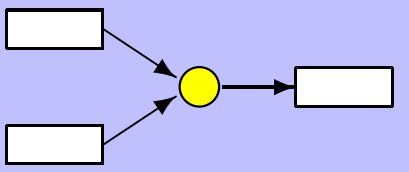
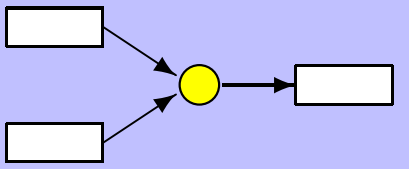
Merge Trees

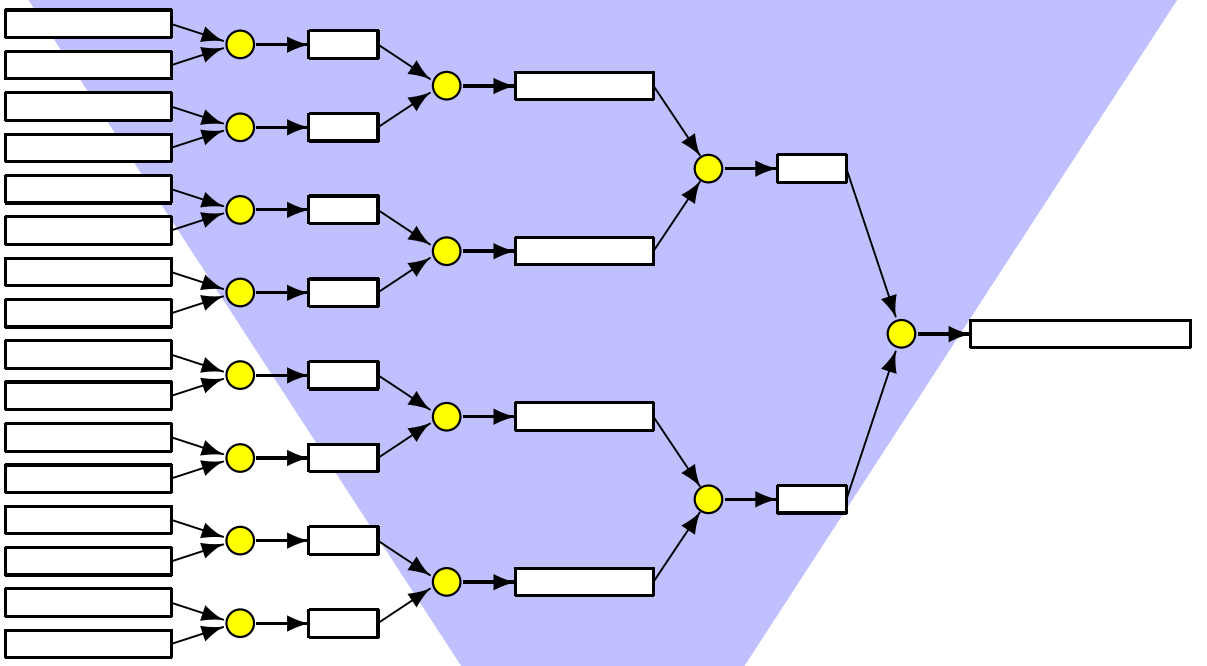


Binary merger

Merge Trees



Binary merger

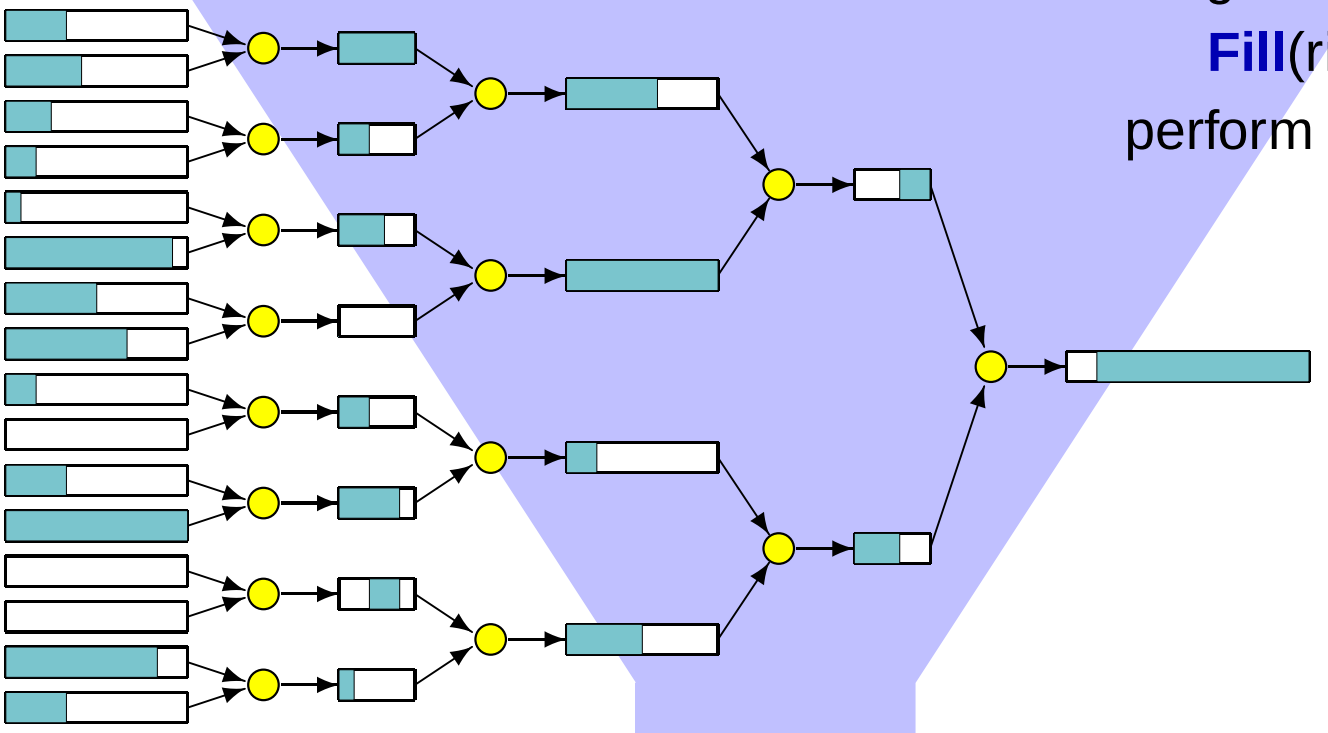


Merge tree

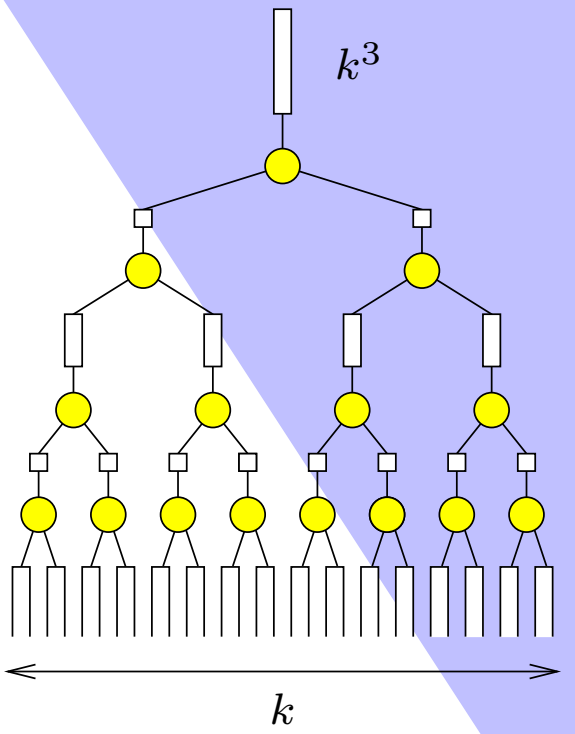
Merging Algorithm

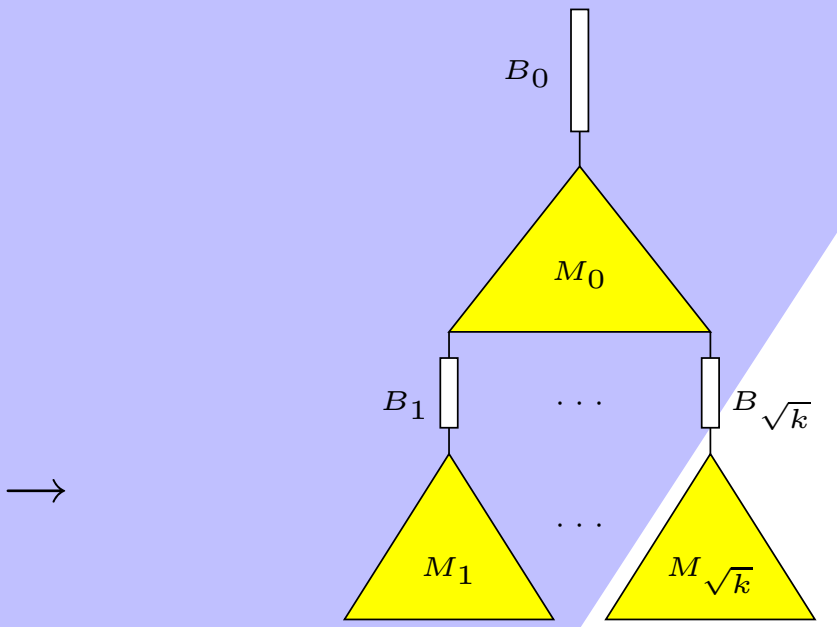
Fill(v)

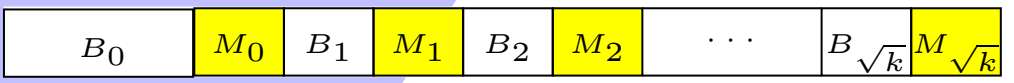
while out-buffer not full
  if left in-buffer empty
    Fill(left child)
  if right in-buffer empty
    Fill(right child)
  perform one merge step



k - merger







Recursive memory layout

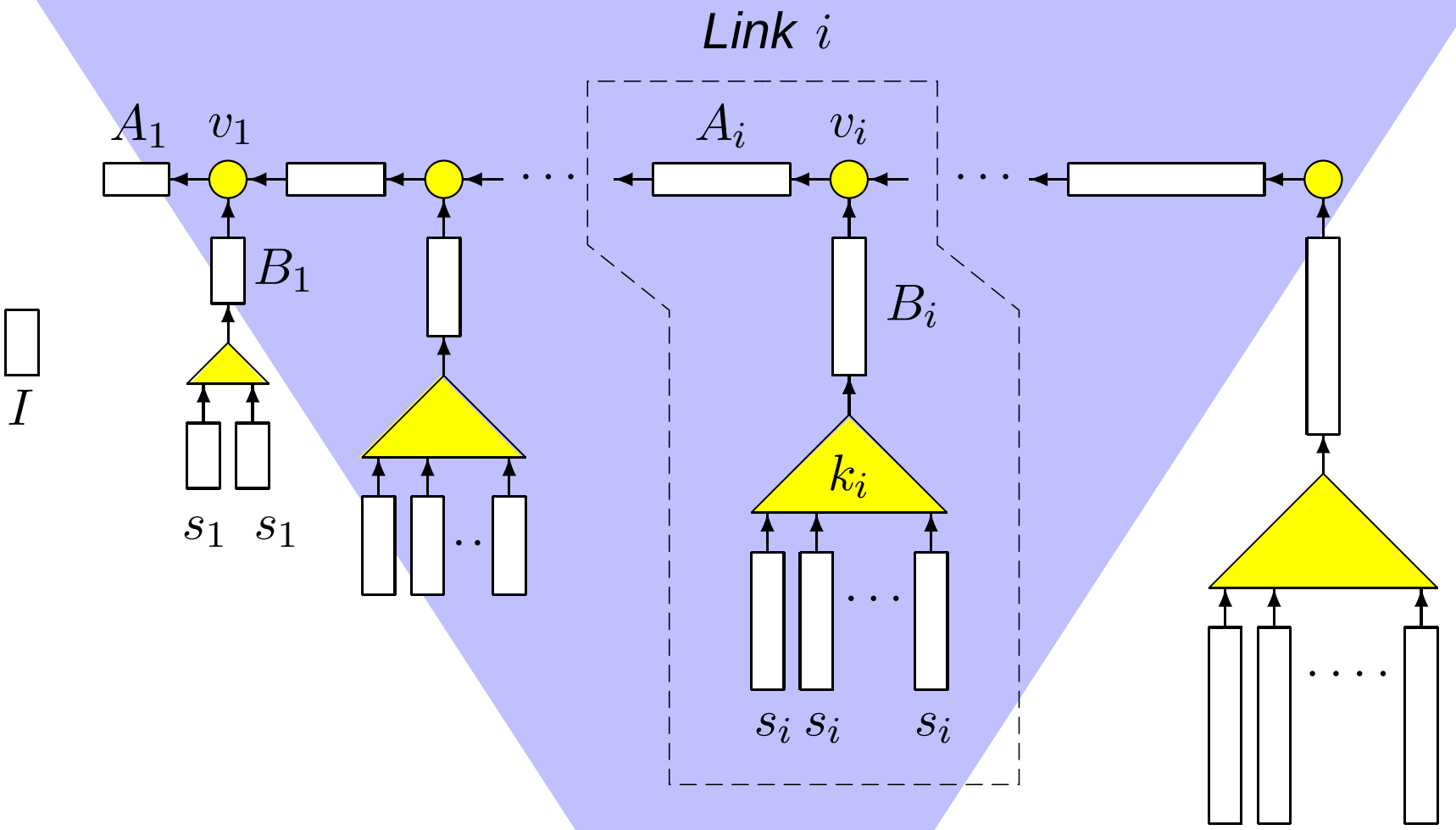
Recursive defi nition of buffer sizes

Lemma: $O(\frac{k^3}{B}\log_M(k^3)+k)$ I/Os per invocation (if $M \geq B^2$)

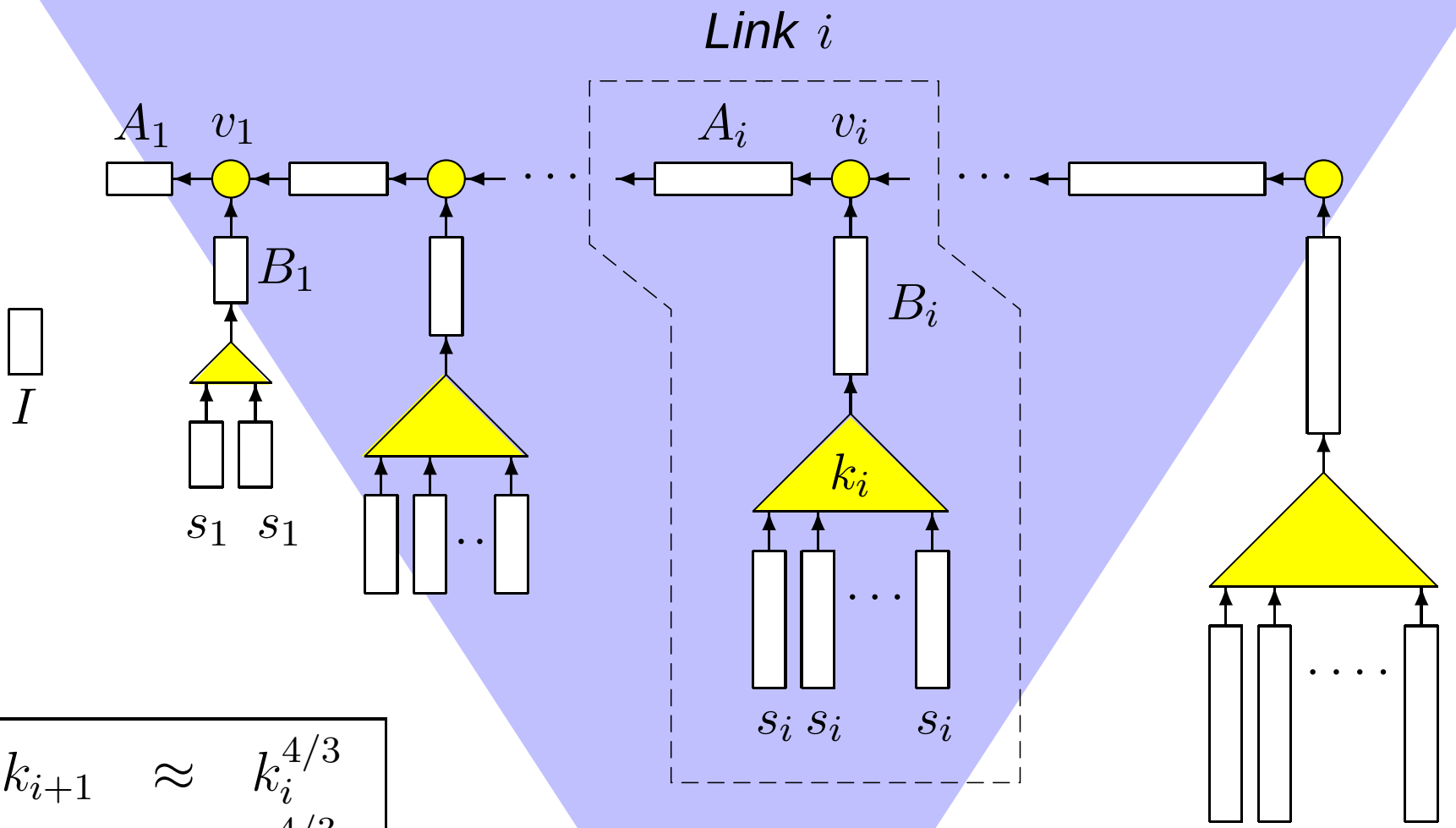
Frigo, Leiserson, Prokop, Ramachandran 1999

Brodal, Fagerberg 2002

The Priority Queue

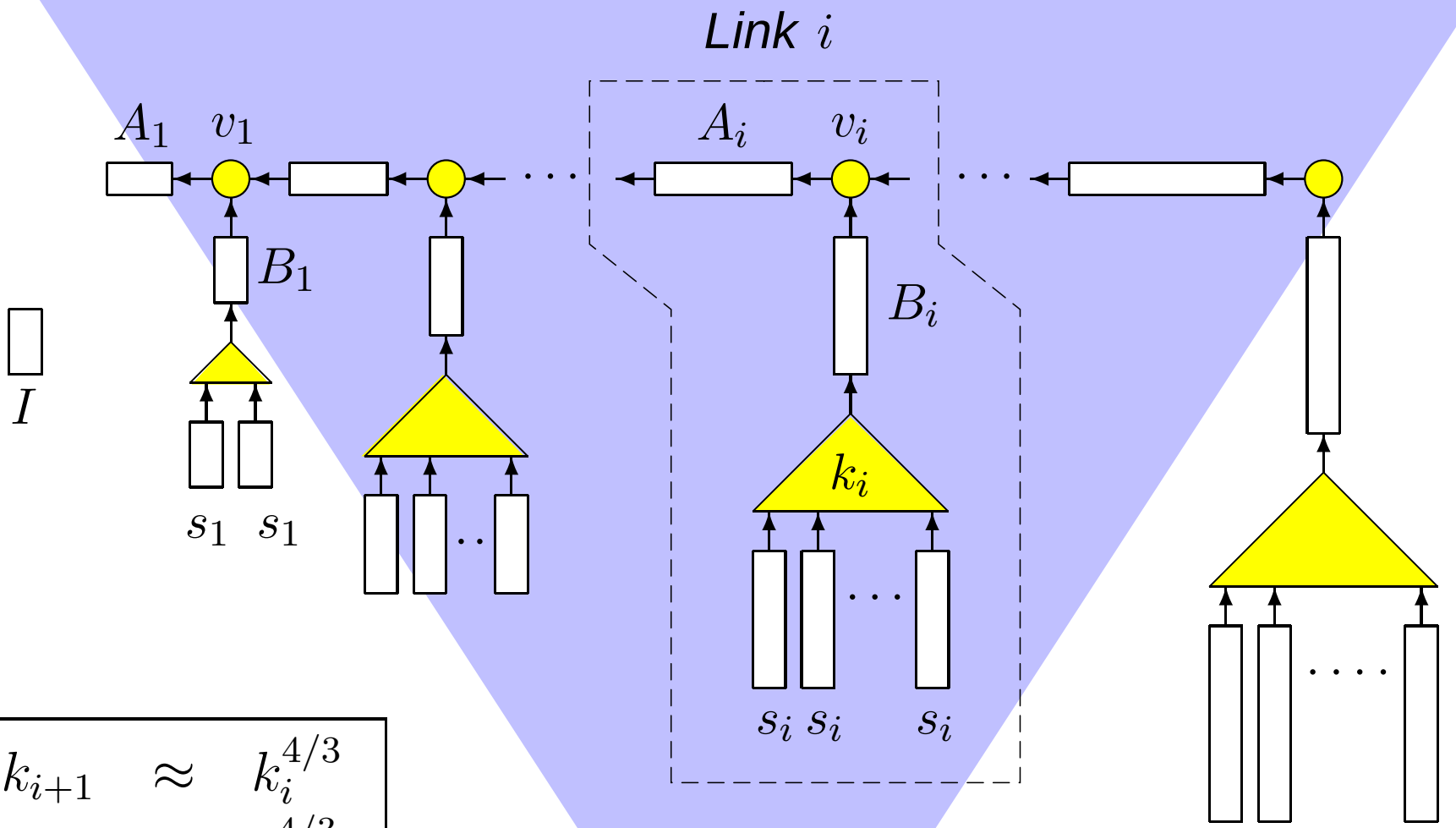


The Priority Queue



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

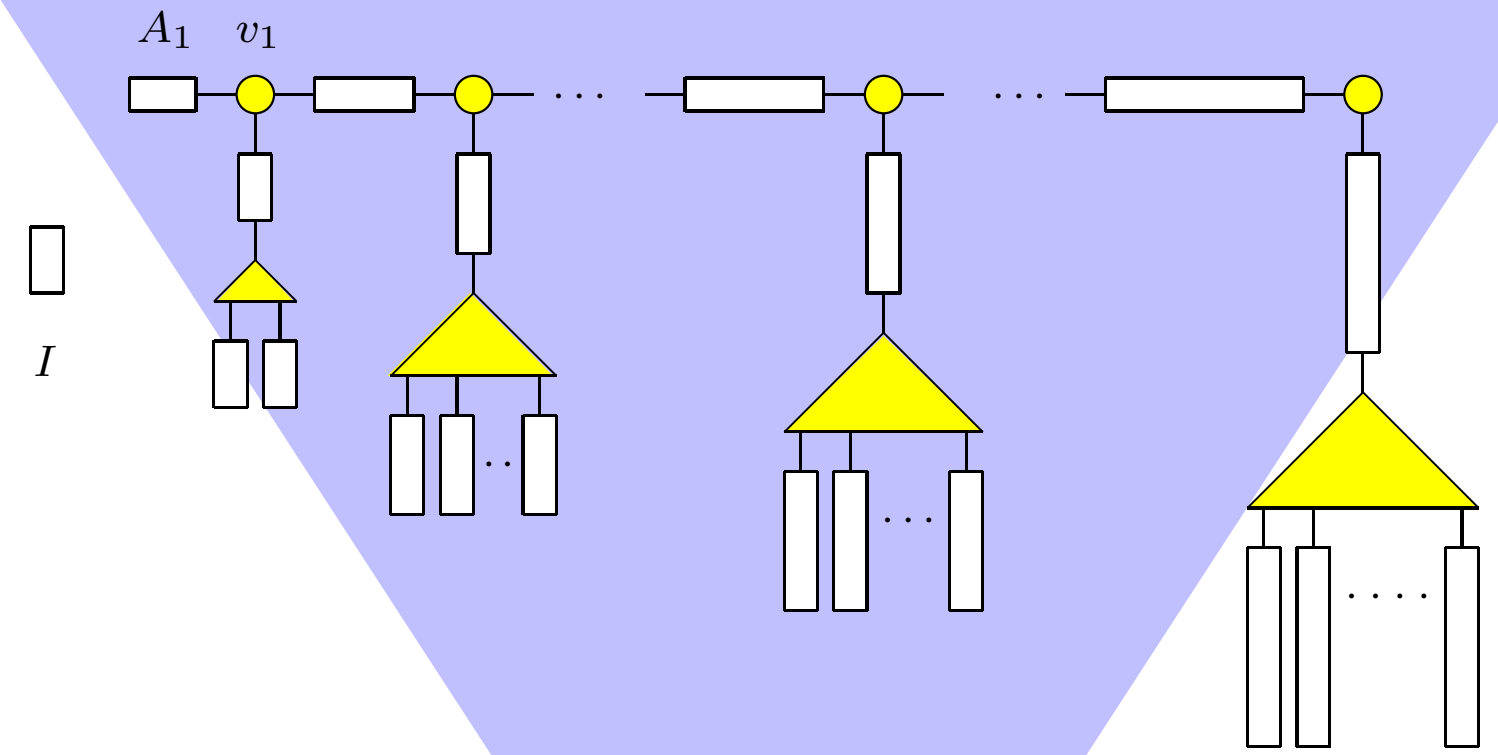
The Priority Queue



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

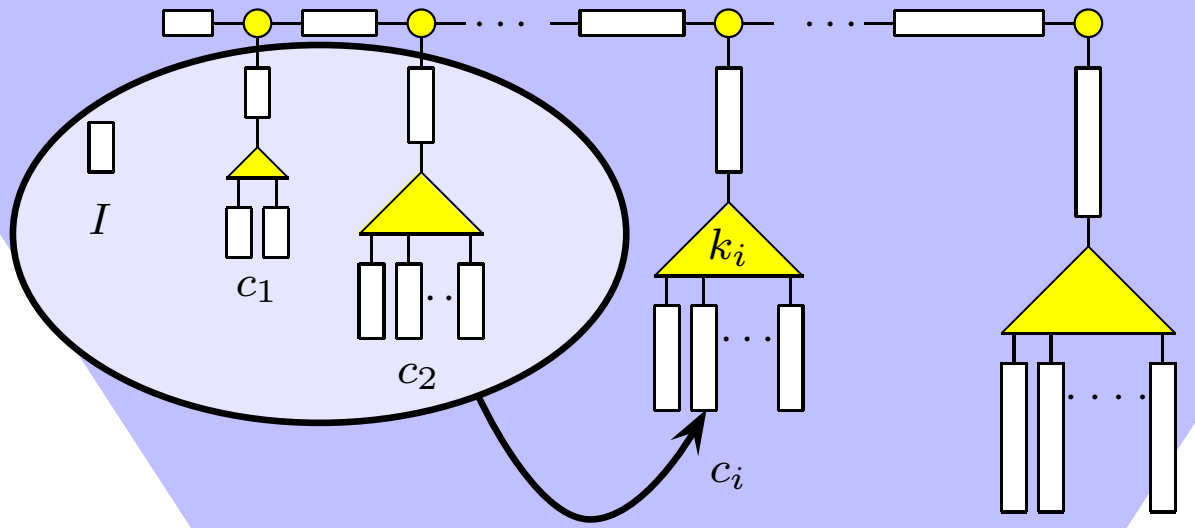
In total: A single binary merge tree

Operations — DeleteMin



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

Operations — Insert

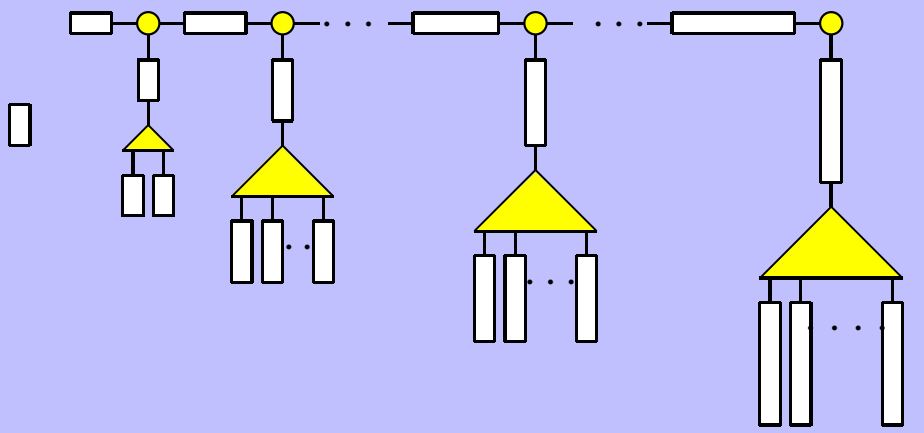


- Insert in I
- If I overflows, call Sweep(i) for first i where $c_i \leq k_i$

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:

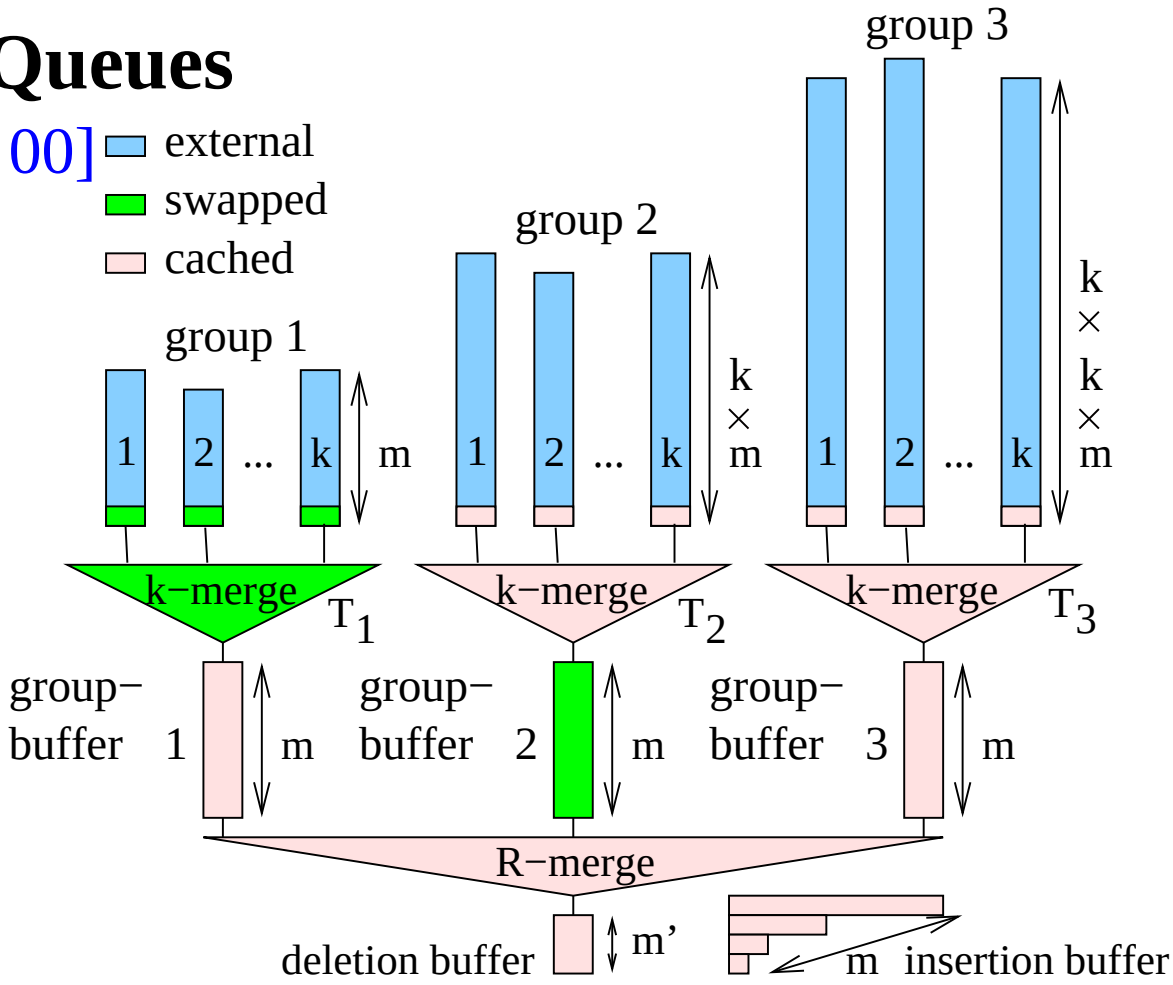
- Number N of insertions performed: $s_{i_{\max}} \leq N$
- Number of I/Os per Insert for link i: $O\left(\frac{1}{B}\log_{M/B} s_i\right)$
- By the doubly-exponentially growth of s_i, the total number of I/Os per Insert is

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

- DeleteMin is amortized for free

Large Queues

[Sanders 00]



insert: group full $\longrightarrow$ merge group; shift into next group.

merge invalid group buffers and move them into group 1.

Delete-Min: Refill. $m' \ll m$. nothing else

Experiments

Keys: random 32 bit integers

Associated information: 32 dummy bits

Deletion buffer size: 32

Group buffer size: 256

Merging degree k: 128

Near optimal : performance on all machines tried!

Compiler flags: Highly optimizing, nothing advanced

Operation Sequence:

$$(\text{Insert-DeleteMin-Insert})^N\,(\text{DeleteMin-Insert-DeleteMin})^N$$

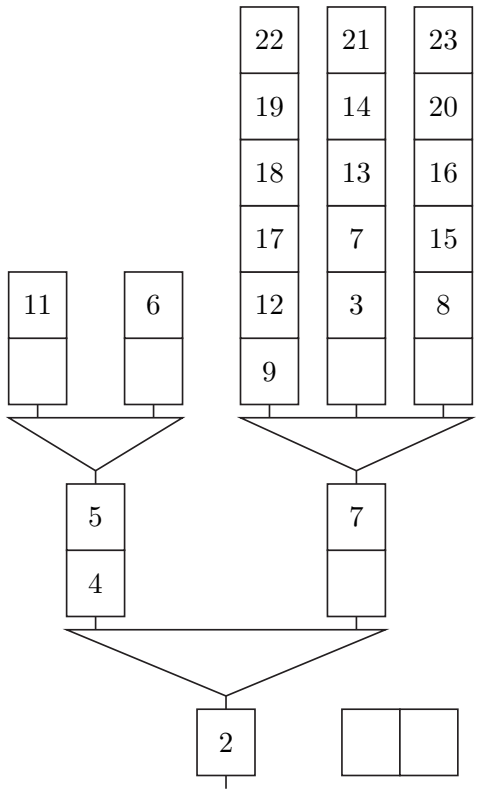
Near optimal performance on all machines tried!

Sekvenčná halda

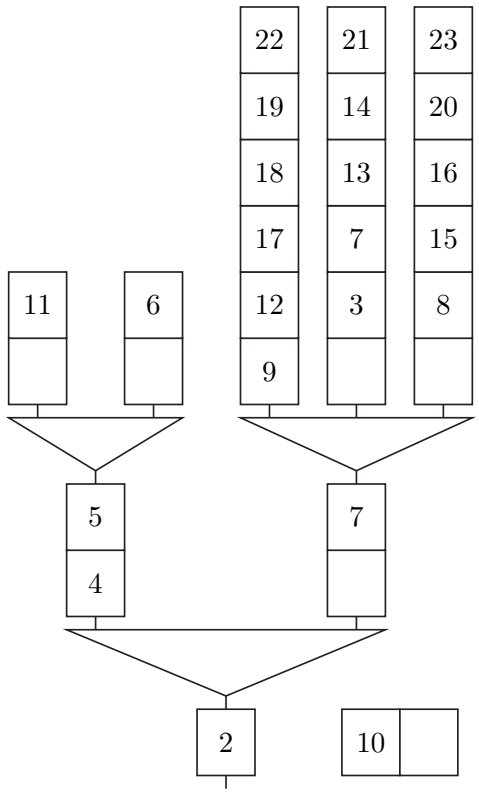
kuko

15.12.2020

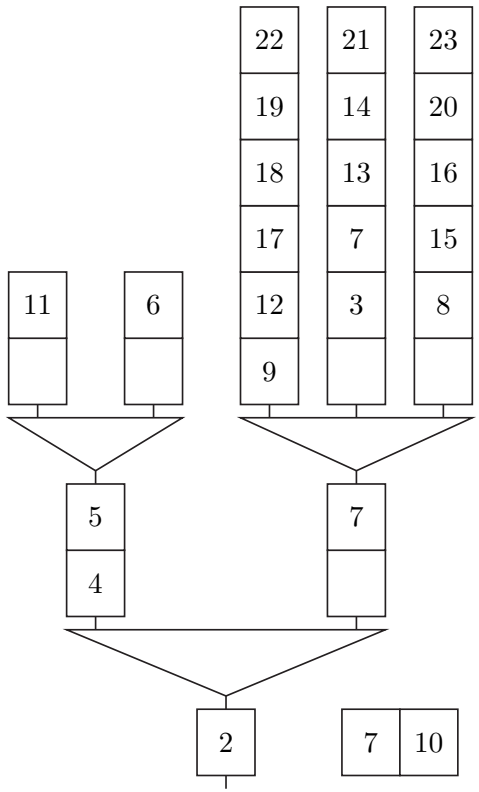
Vybrané partie z dátových štruktúr



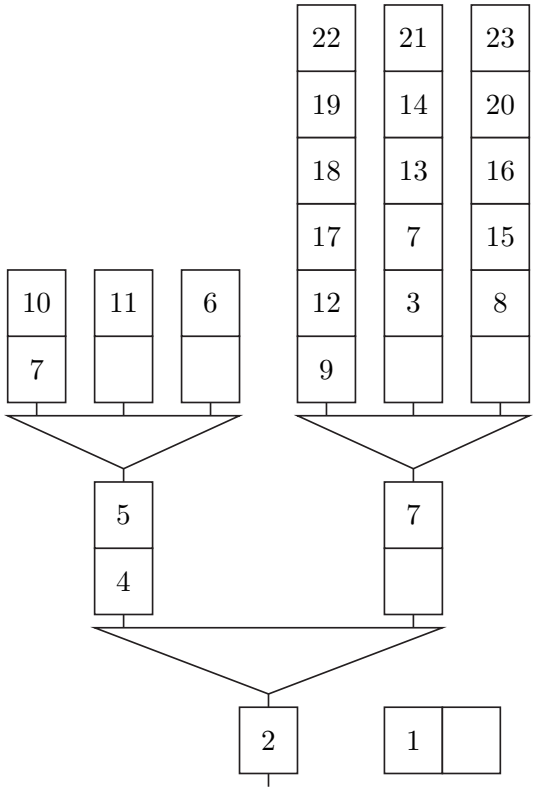
insert(10)



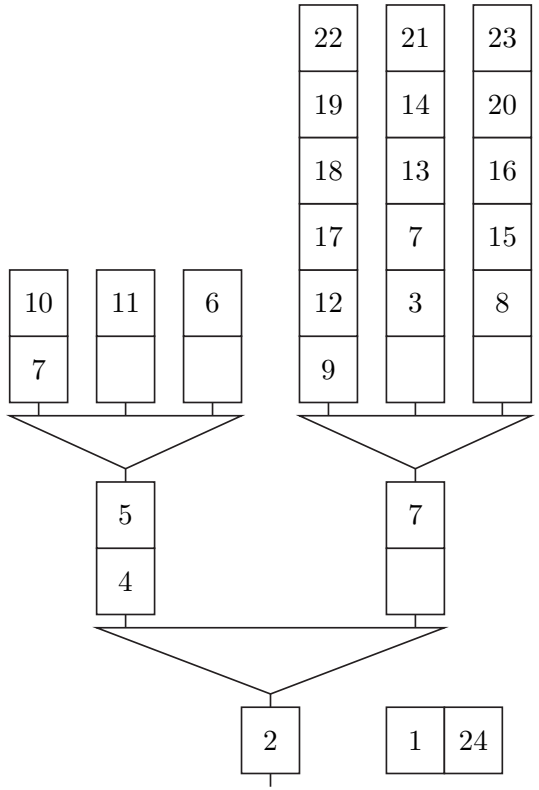
insert(7)



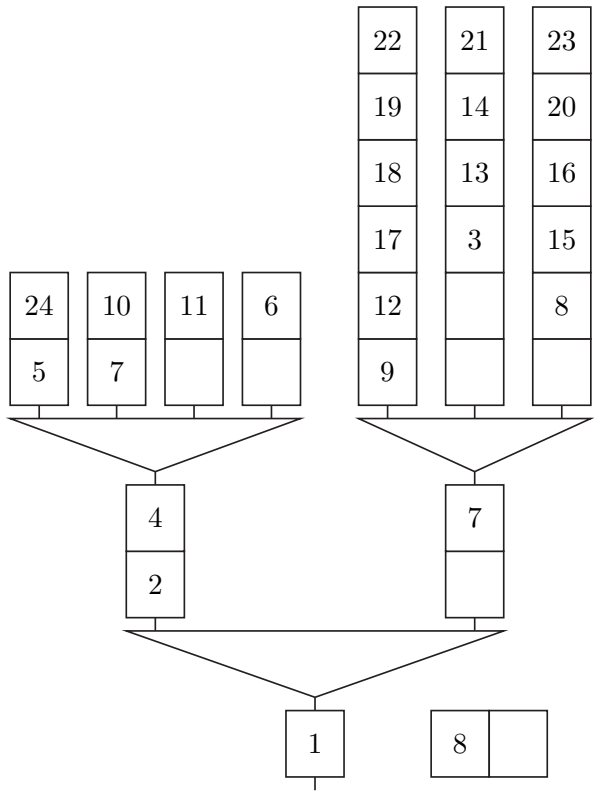
insert(1)



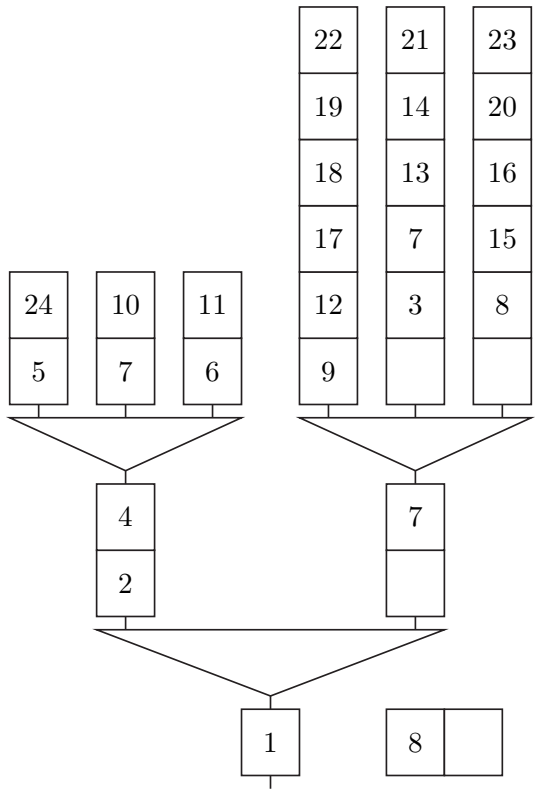
insert(24)



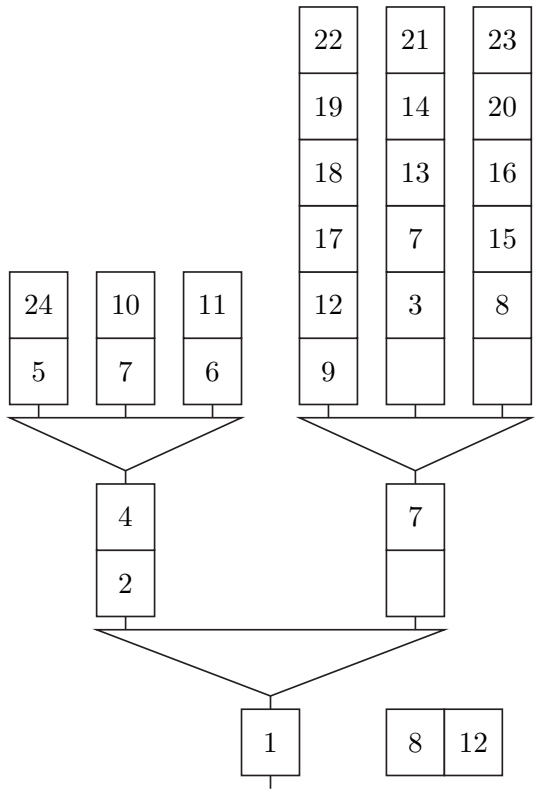
insert(8)



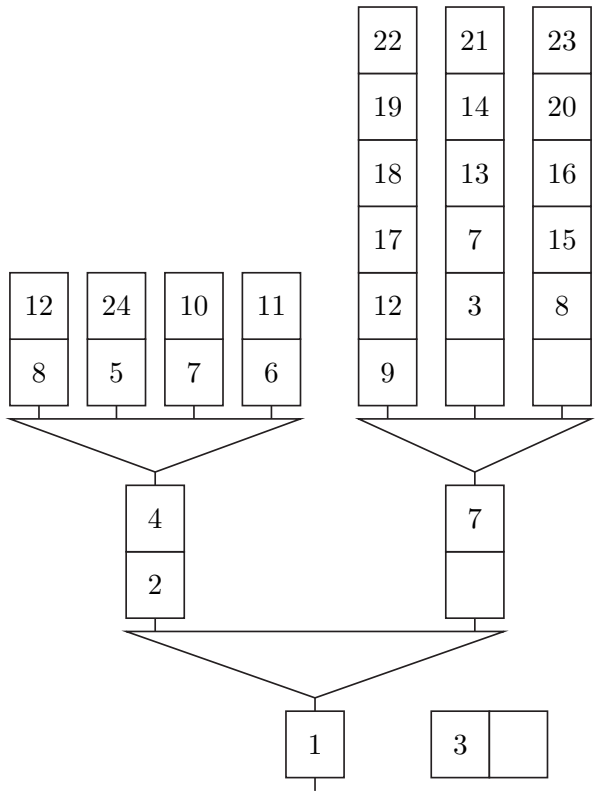
compactify group 1



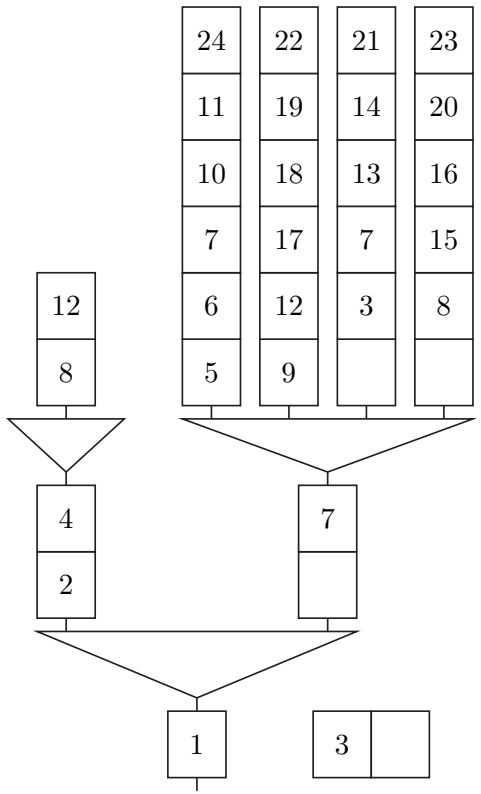
insert(12)



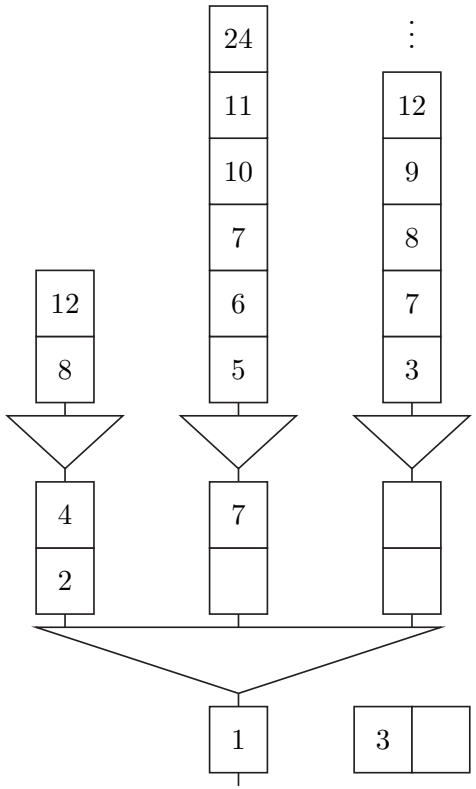
insert(3)



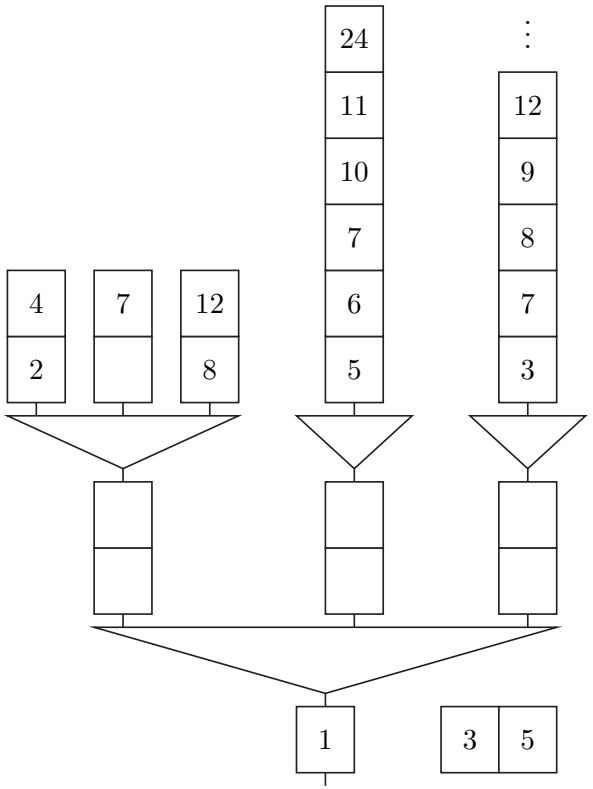
merge group 1



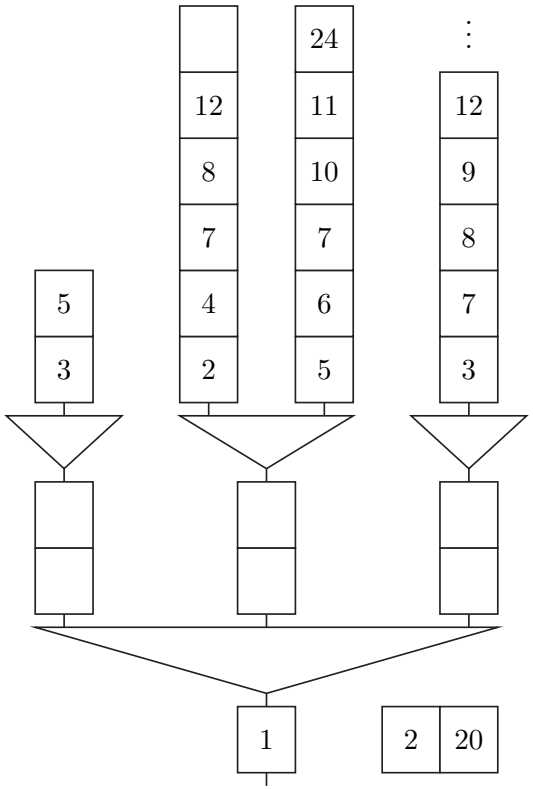
merge group 2



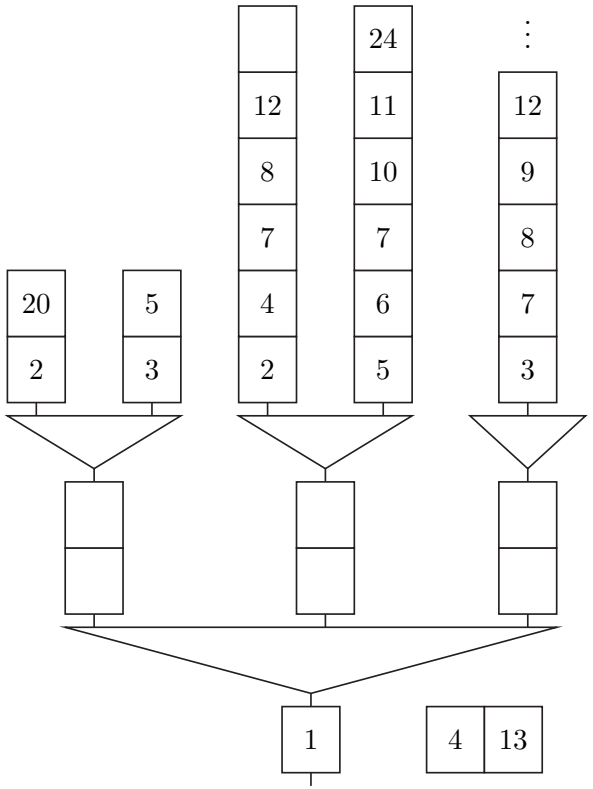
merge group buffers; insert(5)



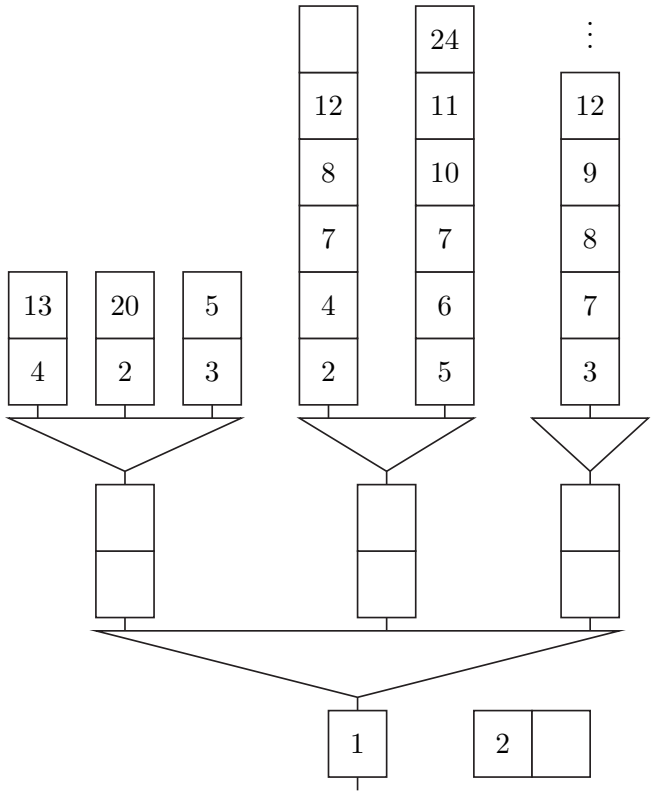
insert(2); insert(20)



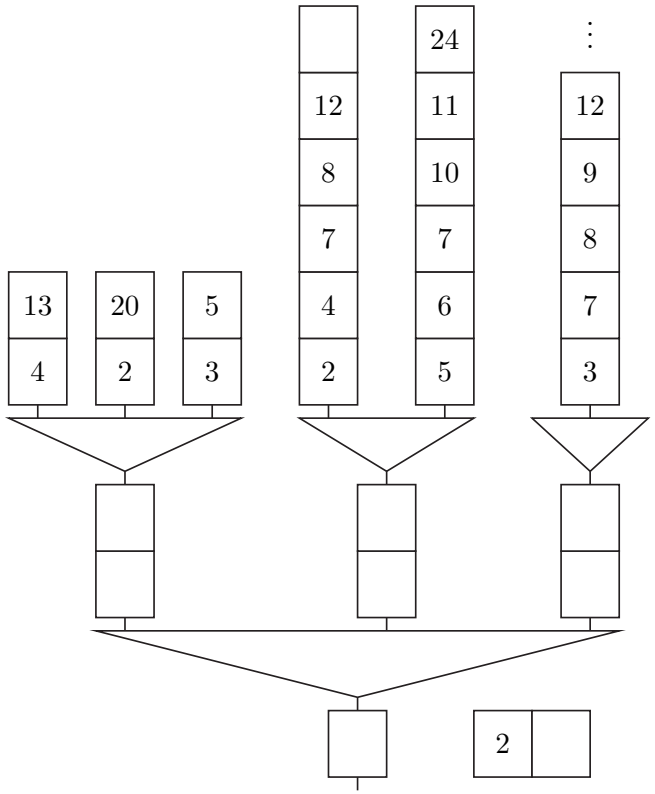
insert(4); insert(13)



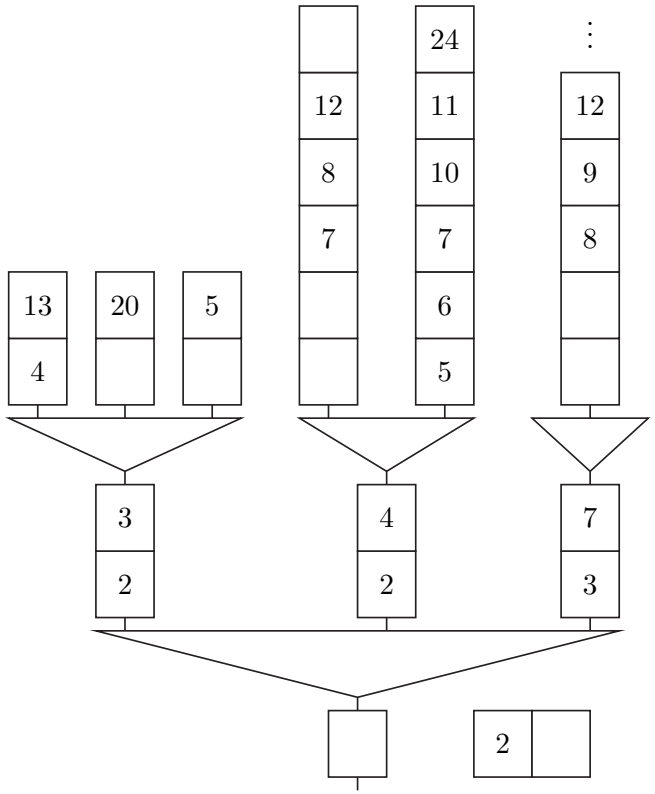
insert(2)



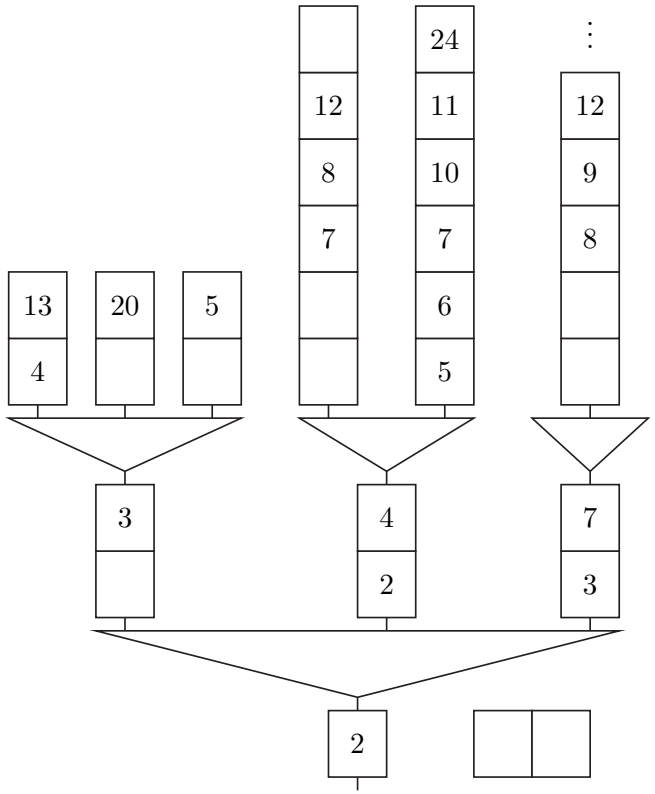
extract-min



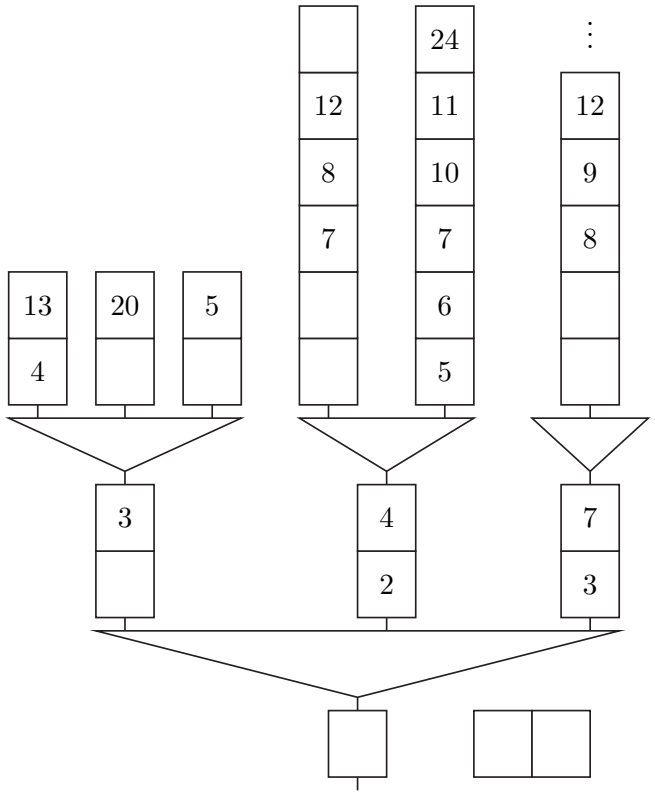
extract-min



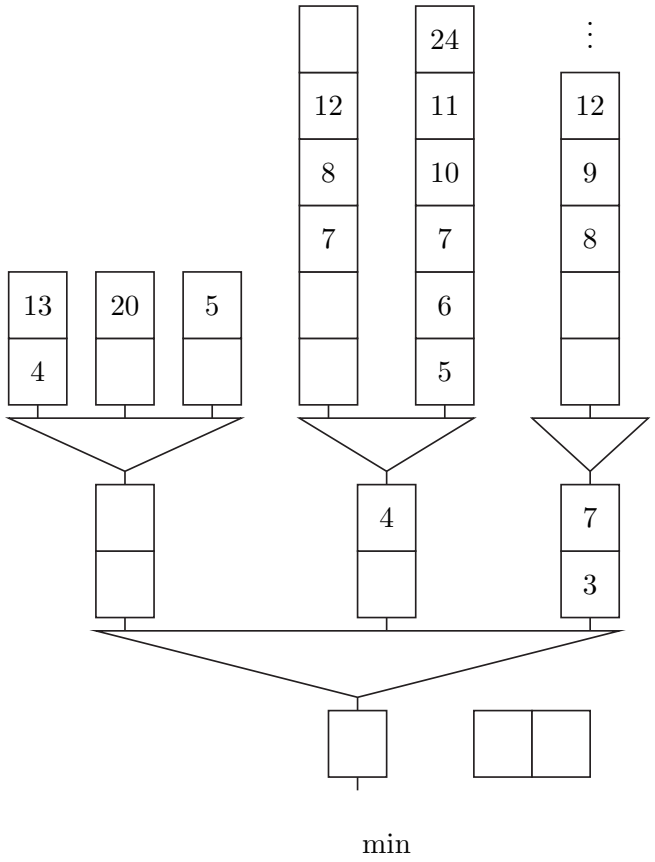
extract-min

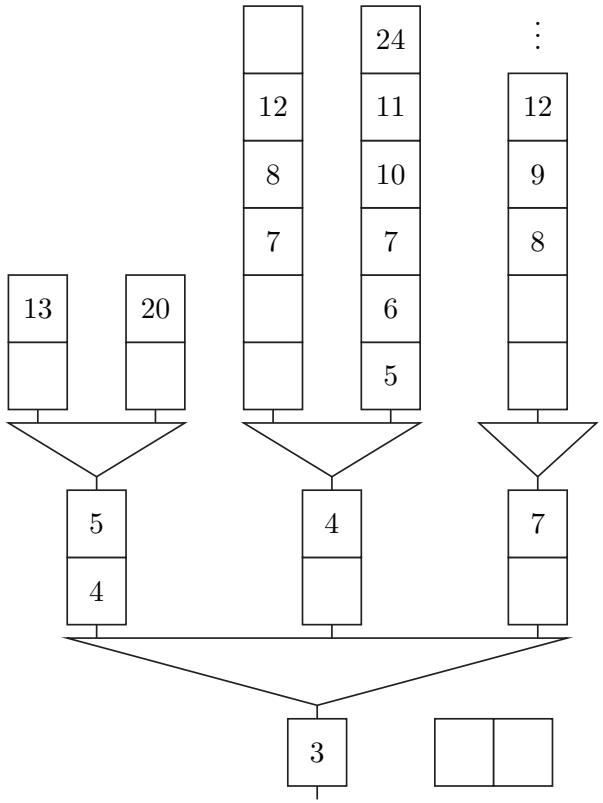


extract-min

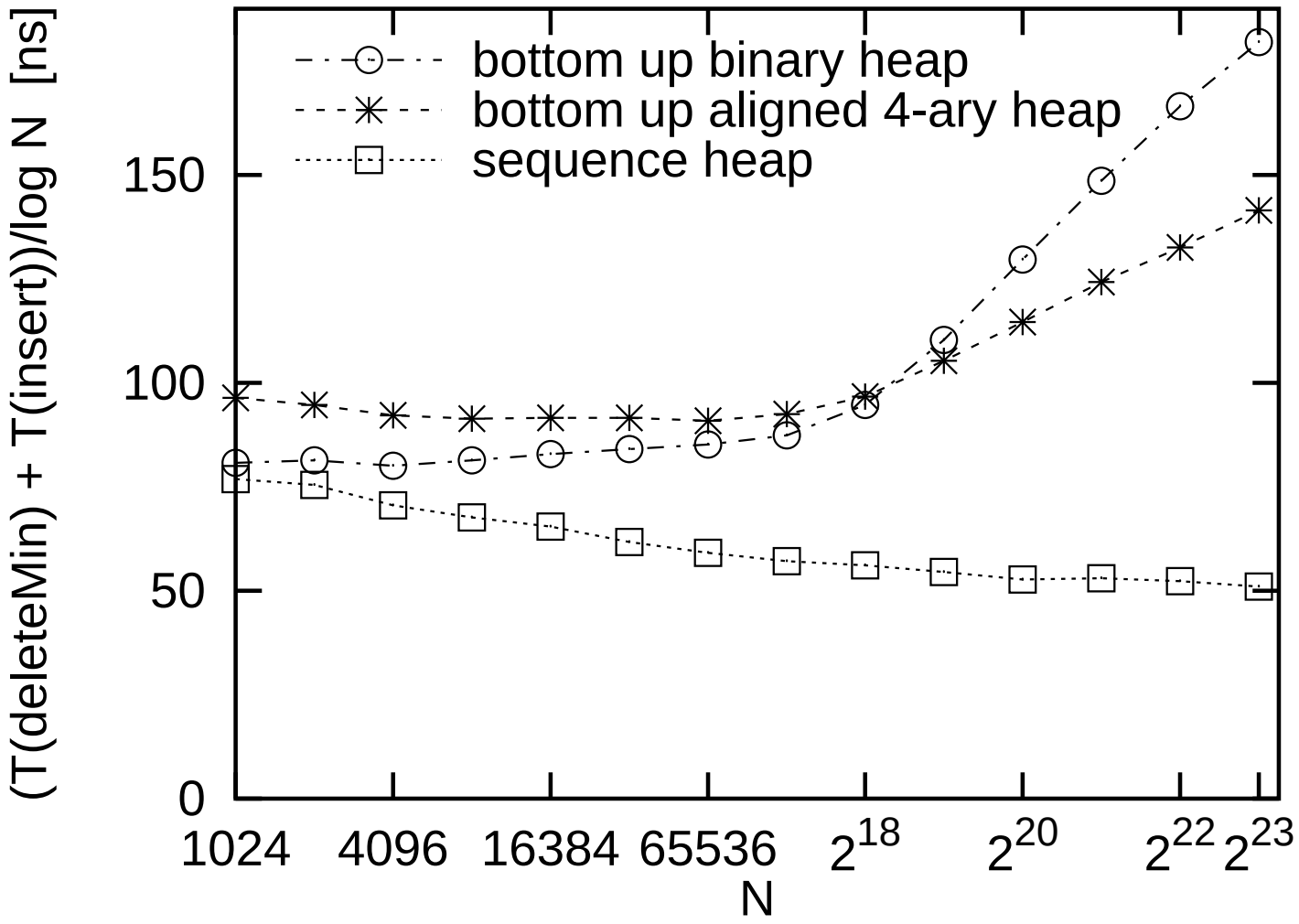


extract-min

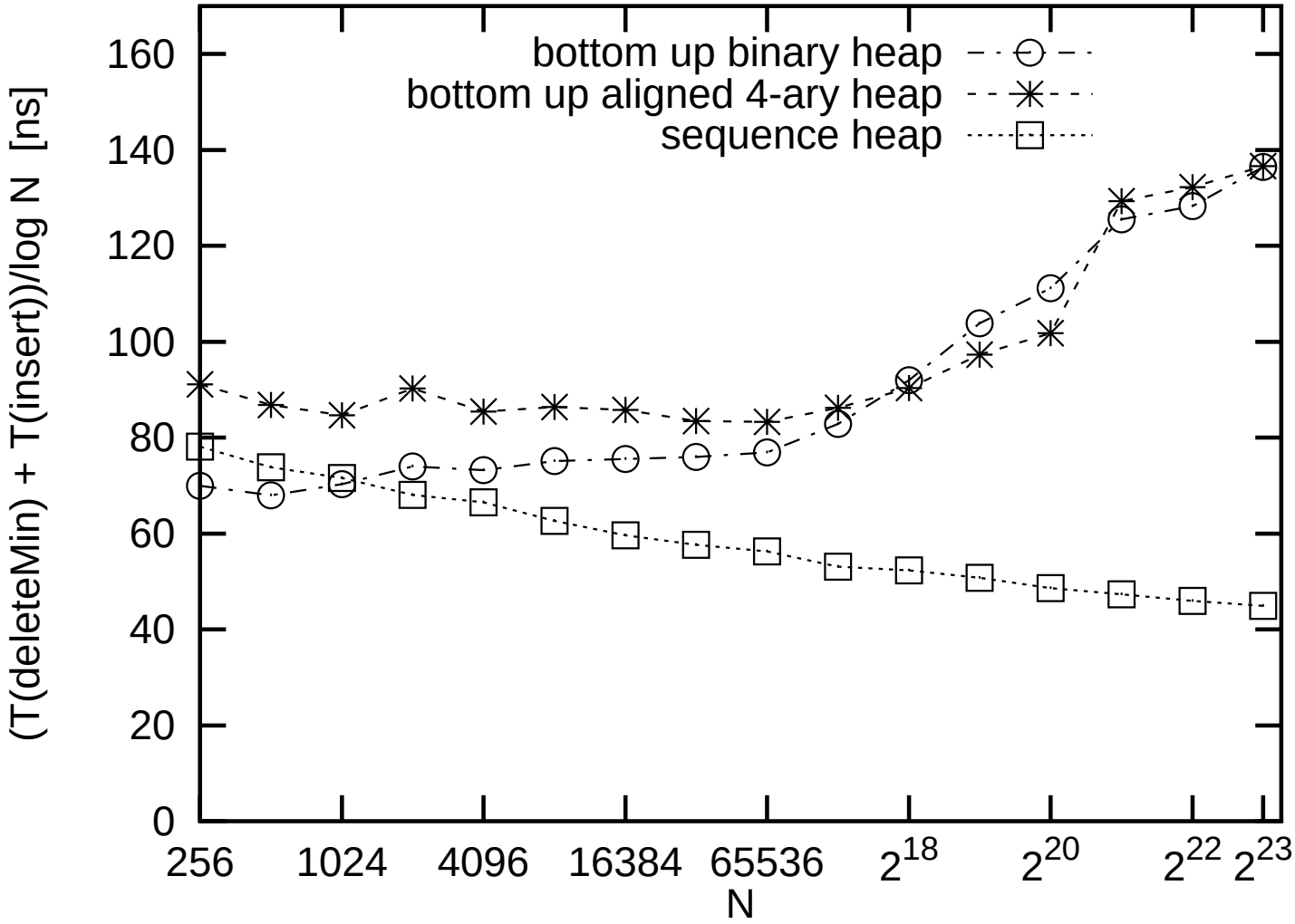

13
4
20
5
12
8
7
24
11
10
7
6
5
⋮
12
9
8
4
7
3
min


13
20
12
8
7
24
11
10
7
6
5
12
9
8
5
4
4
7
3

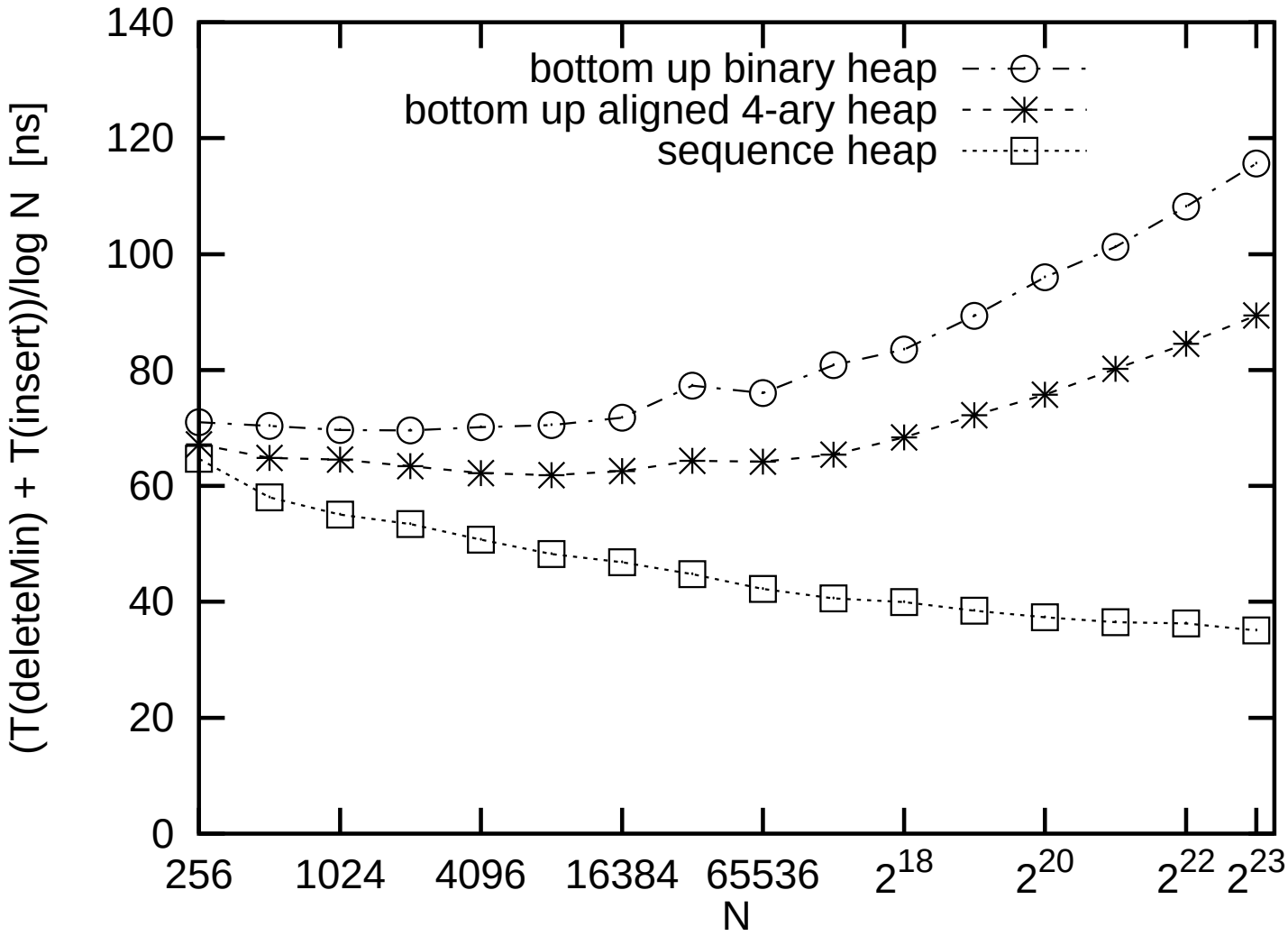
MIPS R10000, 180 MHz



Ultra-SparcIIi, 300 MHz



Alpha-21164, 533 MHz



Pentium II, 300 MHz