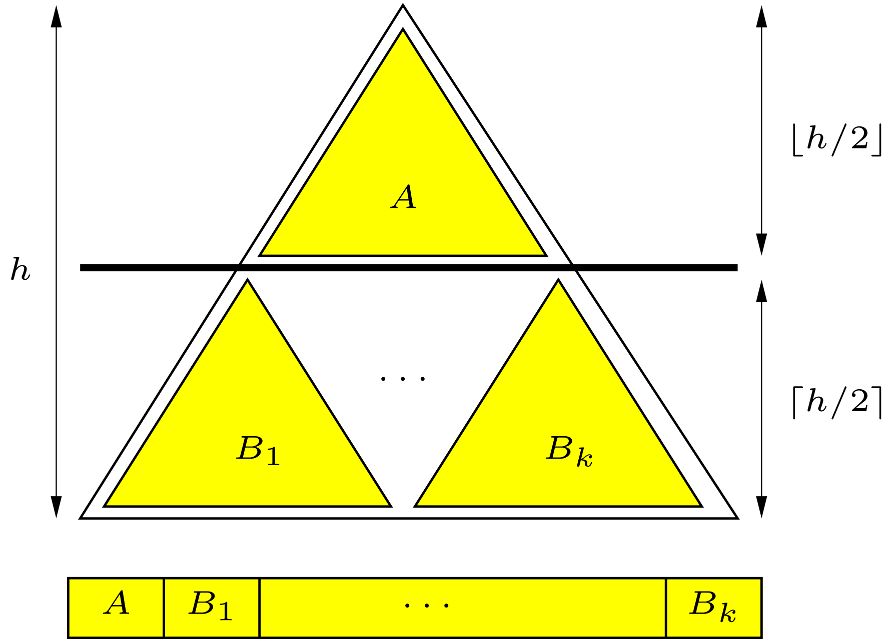
A binary search is cache oblivious and uses $O\left(\log_2 \frac{N}{B}\right)$ I/Os

IO model : B-trees



- Each node stores B keys and has degree $B + 1$
- Searches use $O(\log_B N)$ I/Os

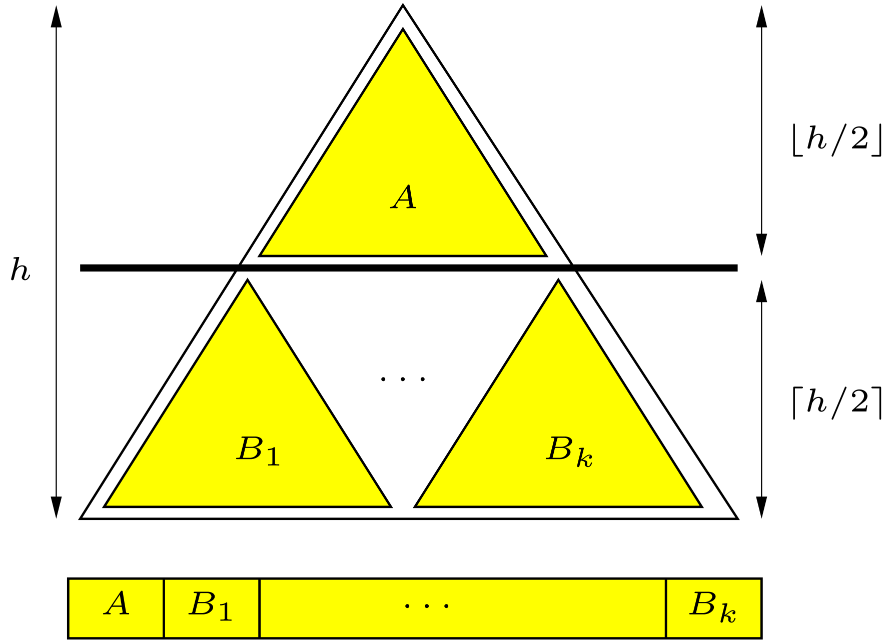
Static Cache Oblivious Dictionary

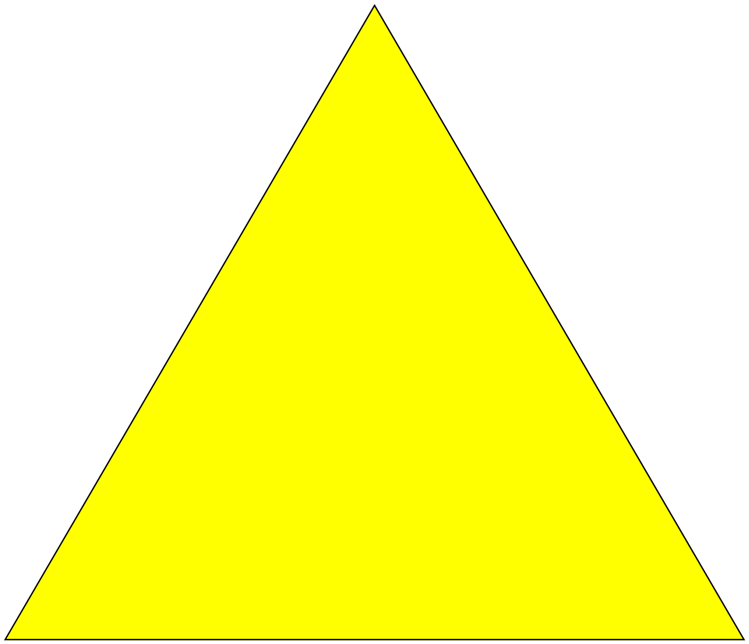


Recursive layout of binary tree

$\equiv$ van Emde Boas layout

Static Cache Oblivious Dictionary

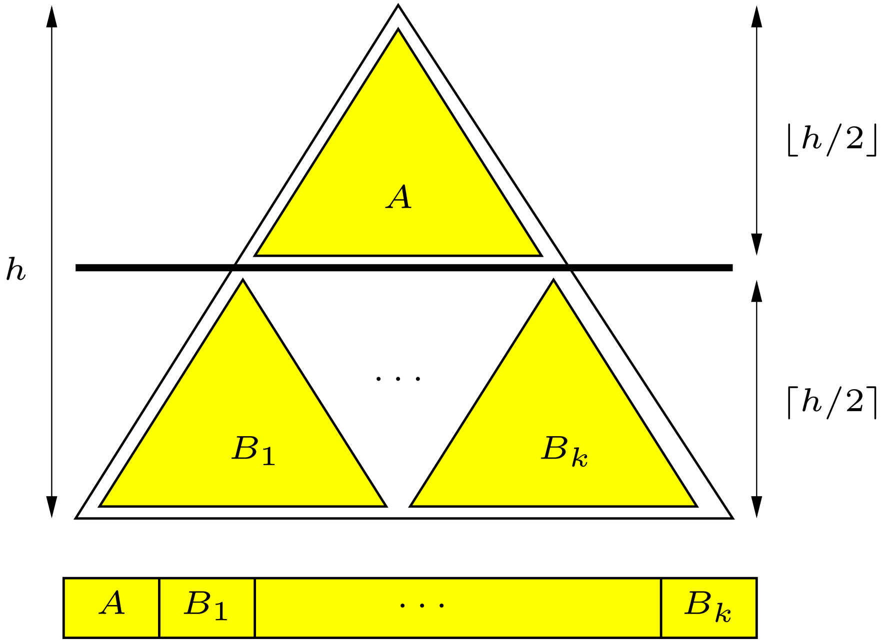


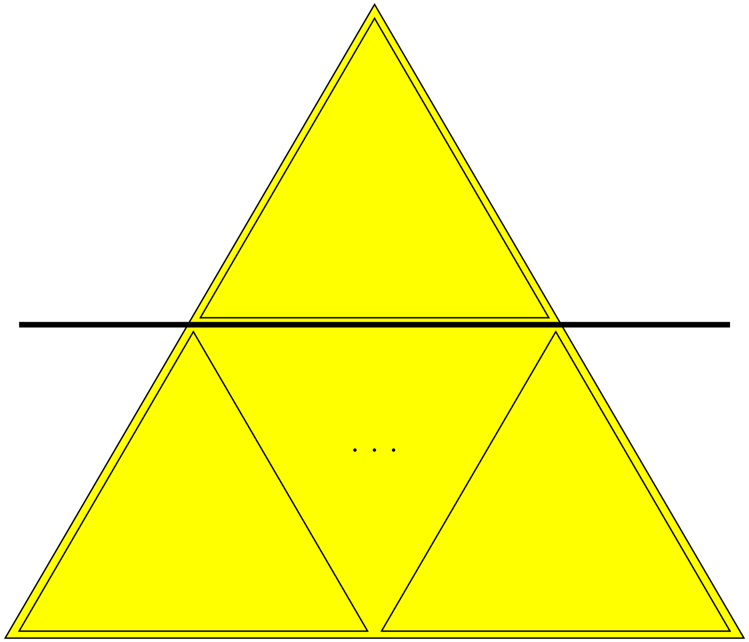


Recursive layout of binary tree

≡ van Emde Boas layout

Static Cache Oblivious Dictionary

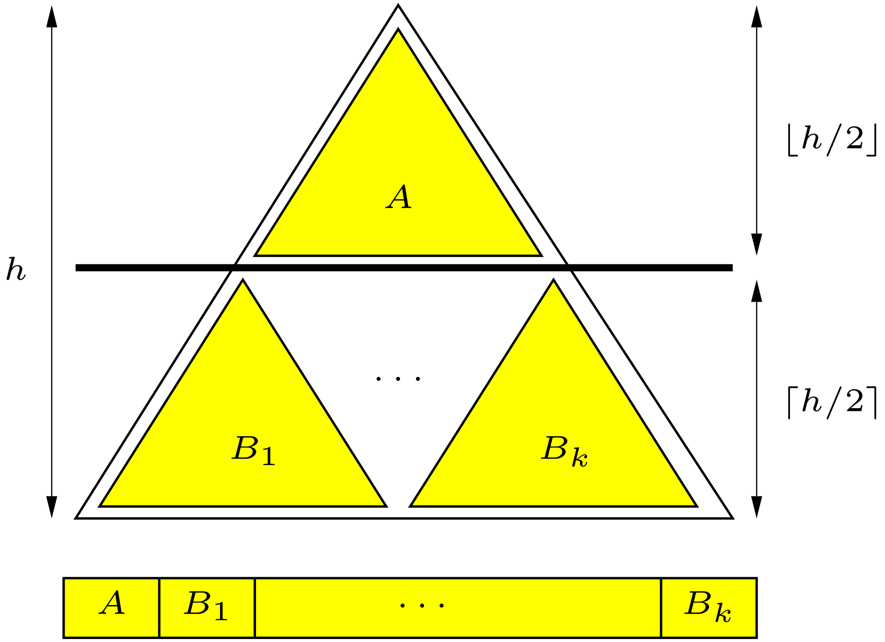


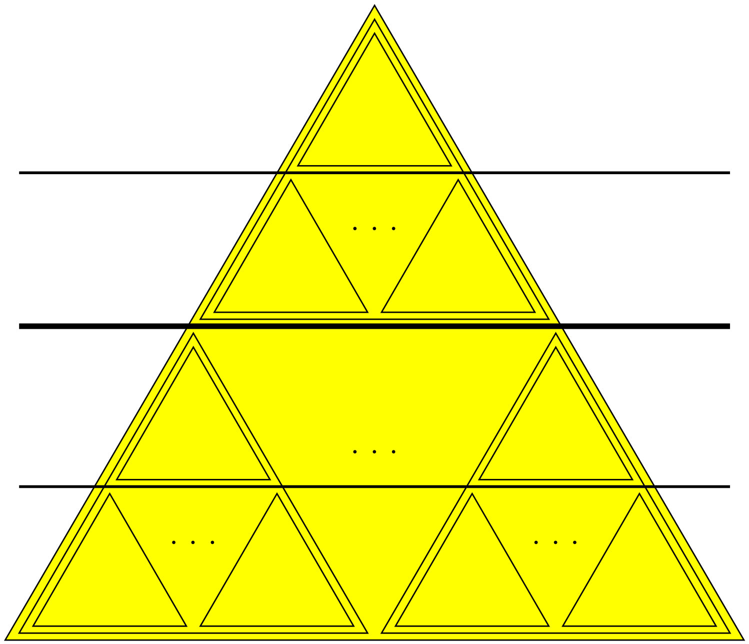


Recursive layout of binary tree

$\equiv$ van Emde Boas layout

Static Cache Oblivious Dictionary

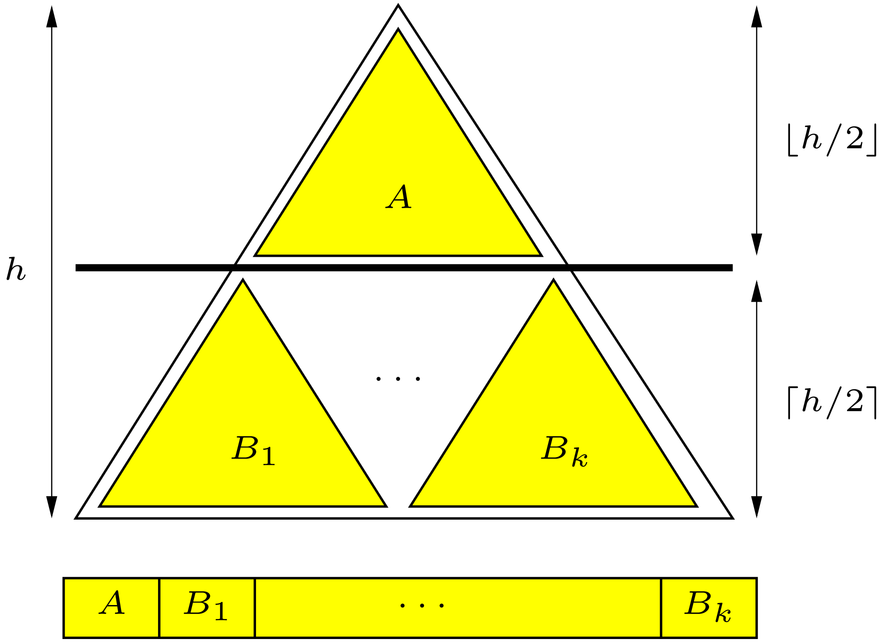


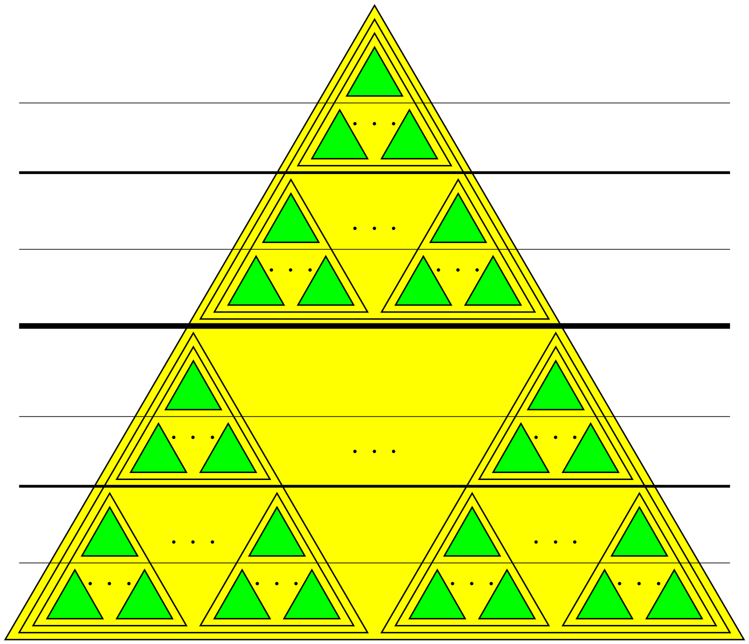


Recursive layout of binary tree

$\equiv$ van Emde Boas layout

Static Cache Oblivious Dictionary

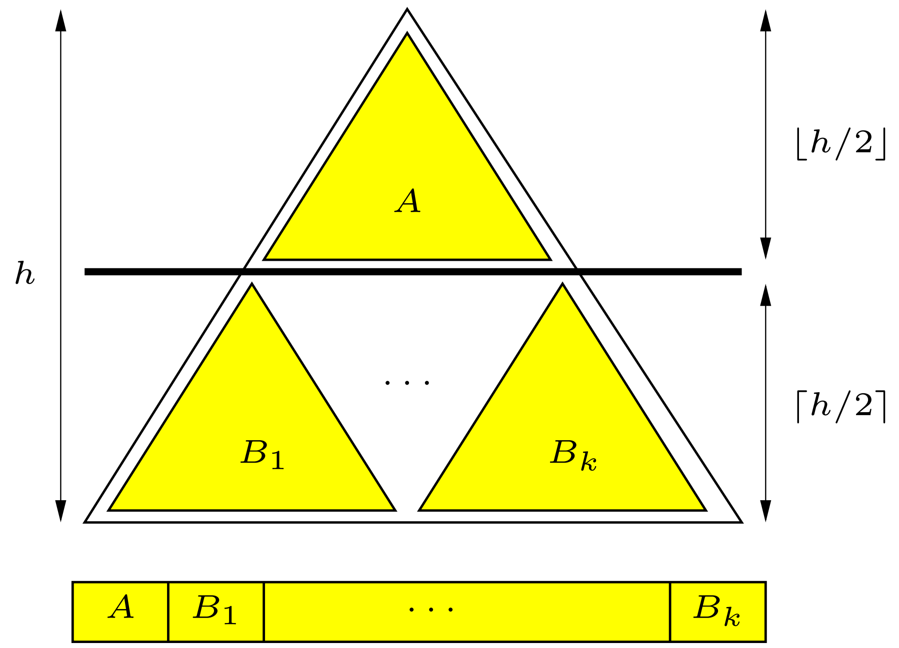




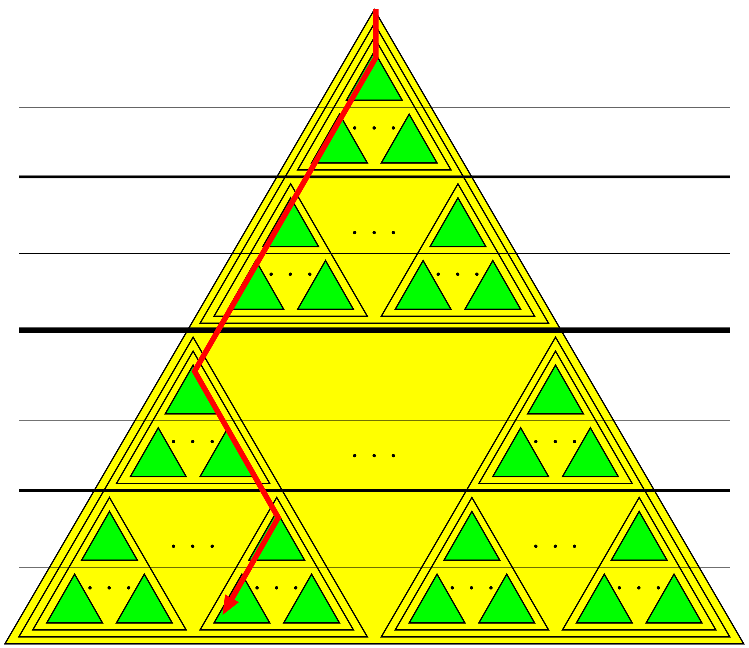
Recursive layout of binary tree

$\equiv$ van Emde Boas layout

Static Cache Oblivious Dictionary



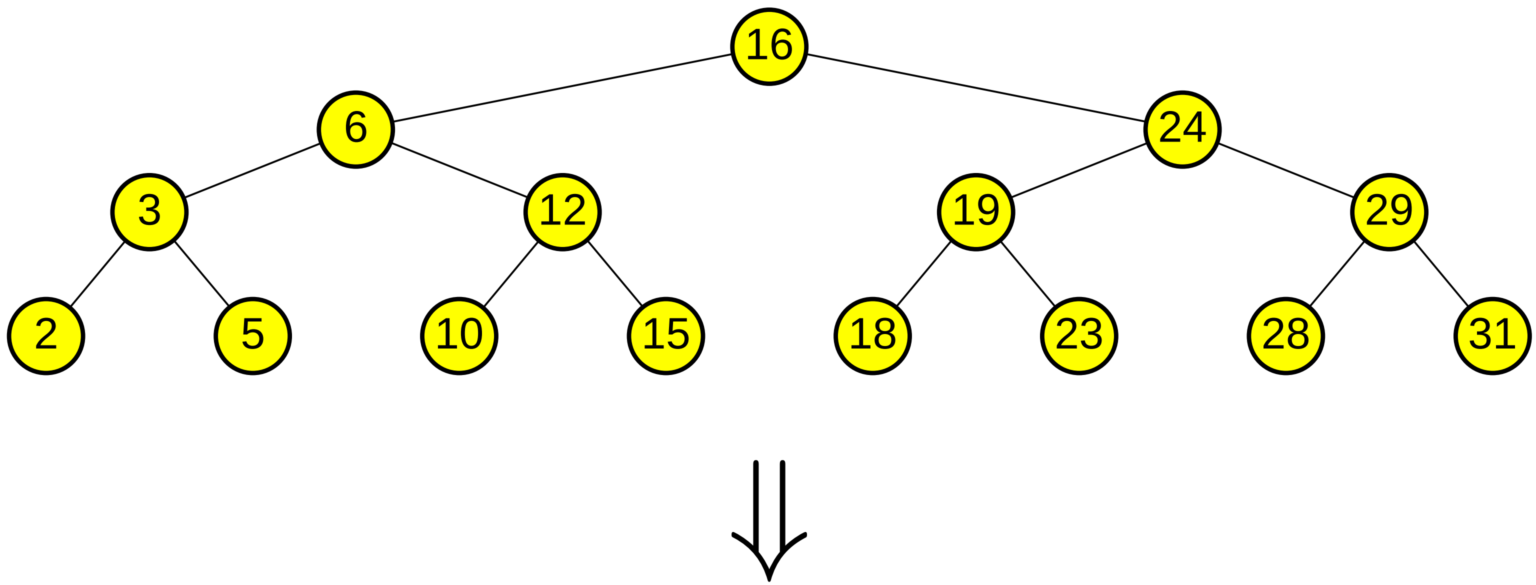
Recursive layout of binary tree
$\equiv$ van Emde Boas layout



Searches use $O(\log_B N)$ I/Os

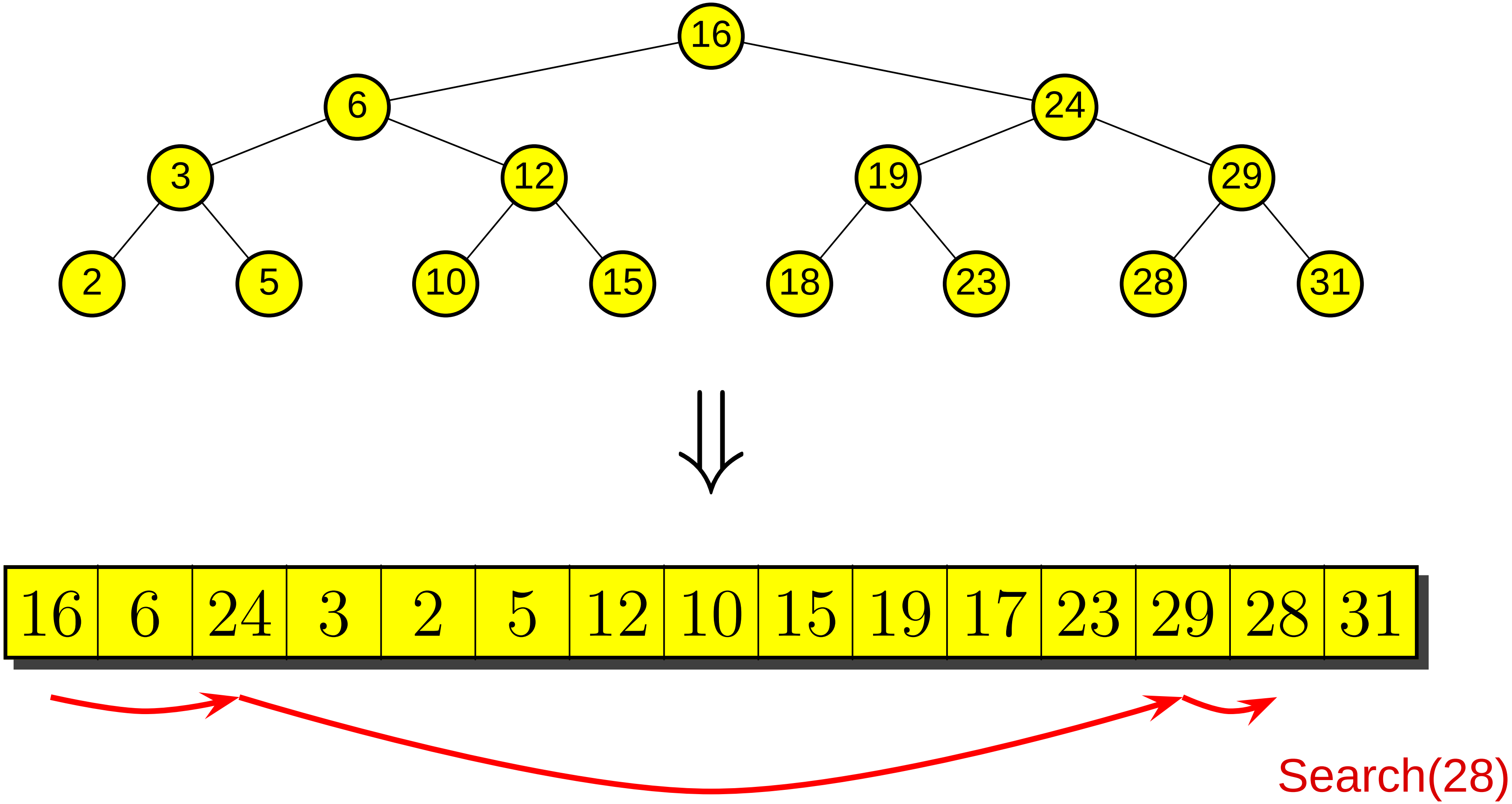
- Each green tree has height between $(\log_2 B)/2$ and $\log_2 B$
- Searches visit between $\log_B N$ and $2\log_B N$ green trees, i.e. perform at most $4\log_B N$ I/Os (misalignment)

Example : Recursive Layout



16	6	24	3	2	5	12	10	15	19	17	23	29	28	31

Example : Recursive Layout



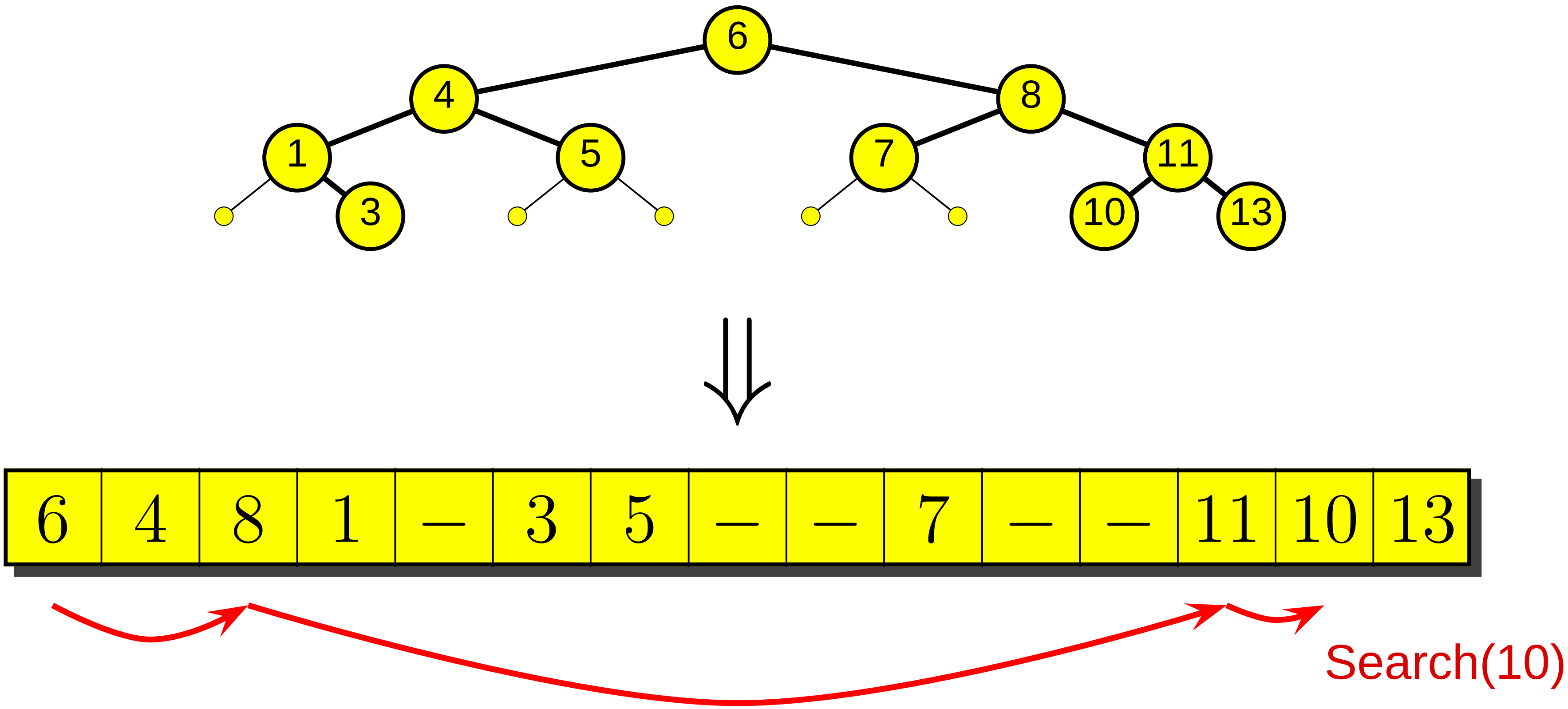
Dynamic Dictionaries

RAM model :	Balanced binary search trees, e.g. AVL-trees and red-black trees
IO model :	B-trees
Cache oblivious model :	?

Dynamic Cache Oblivious Dictionaries

Brodal and Fagerberg 2002

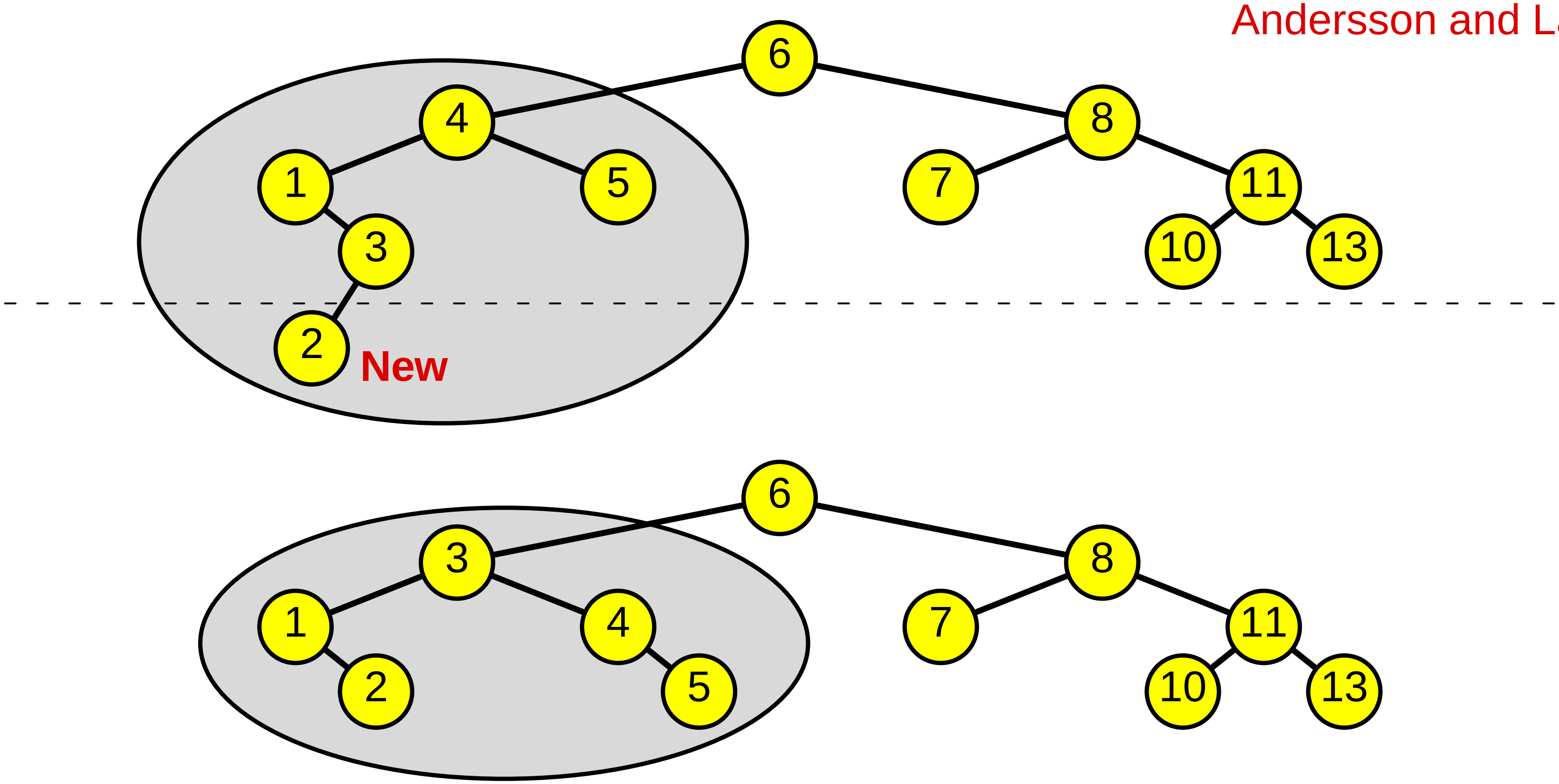
- Embed a dynamic height $\log_2 N + O(1)$ tree in a complete tree
- Static van Emde Boas layout



Dynamic Binary Trees of Small Height

Itai, Konheim and Rodeh 1981

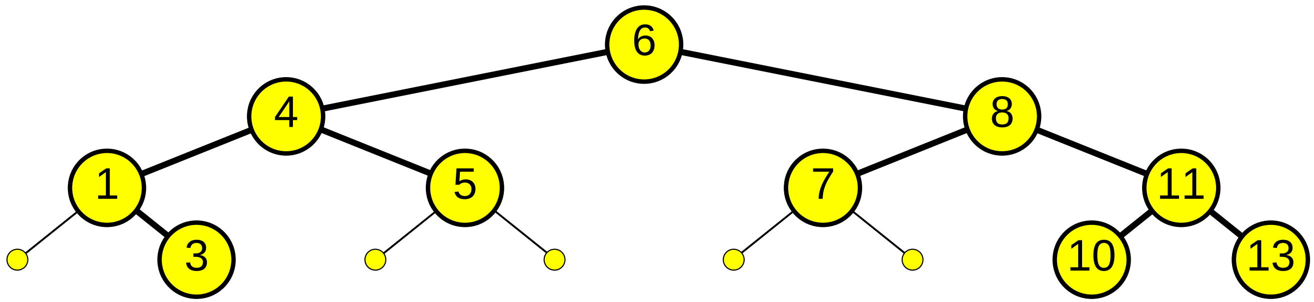
Andersson and Lai 1990



- If an insertion causes non-small height then rebuild subtree at nearest ancestor with suffi cient few descendents
- Insertions require amortized $O(\log^2 N)$ time

Dynamic Cache Oblivious Dictionaries

Brodal and Fagerberg 2002



Search	$O(\log_B N)$
Updates	$O\left(\log_B N + \frac{\log^2 N}{B}\right)$

- Updates can be improved to $O(\log_B N)$ I/Os by buckets of size $\Theta(\log_2 N)$ and one level of indirection

Lower bounds

(Comparison) RAM model : $\log n$ comparisons
(decision tree argument)

IO model : $\log_{B+1} N$ I/Os
(reduction to RAM model)

Cache oblivious model : $\log_{B+1} N$ I/Os
(follows from IO model)

$\log_2 e \cdot \log_B N \approx 1.443 \log_B N$ I/Os

Bender et al. 2003

Outline of Talk

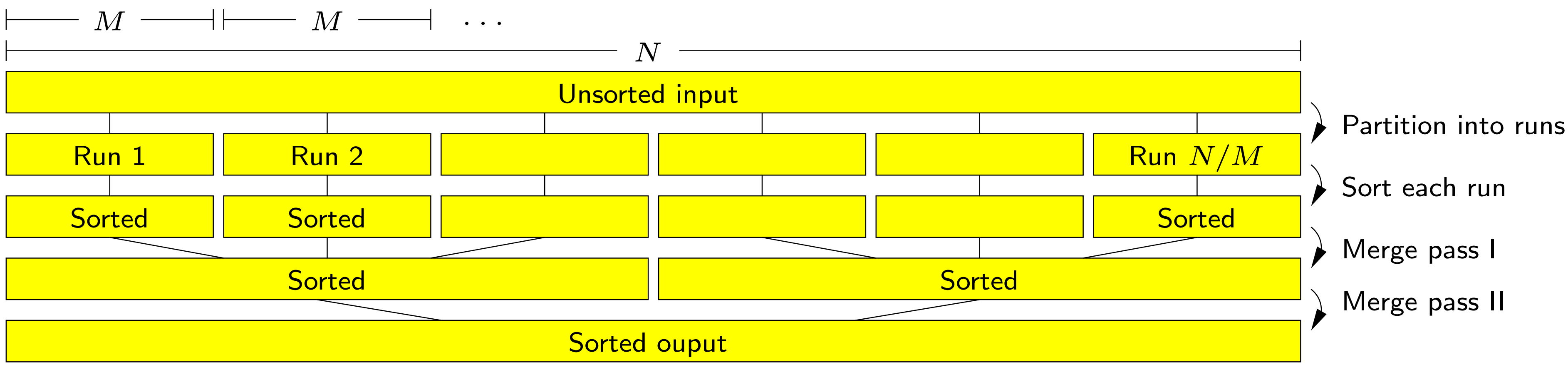
- Hardware
- Computational models
  - RAM model (Random Access Machine)
  - IO model
  - Cache oblivious model
- Binary searching and dictionaries
- ▶ Sorting
- Priority queues
- Concluding remarks

Sorting

RAM model : Binary MergeSort takes $O(N \log_2 N)$ time

IO model : $\Theta\left(\frac{M}{B}\right)$-way MergeSort achieves optimal $O(\text{Sort}(N)) = O\left(\frac{N}{B} \log_{M/B} \frac{N}{B}\right)$ I/Os

Aggarwal and Vitter 1988

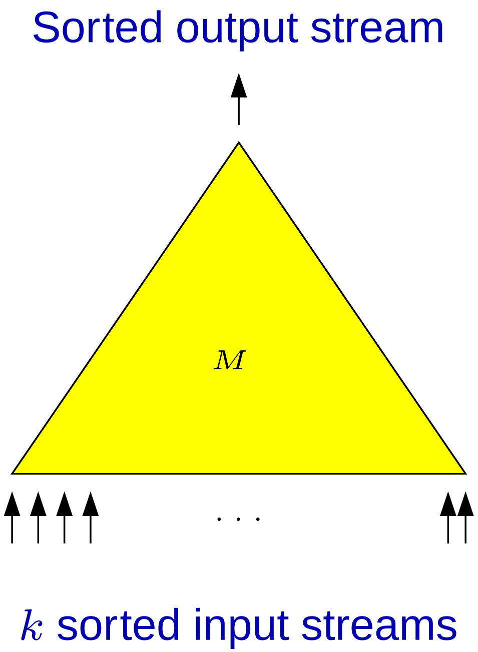


Cache oblivious : FunnelSort achieves $O(\text{Sort}(N))$ I/Os

Frigo, Leiserson, Prokop and Ramachandran 1999
Brodal and Fagerberg 2002

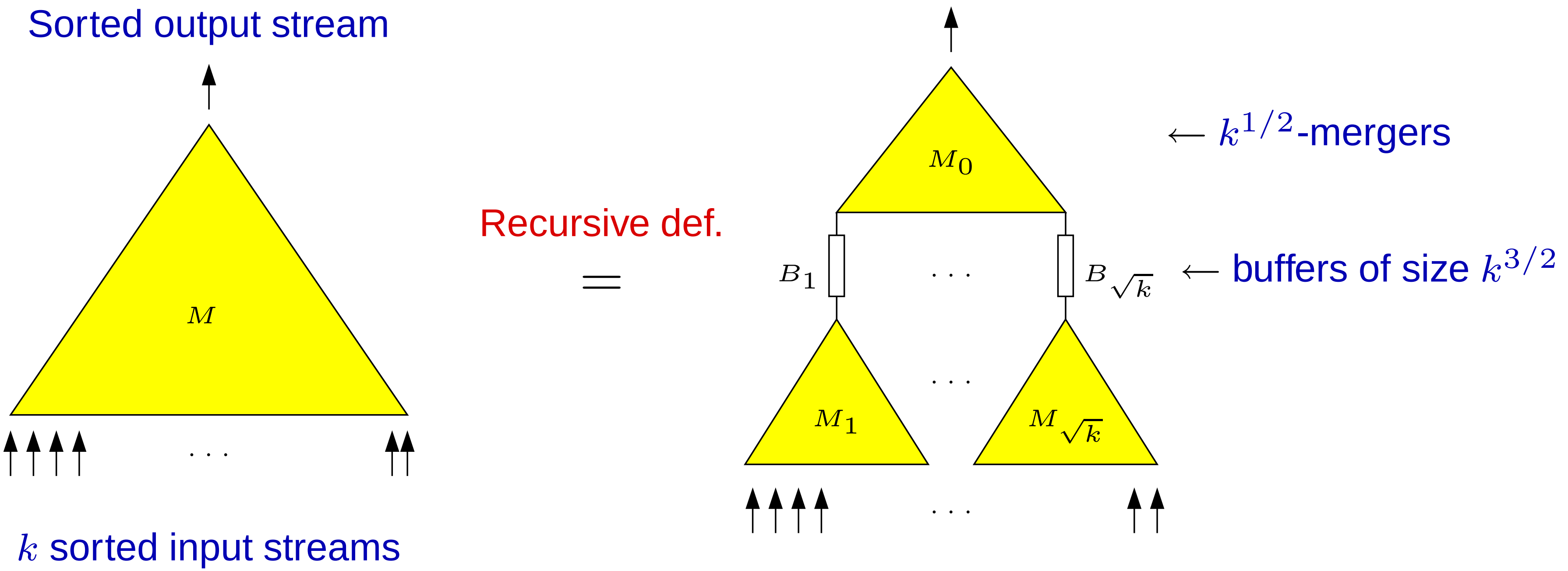
k-merger

Frigo et al., FOCS'99



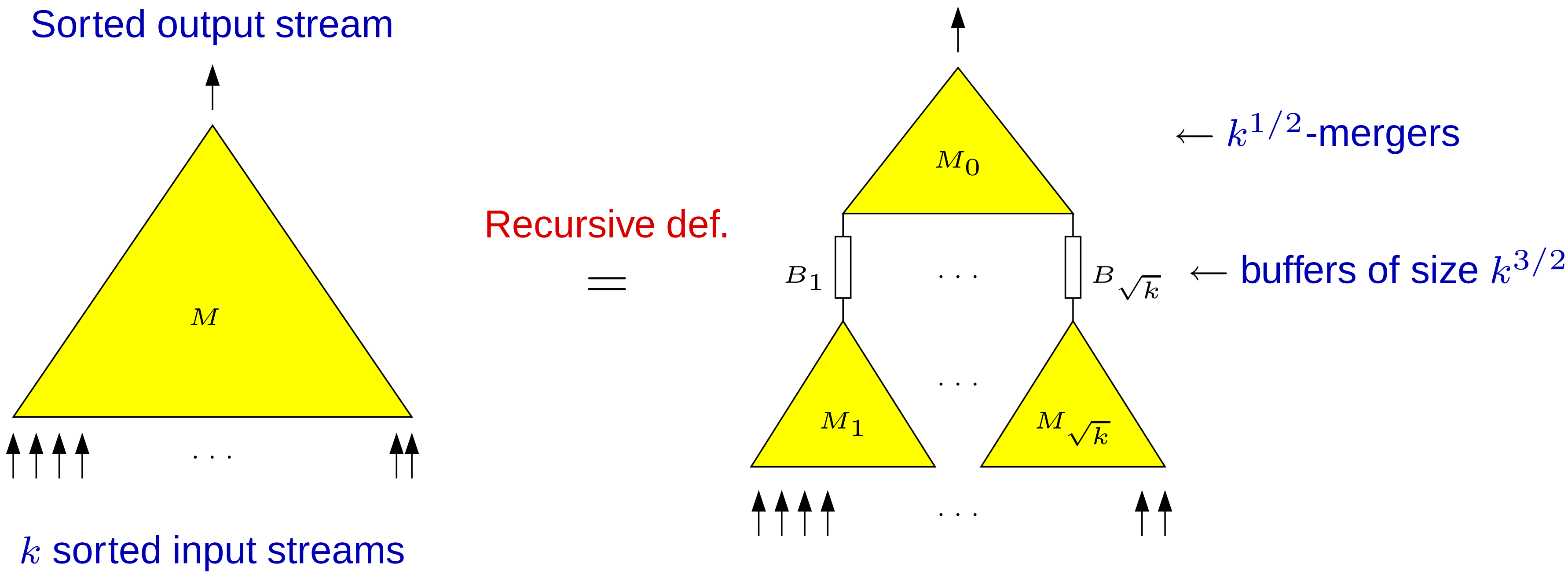
k-merger

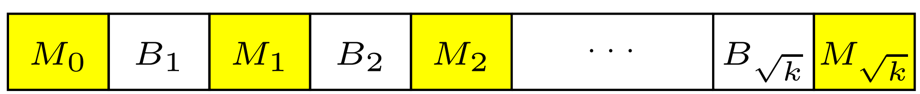
Frigo et al., FOCS'99



k-merger

Frigo et al., FOCS'99

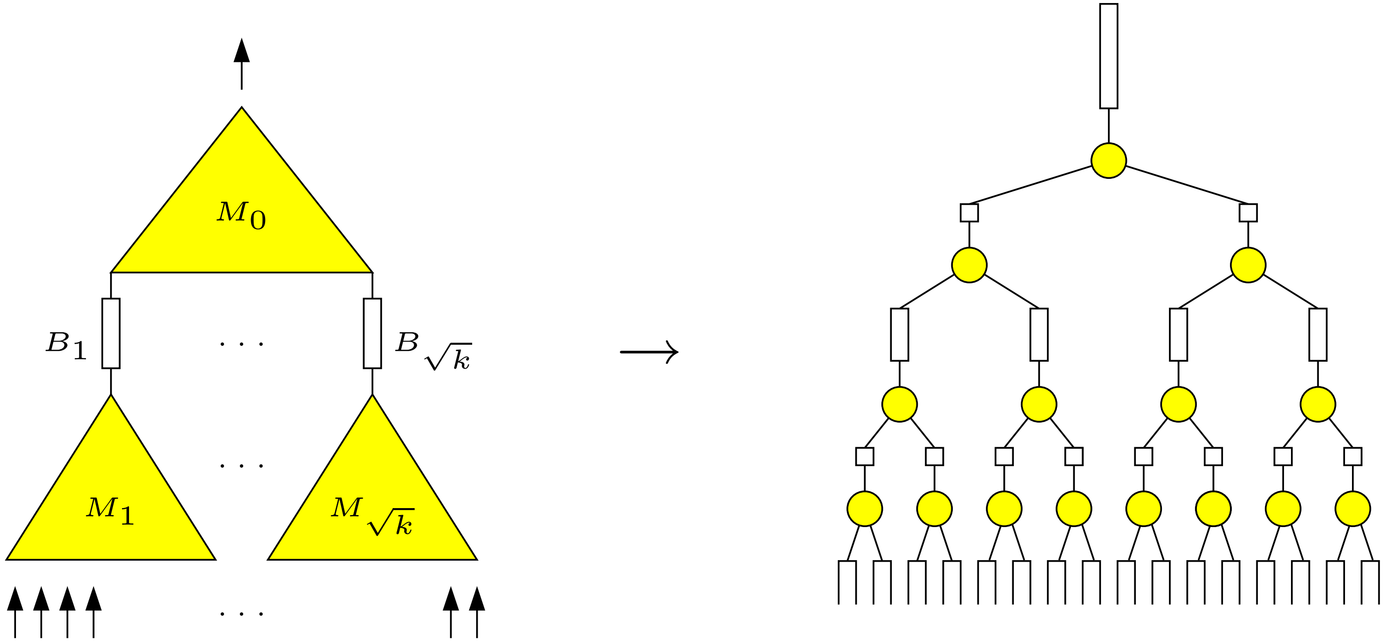




Recursive Layout

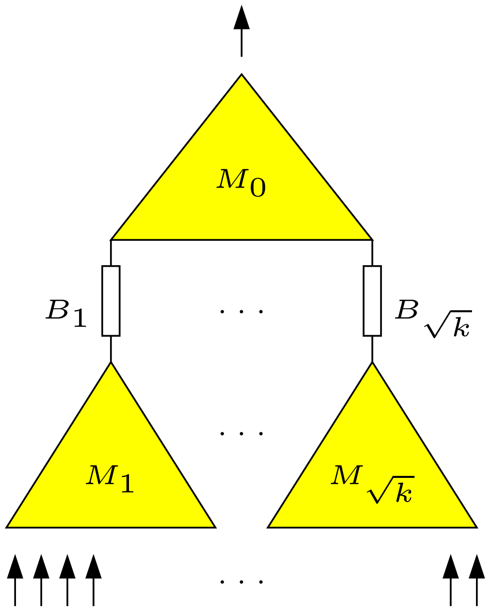
Lazy k-merger

Brodal and Fagerberg 2002

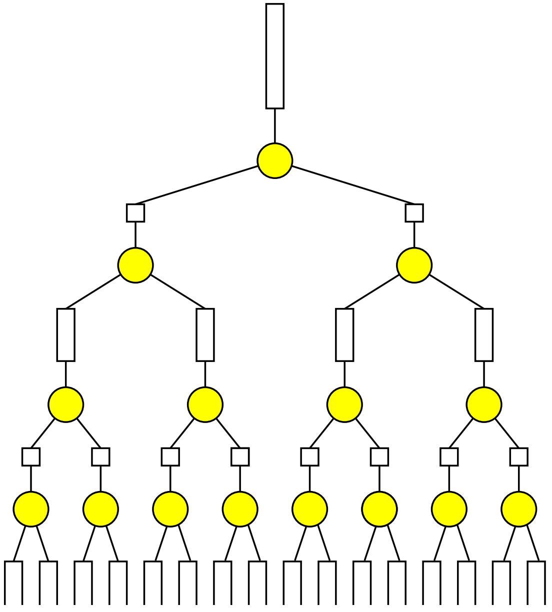


Lazy k-merger

Brodal and Fagerberg 2002



$\longrightarrow$

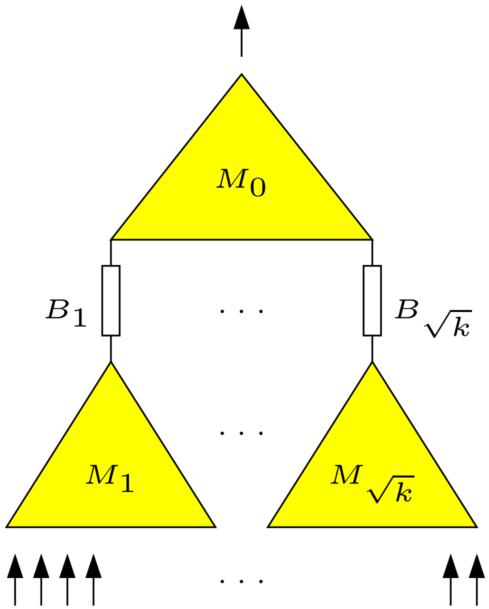


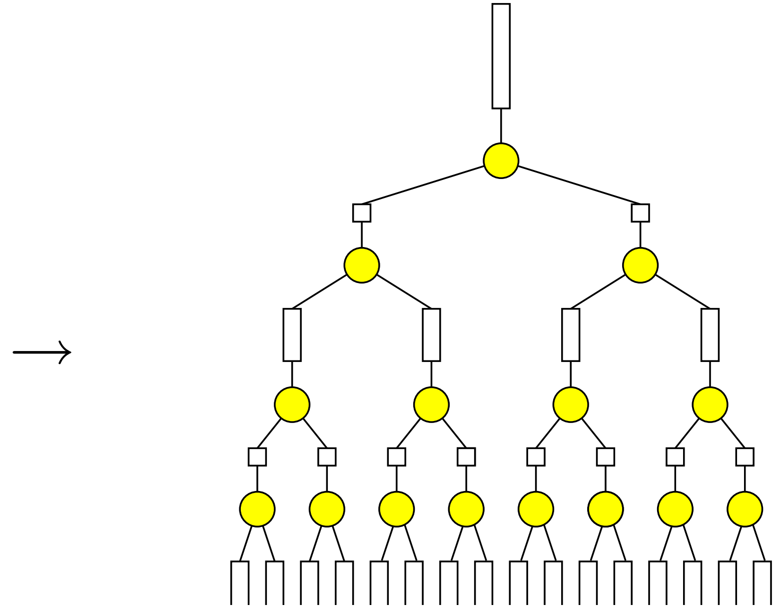
Procedure **Fill**(v)

```
while out-buffer not full
    if left in-buffer empty
        Fill(left child)
    if right in-buffer empty
        Fill(right child)
    perform one merge step
```

Lazy k-merger

Brodal and Fagerberg 2002





Procedure **Fill**(v)

while out-buffer not full
  if left in-buffer empty
    Fill(left child)
  if right in-buffer empty
    Fill(right child)
  perform one merge step

Lemma

If $M \geq B^2$ and output buffer has size k^3 then $O(\frac{k^3}{B}\log_M(k^3)+k)$ I/Os are done during an invocation of **Fill**(root).

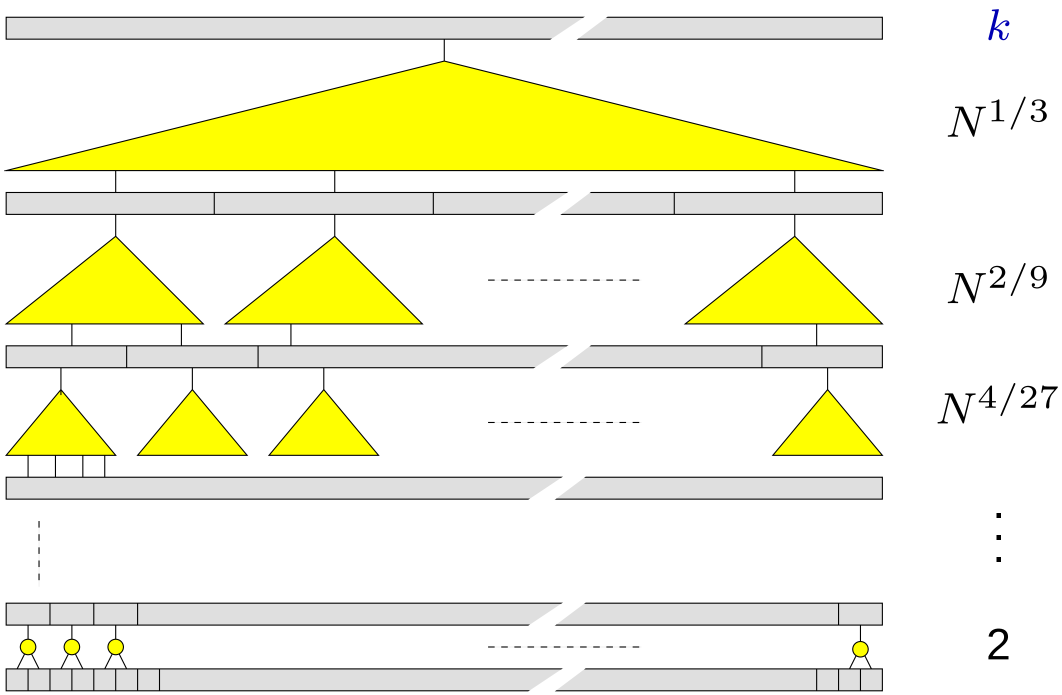
FunnelSort

Brodal and Fagerberg 2002
Frigo, Leiserson, Prokop and Ramachandran 1999

Divide input in $N^{1/3}$ segments of size $N^{2/3}$

Recursively **MergeSort** each segment

Merge sorted segments by an $N^{1/3}$-merger



FunnelSort

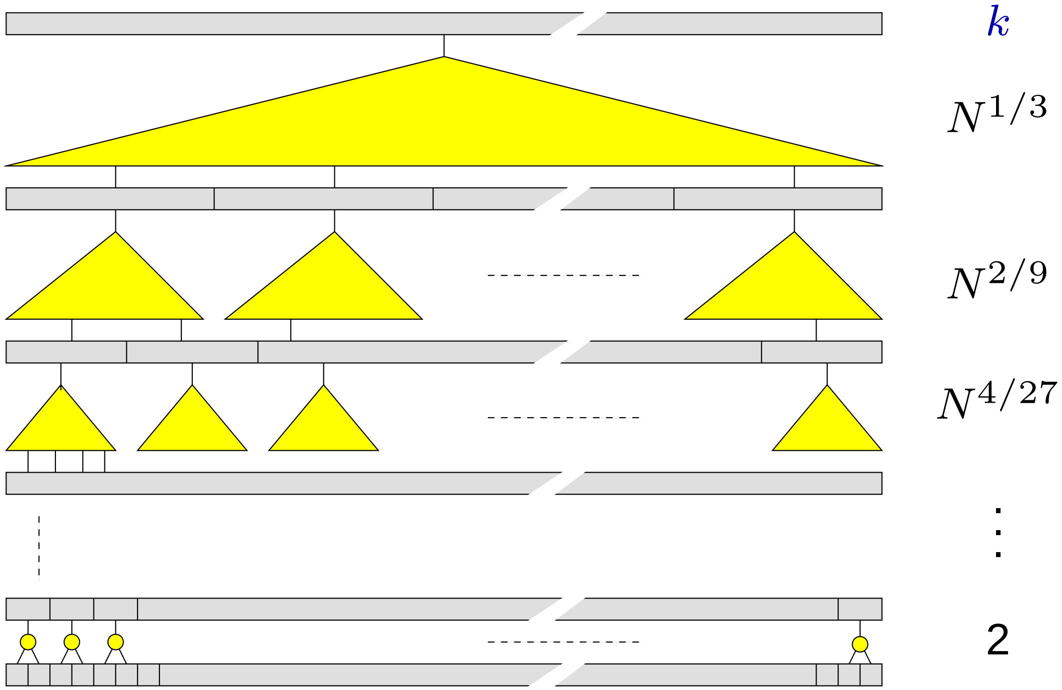
Brodal and Fagerberg 2002

Frigo, Leiserson, Prokop and Ramachandran 1999

Divide input in $N^{1/3}$ segments of size $N^{2/3}$

Recursively **MergeSort** each segment

Merge sorted segments by an $N^{1/3}$-merger



Theorem Provided $M \geq B^2$ (tall cache assumption), FunnelSort performs optimal $O(\mathrm{Sort}(N))$ I/Os

Computational Geometry

Brodal and Fagerberg 2002

Cache oblivious $O(\mathrm{Sort}(N))$ distribution sweeping algorithms for

- Maxima for point set (3D)
- Measure of a set of axis-parallel rectangles (2D)
- Visibility of non-intersecting line segments from a point (2D)
- All nearest neighbors for point set (2D)

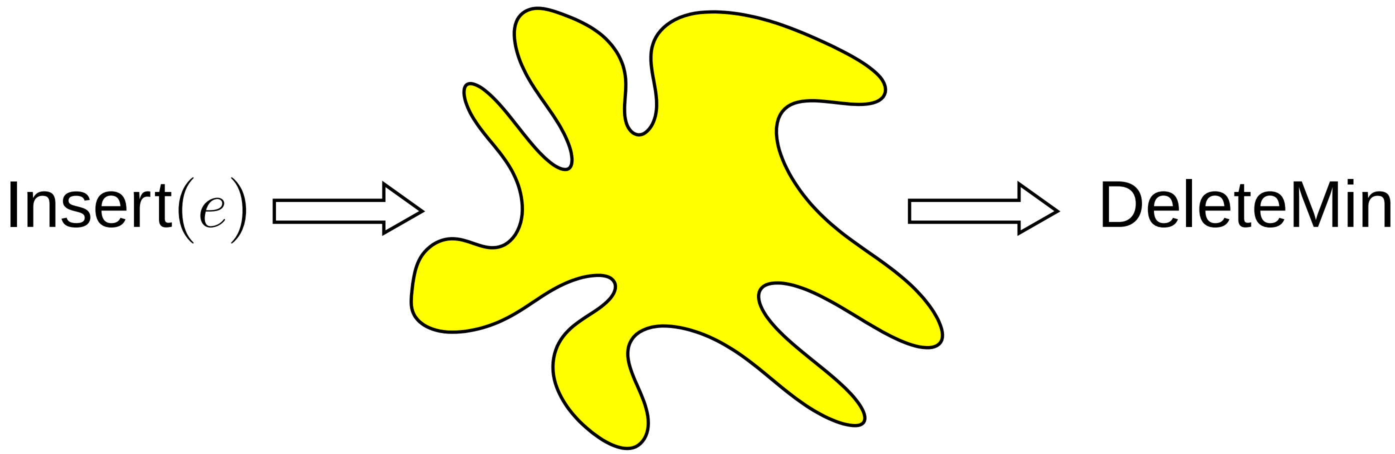
Cache oblivious $O(\mathrm{Sort}(N) + \frac{\mathrm{output}}{B})$ algorithms for

- Orthogonal line segment intersection reporting (2D)
- Batched orthogonal range queries on point set (2D)
- Pairwise intersections of axis-parallel rectangles (2D)

Outline of Talk

- Hardware
- Computational models
  - RAM model (Random Access Machine)
  - IO model
  - Cache oblivious model
- Binary searching and dictionaries
- Sorting
- ▶ Priority queues
- Concluding remarks

Priority Queues

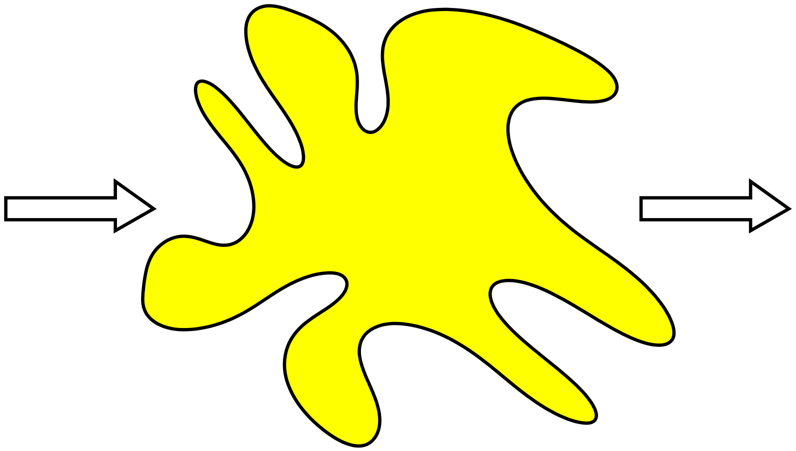


Classic RAM:

- Heap: $O(\log_2 n)$ time — Williams 1964

Priority Queues

Insert(e)

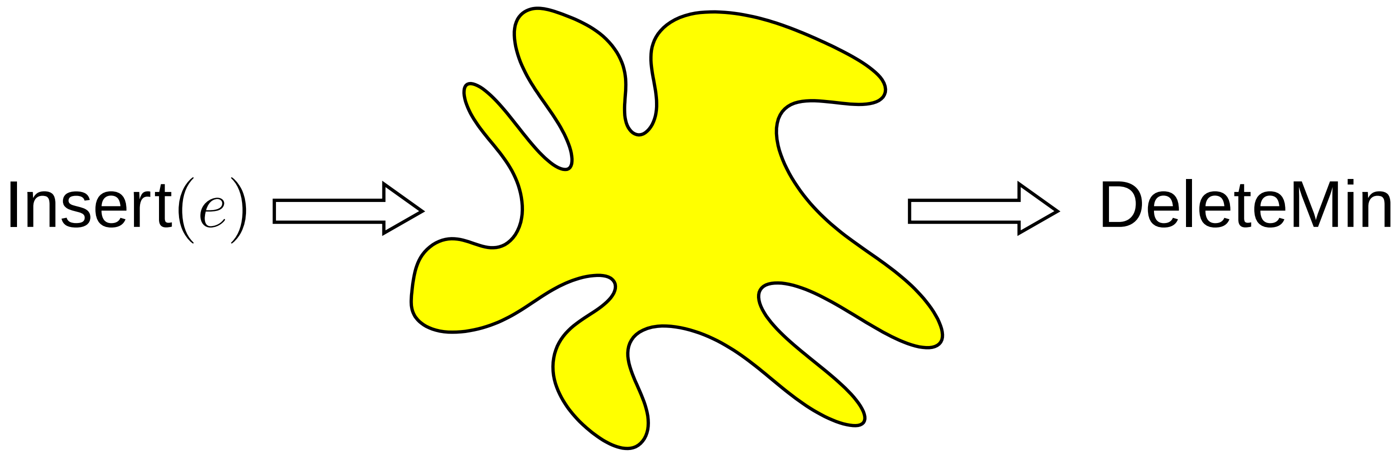


DeleteMin

Classic RAM:

- Heap: $O(\log_2 n)$ time, $O\left(\log_2 \frac{N}{M}\right)$ I/Os   Williams 1964

Priority Queues



Classic RAM:

- Heap: $O(\log_2 n)$ time, $O\left(\log_2 \frac{N}{M}\right)$ I/Os — Williams 1964

I/O model:

- Buffer tree: $O\left(\frac{1}{B} \log_{M/B} \frac{N}{B}\right) = O\left(\frac{\text{Sort}(N)}{N}\right)$ I/Os — Arge 1995

Cache-Oblivious Priority Queues

- $O\left(\frac{1}{B}\log_{M/B}\frac{N}{B}\right)$ I/Os

  Arge, Bender, Demaine, Holland-Minkley and Munro 2002

  – Uses sorting and selection as subroutines

  – Requires tall cache assumption, $M \geq B^2$

- Funnel heap

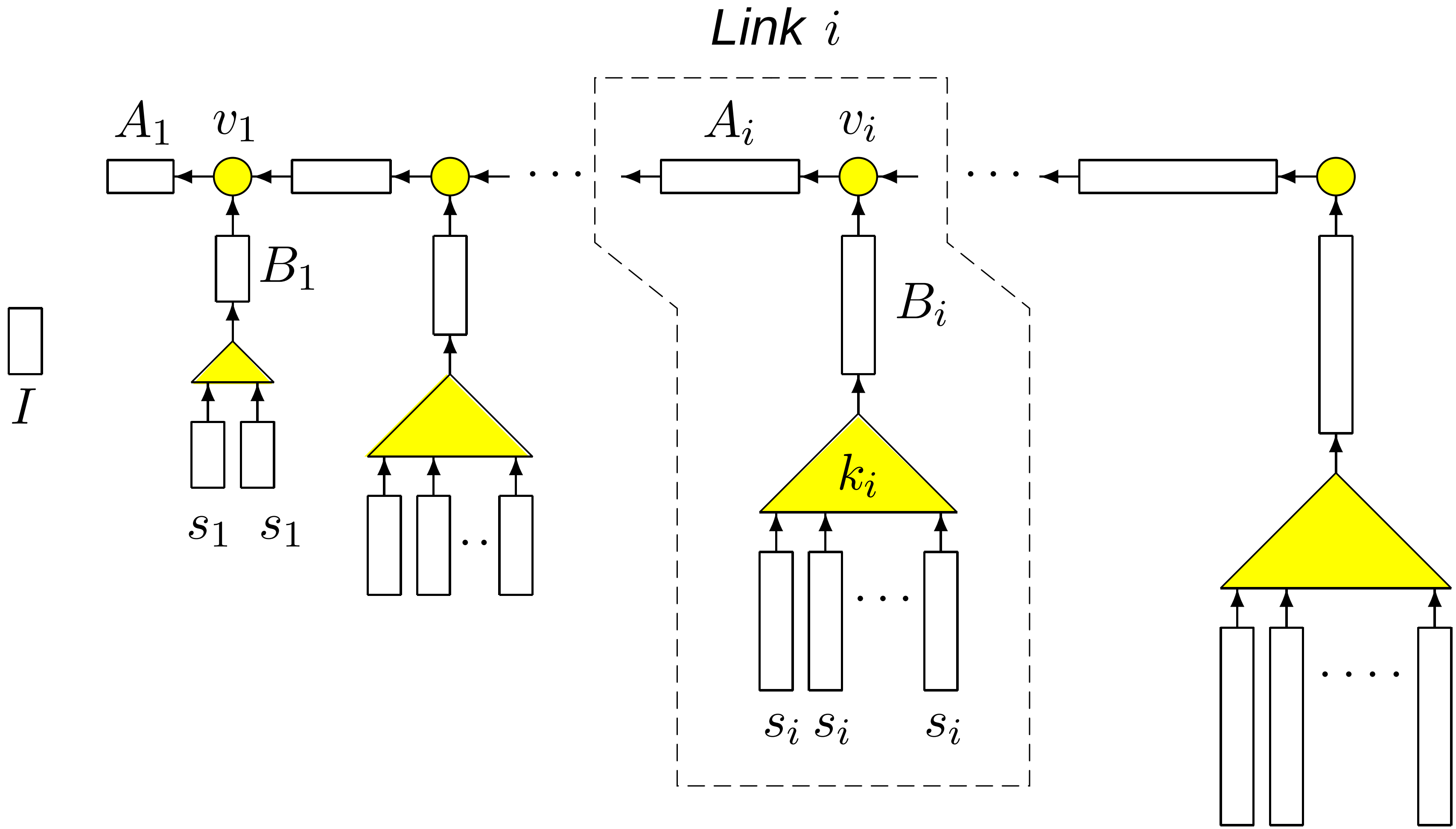
  Brodal and Fagerberg 2002

  – Uses only binary merging

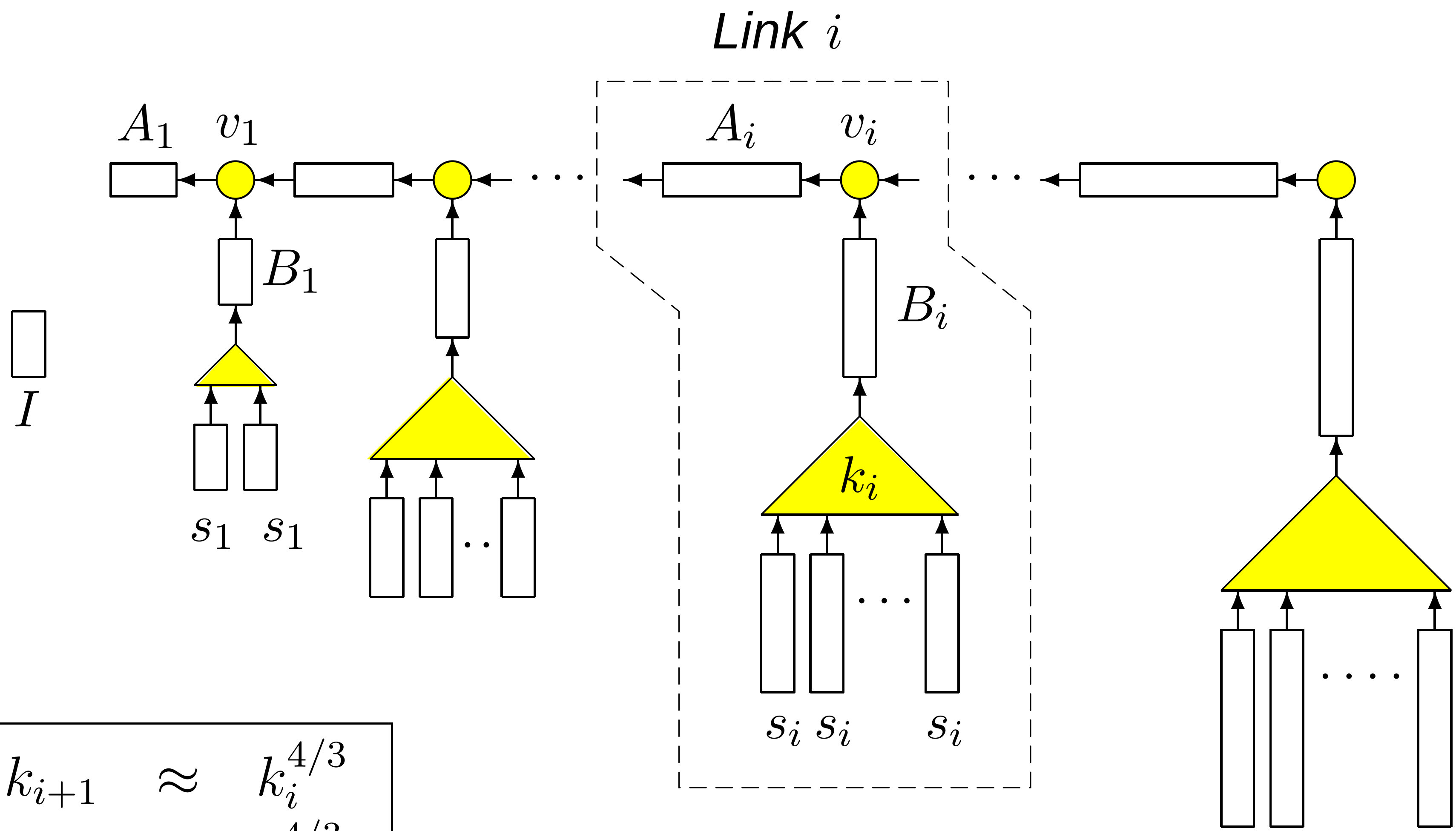
  – Profi le adaptive, i.e. $O\left(\frac{1}{B}\log_{M/B}\frac{N_i}{B}\right)$ I/Os

  N_i is either the size profi le, max depth profi le, or #insertions during the lifetime of the ith inserted element

The Priority Queue

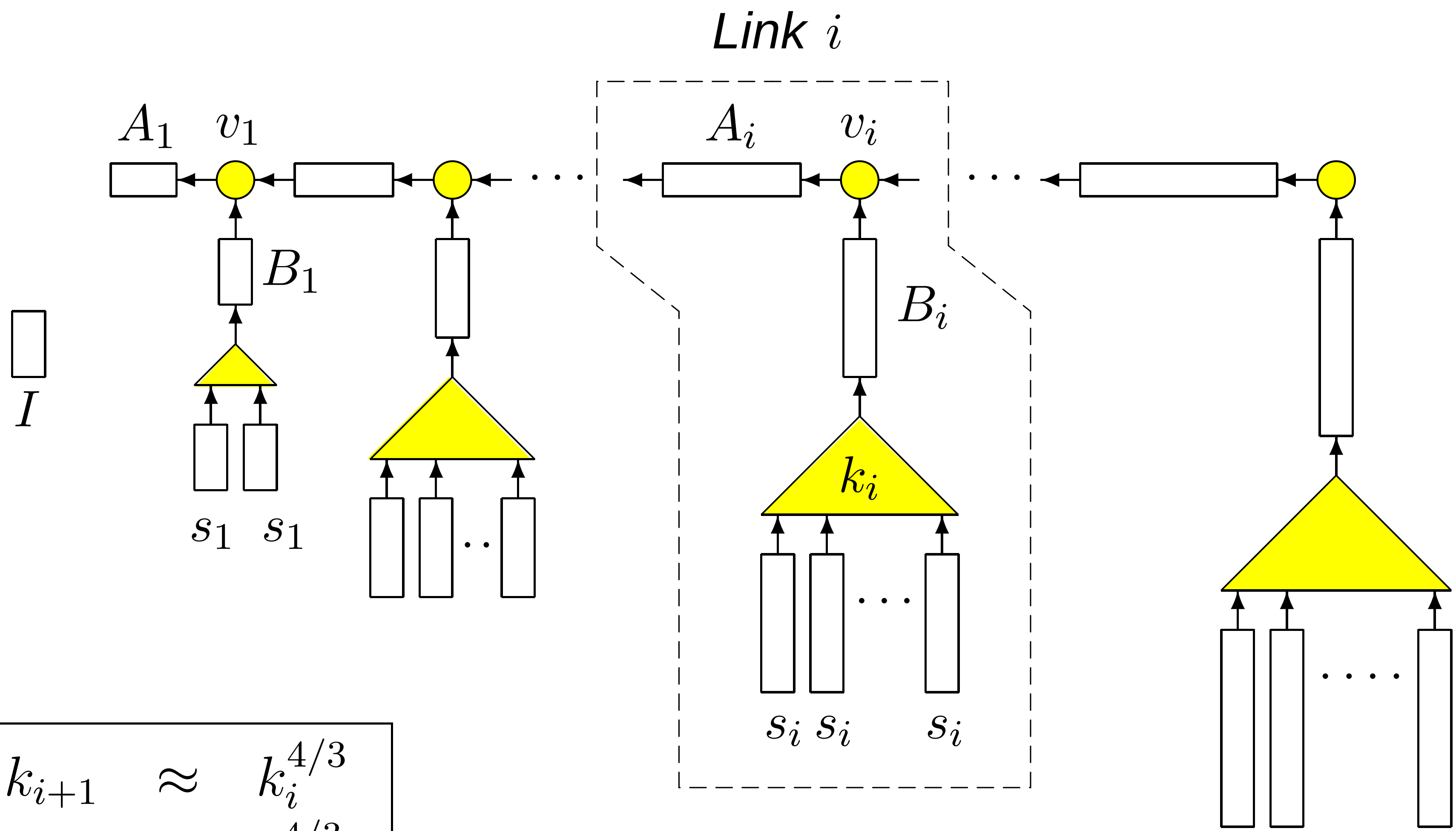


The Priority Queue



$$k_{i+1} \approx k_i^{4/3}$$
$$s_{i+1} \approx s_i^{4/3}$$
$$k_i \approx s_i^{1/3}$$

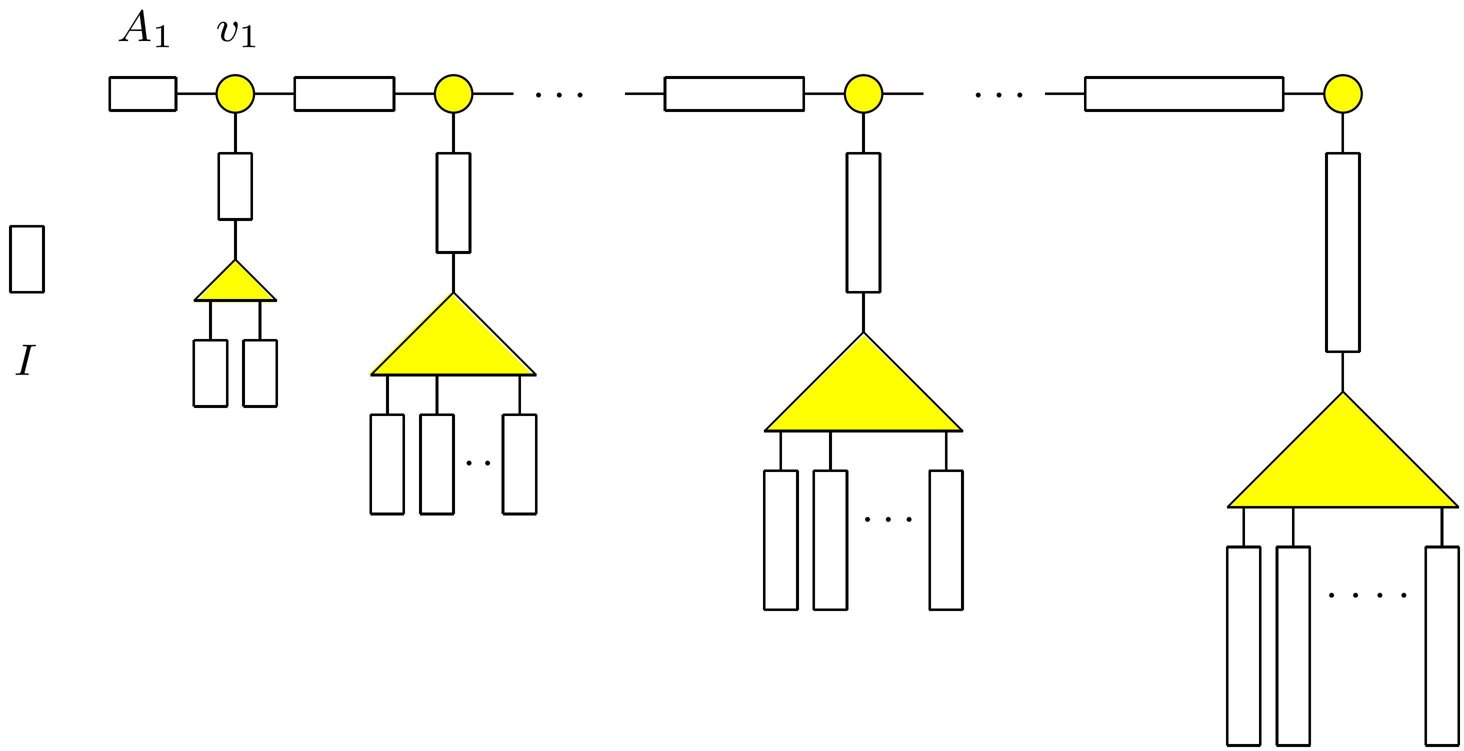
The Priority Queue



$$k_{i+1} \approx k_i^{4/3}$$
$$s_{i+1} \approx s_i^{4/3}$$
$$k_i \approx s_i^{1/3}$$

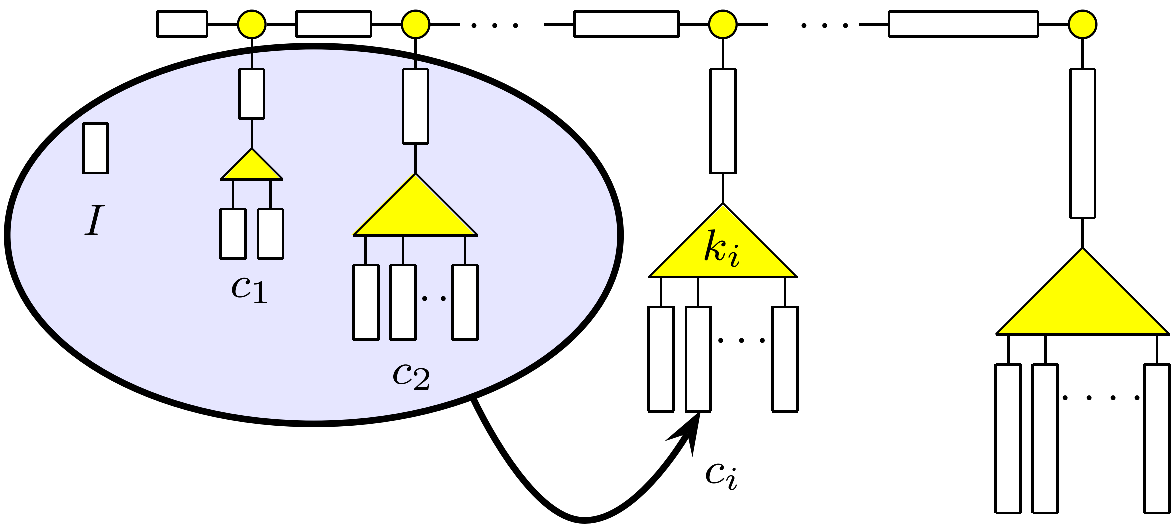
In total: A single binary merge tree

Operations — DeleteMin



- If A_1 is empty, call Fill(v_1)
- Search I and A_1 for minimum element

Operations — Insert

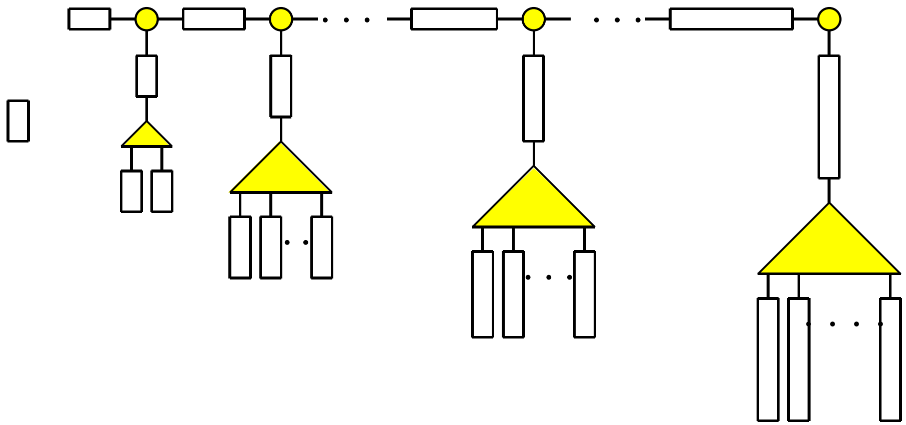


- Insert in I
- If I overflows, call Sweep(i) for first i where $c_i \leq k_i$

Sweep $\approx$ addition of one to number $c_1 c_2 .. c_i .. c_{\max}$

$$s_i = s_1 + \sum_{j=1}^{i-1} k_j s_j$$

Analysis



We can prove:

- Number N of insertions performed: $s_{i_{\max}} \leq N$
- Number of I/Os per Insert for link i: $O\left(\frac{1}{B}\log_{M/B} s_i\right)$
- By the doubly-exponentially growth of s_i,
  the total number of I/Os per Insert is

$$O\left(\sum_{k=0}^{\infty} \frac{1}{B} \log_{M/B} N^{(3/4)^k}\right) = O\left(\frac{\mathrm{Sort}(N)}{N}\right)$$

- DeleteMin is amortized for free

Outline of Talk

- Hardware
- Computational models
  - RAM model (Random Access Machine)
  - IO model
  - Cache oblivious model
- Binary searching and dictionaries
- Sorting
- Priority queues
- ▶ Concluding remarks

Some Cache-Oblivious Results

- Scanning $\Rightarrow$ stack, queue, median finding,. . . .
- Sorting, matrix multiplication, FFT

  Frigo, Leiserson, Prokop, Ramachandran, FOCS'99
- Cache oblivious search trees

  Prokop 99

  Bender, Demaine, Farach-Colton, FOCS'00

  Rahman, Cole, Raman, WAE'01

  Bender, Duan, Iacono, Wu and Brodal, Fagerberg, Jacob, SODA'02
- Priority queue and graph algorithms

  Arge, Bender, Demaine, Holland-Minkley, Munro, STOC'02

  Brodal, Fagerberg, ISAAC'02
- Computational geometry

  Brodal, Fagerberg, ICALP'02

  Bender, Cole, Raman, ICALP'02
- Scanning dynamic sets

  Bender, Cole, Demaine, Farach-Colton, ESA'02

Cache Oblivious Technics

- Scanning
- Sorting
- Recursion
- Recursive layout (van Emde Boas layout)
- Merging (FunnelSort, distribution sweeping, FunnelHeap)

Conclusions

- Cache oblivious model : Simpel and general
- Algorithms exist for many problems
  - stacs, queues, dictionaries, priority queues, sorting, selection, permuting, matrix multiplicataion, FFT, graph algorithms, computational geometry...
- Limitations
  - searching costs a factor $\log_2 e$
  - sorting and priority queues requires a tall cache

Brodal and Fagerberg 2003

Open problems

- Other algorithms ...
- Cache obliviousness *vs* parallel disks ?
- Implementations and experiments ?
- Libraries ?
- ...

References

- **The Cost of Cache-Oblivious Searching**, Michael A. Bender, Gerth Stølting Brodal, Rolf Fagerberg, Dongdong Ge, Simai He, Haodong Hu, John Iacono, and Alejandro López-Ortiz. Submitted.
- **On the Limits of Cache-Obliviousness**, Gerth Stølting Brodal and Rolf Fagerberg. To appear in *Proc. 35th Annual ACM Symposium on Theory of Computing*, 2003.
- **Funnel Heap - A Cache Oblivious Priority Queue**, Gerth Stølting Brodal and Rolf Fagerberg. In *Proc. 13th Annual International Symposium on Algorithms and Computation*, volume 2518 of *Lecture Notes in Computer Science*, pages 219-228. Springer Verlag, Berlin, 2002.
- **Cache Oblivious Distribution Sweeping**, Gerth Stølting Brodal and Rolf Fagerberg. In *Proc. 29th International Colloquium on Automata, Languages, and Programming*, volume 2380 of *Lecture Notes in Computer Science*, pages 426-438. Springer Verlag, Berlin, 2002.
- **Cache-Oblivious Search Trees via Trees of Small Height**, Gerth Stølting Brodal, Rolf Fagerberg, and Riko Jacob. In *Proc. 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 39-48, 2002.