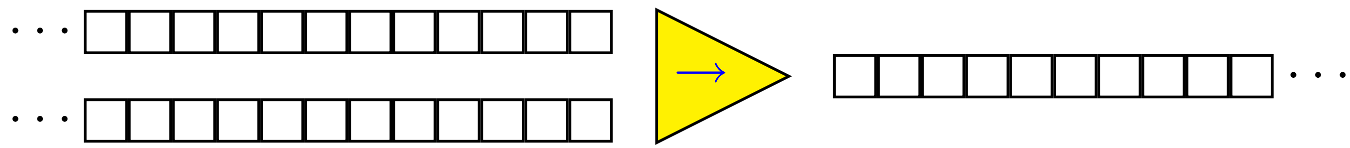
# Sorting

## Upper and Lower bounds

**[Aggarwal, Vitter, 88]**

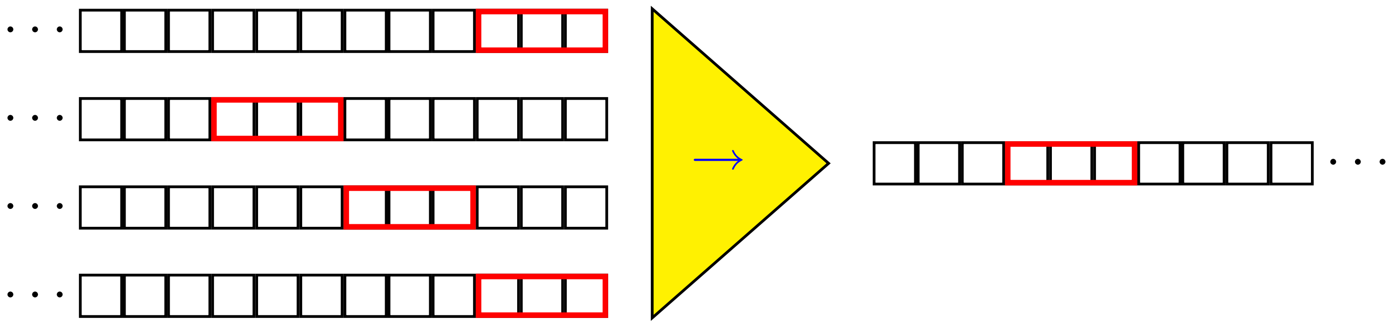
# Standard MergeSort

Merge of two sorted sequences $\sim$ sequential access



$$\boxed{\text{MergeSort:} \quad O(N \log_2(N/M)/B) \text{ I/Os}}$$

# Multiway Merge



- For I/=-efficient $k$-way merge of sorted lists we need:

$$M \geq B(k+1) \Leftrightarrow M/B - 1 \geq k$$

- Number of I/Os: $2N/B$.

# Multiway MergeSort

- $N/M$ times sort M elements internally $\Rightarrow N/M$ sorted *runs* of length $M$.
- Merge $k$ runs at at time, to produce $(N/M)/k$ sorted runs of length $kM$.
- Repeat: Merge $k$ runs at at time, to produce $(N/M)/k^2$ sorted runs of length $k^2M$, . . .

At most $\log_k N/M$ phases, each using $2N/B$ I/Os.

Best $k$: M/B-1.

$\boxed{O(N/B \log_{M/B}(N/M)) \text{ I/Os}}$

# Multiway MergeSort

$$1 + \log_{M/B}(x) = \log_{M/B}(M/B) + \log_{M/B}(x) = \log_{M/B}(x \cdot M/B)$$

$$\Downarrow$$

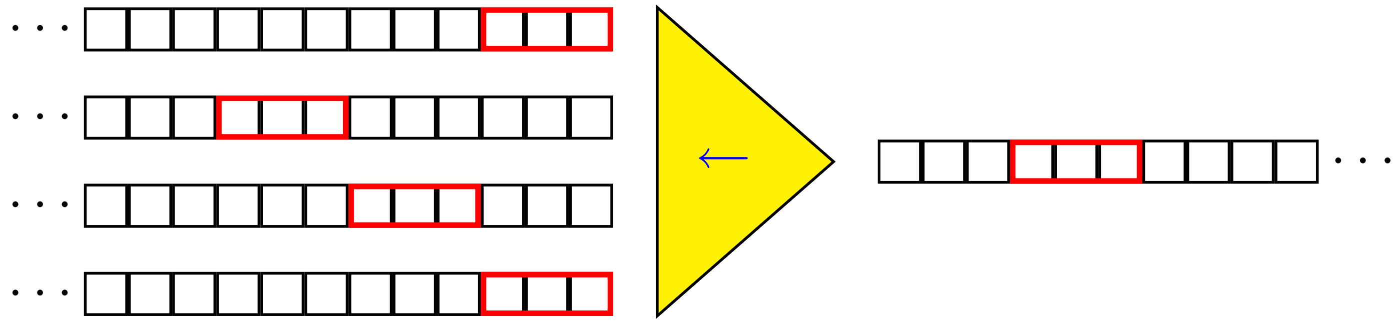
$$O(N/B \log_{M/B}(N/M)) = O(N/B \log_{M/B}(N/B))$$

Defining $\boxed{\mathsf{n} = \mathsf{N/B}}$ and $\boxed{\mathsf{m} = \mathsf{M/B}}$ we get

$$\boxed{\text{Multiway MergeSort:} \quad O(n \log_m(n))}$$

# Multiway QuickSort (DistributionSort)

Multiway splitting according to $k$ splitting elements:



- For I/O-efficient $k$-way distribution of sorted lists we need:

$$M \geq B(k+1) \Leftrightarrow M/B - 1 \geq k$$

- Number of I/Os: $2N/B$.
- We would also like to choose the $k$ elements elements such that $k$ is sufficiently large and the split is even (all subsequences are sufficiently reduced in size).

# Finding Partitioning elements

Fact: We can in O(N/B) I/Os choose $\sqrt{M/B}$ partitioning elements such that each subsequence is of size at most $\frac{3}{2}\frac{N}{\sqrt{M/B}}$.

Since $2\log_y(x) = \log_{\sqrt{y}}(x)$, an analysis somewhat similar to that for multiway mergesort gives that an I/O-optimal sorting algorithm based on distribution is possible.

# Selection

Finding splitting elements uses selection (finding $i$'th element in sorted order) as a subrutine.

Classic linear time (CPU-wise) algorithm:

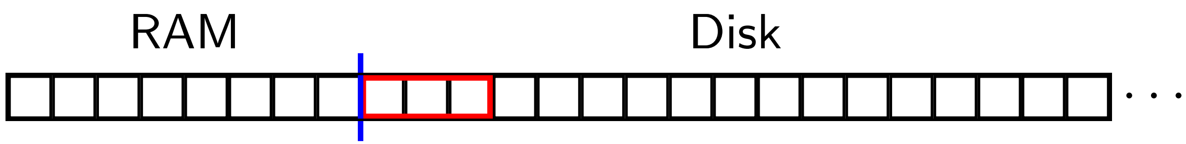
1. Split into groups of 5 elements, select median of each.
2. Recursively find the median of this set of selected elements.
3. Split entire input into two parts using this element as pivot.
4. Recursively select in relevant part.

Step 1 and 3 are scans, step 2 recurse on $N/5$ elements, and none of the lists made in step 3 are larger than around $7N/10$ elements.

As $(N/B)$ is the solution to $T(N) = O(N/B) + T(N/5) + T(7N/10)$, the algorithm is also linear in terms of I/Os.

# Sorting Lower Bound

Model of memory:




- Comparison based model: elements may be compared in internal memory. May be moved, copied, destroyed. Nothing else.
- Assume $M \geq 2B$.
- May assume I/Os are block-aligned, and that at start, input contiguous in lowest positions on disk.
- Adversary argument: adversary gives order of elements in internal memory (chooses freely among consistent answers).
- Given an execution of a sorting algorithm: $S_t$ = number of permutations consistent with knowledge of order after $t$ I/Os.

# Adversary Strategy

After an I/O, adversary must give new answer, i.e. must give order of elements currently in RAM.

If number of possible (i.e. consistent with current knowledge) orders is $X$, then there exist answer such that

$$S_{t+1} \geq S_t / X.$$

This is because any single answer induces a subset of the $S_t$ currently possible permutations (consisting of the permutations consistent with this answer), and the $X$ such subsets clearly form a partition of the $S_t$ permutations. If no subset has size $S_t/X$, the subsets cannot add up to $S_t$ permutations.

Adversary chooses answer fulfilling the inequality above.

# Possible X's

| Type of I/O | Read untouched block | Read touched block | Write |
|---|---|---|---|
| $X$ | $\binom{M}{B}B!$ | $\binom{M}{B}$ | 1 |

Note: at most $N/B$ I/0s on untouched blocks.

From $S_0 = N!$ and $S_{t+1} \geq S_t/X$ we get

$$S_t \geq \frac{N!}{\binom{M}{B}^t (B!)^{N/B}}$$

Sorting algorithm cannot stop before $S_t = 1$. Thus,

$$1 \geq \frac{N!}{\binom{M}{B}^t (B!)^{N/B}}$$

for any correct algorithm making $t$ I/Os.

# Lower Bound Computation

$$1 \geq \frac{N!}{\binom{M}{B}^t (B!)^{N/B}}$$

$$t \log \binom{M}{B} + (N/B) \log(B!) \geq \log(N!)$$

$$3tB \log(M/B) + N \log B \geq N(\log N - 1/\ln 2)$$

$$3t \geq \frac{N(\log N - 1/\ln 2 - \log B)}{B \log(M/B)}$$

$$t = \Omega(N/B \log_{M/B}(N/B))$$

Lemma was used:

a) $\log(x!) \geq x(\log x - 1/\ln 2)$

b) $\log(x!) \leq x \log x$

c) $\log \binom{x}{y} \leq 3y \log(x/y)$ when $x \geq 2y$

# Proof of Lemma

**Lemma:**

a) $$\log(x!) \geq x(\log x - 1/\ln 2)$$

b) $$\log(x!) \leq x \log x$$

c) $$\log \binom{x}{y} \leq 3y \log(x/y) \text{ when } x \geq 2y$$

$$\boxed{\text{Stirlings formula: } x! = \sqrt{2\pi x} \cdot (x/e)^x \cdot (1 + O(1/12x))}$$

Proof (using Stirling):

a) $$\log(x!) \geq \log(\sqrt{2\pi x}) + x(\log x - 1/\ln 2) + o(1)$$

b) $$\log(x!) \leq \log(x^x) = x \log x$$

c) $$\log \binom{x}{y} \leq \log\left(\frac{x^y}{(y/e)^y}\right) = y(\log(x/y) + \log(e))$$
$$\leq 3y \log(x/y) \text{ when } x \geq 2y$$

# The I/O-Complexity of Sorting

Defining

$$n = N/B$$
$$m = M/B$$
$$N/B \log_{M/B}(N/B) = \mathrm{sort}(N)$$

we have proven

**I/O cost of sorting:**

$$\boxed{\begin{aligned} &\Theta(N/B \log_{M/B}(N/B)) \\ &= \Theta(n \log_m(n)) \\ &= \Theta(\mathrm{sort}(N)) \end{aligned}}$$