§2.1. Input-Output Model

Qin Zhang
• Standard theoretical model of computation:  
  ● a processor and an infinite size memory  
  ● probing each cell of the memory has a unit cost

• Simple model crucial for success of computer industry
The reality: memory hierarchy

- **Cost:** \# blocks read/write from/to the disk
  - Disk access is $10^6$ times slower than main memory access
  - Disk systems try to amortize large access time transferring large contiguous blocks of data
  - Important to store/access data to take advantage of blocks
OS is not enough

- Most programs developed in RAM-model
  - Run on large datasets because OS moves blocks as needed

- Moderns OS utilizes sophisticated paging and prefetching strategies
  - But if program makes scattered accesses even good OS cannot take advantage of block access
The I/O-model (Aggarwal and Vitter CACM 1988)

1. $N =$ # of items in the problem instance
2. $B =$ # of items per disk block
3. $M =$ # of blocks that fit in main memory

We assume (for the convenience of analysis, you can ignore at this moment) that $M > B$
The I/O-model (Aggarwal and Vitter CACM 1988)

<table>
<thead>
<tr>
<th>Internal</th>
<th>External</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scanning:</td>
<td>$N$</td>
</tr>
<tr>
<td>Sorting:</td>
<td>$N \log N$</td>
</tr>
<tr>
<td>Permuting:</td>
<td>$N$</td>
</tr>
<tr>
<td>Searching:</td>
<td>$\log_2 N$</td>
</tr>
</tbody>
</table>

- Linear I/O: $O(N/B)$
- Permuting and sorting bounds are equal in all practical cases
- $B$ factor VERY important:
  - Example: $N = 256 \times 10^6$, $B = 8000$, 1ms disk access time
    - $N$ I/Os take $256 \times 10^3$ sec = 4266 min = 71 hr
    - $N/B$ I/Os take $256/8$ sec = 32 sec
- Cannot sort optimally with a binary search tree
Queues and Stacks

- **Queue:**
  - Maintain push and pop blocks in main memory

  ![Queue Diagram]

  $O(1/B)$ Push/Pop operations

- **Stack:**
  - Maintain push and pop blocks in main memory

  ![Stack Diagram]

  $O(1/B)$ Push/Pop operations
Sorting

- $< M$ sorted lists (queues) can be merged in $O(N/B)$ I/Os

- Unsorted list (queue) can be distributed using $< M$ split elements in $O(N/B)$ I/Os
Sorting (cont.)

• Merge sort:
  - Create $N/M$ memory sized sorted lists
  - Repeatedly merge lists together $\Theta(M)$ at a time

$\Rightarrow O(\log_M \frac{N}{M})$ phases using $O(N/B)$ I/Os each

$\Rightarrow O\left(\frac{N}{B} \log_M \frac{N}{B}\right)$
Distribution sort (multiway quicksort):

- Compute $\Theta(M)$ splitting elements
- Distribute unsorted list into $\Theta(M)$ unsorted lists of equal size
- Recursively split lists until fit in memory

$\Rightarrow O(\log_M \frac{N}{M})$ phases
$\Rightarrow O(\frac{N}{B} \log_M \frac{N}{B})$ I/Os if splitting elements computed in $O(\frac{N}{B})$ I/Os
Some of the contents are based on Lars Arge’s course
https://services.brics.dk/java/courseadmin/IOF14,