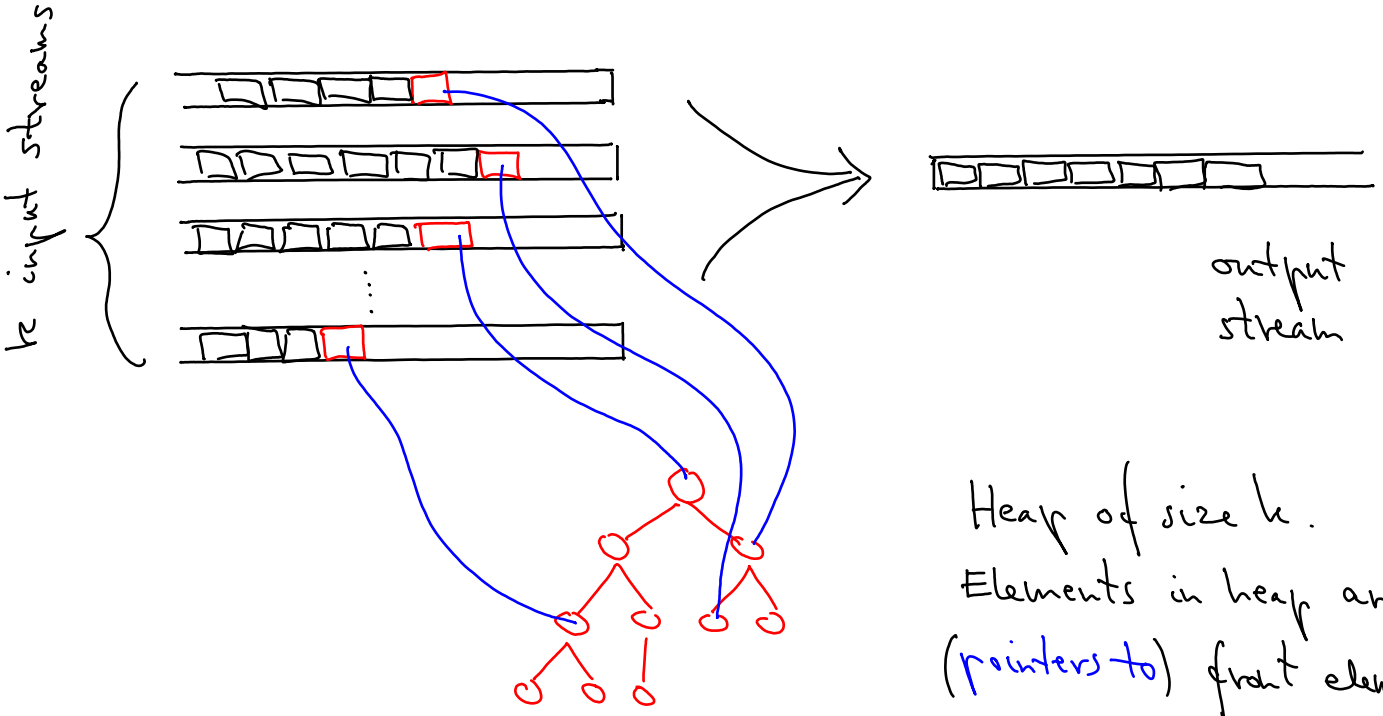
CPU Cost of Multiway Mergesort

We know that multiway mergesort uses

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

I/O's in the I/O-model. But what is its use of CPU cycles? This depends on how merge is implemented. Here is one efficient implementation:



Heap of size k.
Elements in heap are (pointers to) front elements in streams.

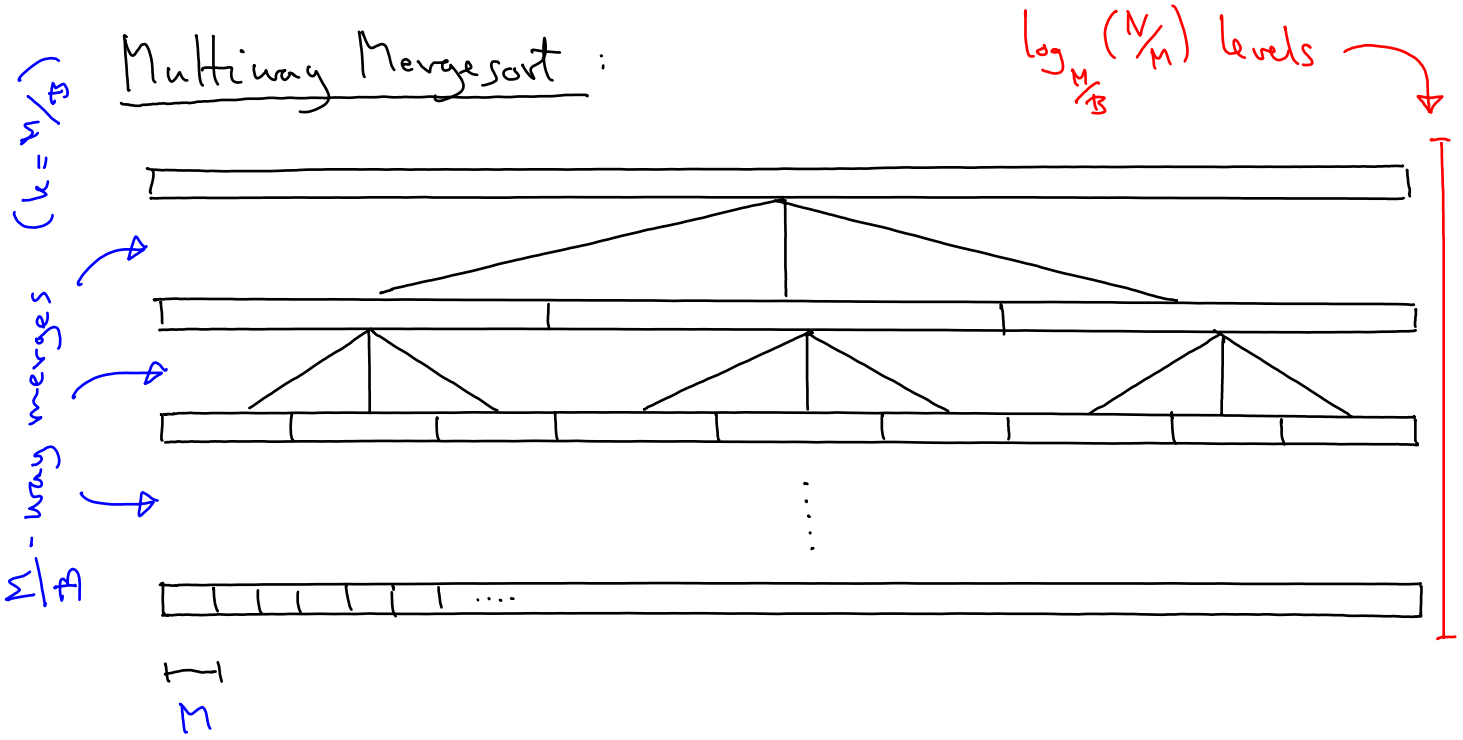
To find next element to move from the k input streams to the output stream:

1 ExtractMax (to find next element to move).

1 Insert (to insert new front element from affected input stream into heap).

This takes $O(\log k)$ CPU cycles per element moved.

Multiway Mergesort:



Each of the N elements is

i) Sorted internally in groups of size M.
This is $M \cdot \log_2(M)$ CPU cycles per group, or $\log_2(M)$ CPU cycles per element.

ii) Moved $\log_{M/B}(N/M)$ levels up via merges, each move costing $\log_2(M/B)$ CPU cycles.

In total per element this is $\log_2(M) + \log_{M/B}(N/M) \cdot \log_2(M/B)$

$$= \log_2(M) + \frac{\log_2(N/M)}{\log_2(M/B)} \cdot \log_2(M/B)$$

$$= \log_2(M) + \log_2(N/M) = \log_2(M) + \log_2(N) - \log_2(M)$$

$$= \log_2(N).$$

In total for each of the N elements this is

$$N \cdot \log_2(N).$$

From previous courses, we know this is asymptotically optimal. Hence, multiway mergesort (with the right implementation) is asymptotically optimal with respect to both I/O's and CPU cycles.