AA Lecture 25 — 4/13/21

MWU for LP

LP feasibility: $\exists x \in \mathbb{R}^n : Ax \geq b$.

Approx. LP: distinguish:

1) $\exists x \in \mathbb{R}^n$ s.t. $Ax \geq b - \varepsilon \cdot \mathbb{1}_m$.

2) $\nexists x \in \mathbb{R}^n$ $Ax \geq b$.

Def: [oracle] solves LP on average.

- input: $p \in \mathbb{R}^m_+$.
- outputs: $x \in \mathbb{R}^n$ s.t. $p^T \cdot A \cdot x \geq p^T b$.

$$\left(\sum_{i=1}^m p_i \cdot A_i\right) \cdot x \geq \sum_{i=1}^m p_i \cdot b_i$$

$$|A_i x - b_i| \leq 1 \quad \forall i \in [m].$$

"no" if no such x.

Thm: solve ε-approx LP feasibility using $O\left(\frac{\lg m}{\varepsilon^2}\right)$ oracle calls.

Pf: Algo: Init. $w_i^1 = 1$, $i = 1 \dots m$.

for $t = 1 \dots T$:

- $p_i^t = \frac{w_i^t}{\Phi^t}$ $\quad \Phi^t = \sum_i w_i^t$.
- x^t = oracle on p^t.
- if oracle says "no" then is feasible.
- $w_i^{t+1} = w_i^t (1 - \varepsilon f_i^t)$, where $f_i^t = A_i x^t - b_i$.

return $\frac{1}{T} \sum_{t=1}^{T} x^t$.

$f_i^t = A_i x - b_i \in [-1, +1]$.

$f^t = (f_1^t, f_2^t, \dots f^t)$.

MWU. $\Longrightarrow$ $M^T \le \min_{i \in [m]} m_i^T + \frac{\ln m}{\varepsilon} + \varepsilon T$.

M^T = expected error by algo

$= \sum_{t=1}^{T} \langle p^t, f^t \rangle$.

$$m_i^T = \sum_t f_i^t .$$

Claim: $M^T \geq 0$:

$$M^T = \sum_t \langle p^t, f^t \rangle = \sum_t \sum_i p_i^t \cdot (A_i x^t - b_i)$$

$$= \sum_t \left[\left(\sum_i p_i^t A_i \right) x^t - \left(\sum_i p_i^t \cdot b_i \right) \right]$$

$$\geq \sum_t 0 \quad \text{by oracle prop.} \quad \square$$

$$\Rightarrow \forall i \in [m]: \quad 0 \leq \sum_t f_i^t + \frac{\ln m}{\varepsilon} + \varepsilon T$$

$$\Rightarrow \quad 0 \leq \sum_{t=1}^{T} \left(A_i x^t - b_i \right) + \frac{\ln m}{\varepsilon} + \varepsilon T .$$

$$\Rightarrow \sum_t A_i x^t \geq T \cdot b_i - \frac{\ln m}{\varepsilon} - \varepsilon T$$

$$\Rightarrow A_i \cdot \frac{1}{T} \sum x^t \geq b_i - \varepsilon - \frac{\ln m}{\varepsilon T} .$$

$$= \varepsilon \text{ if } T = \frac{\ln m}{\varepsilon^2} .$$

$\Rightarrow A_i \cdot \frac{1}{T}\sum x^t \geq b_i - 2\varepsilon.$

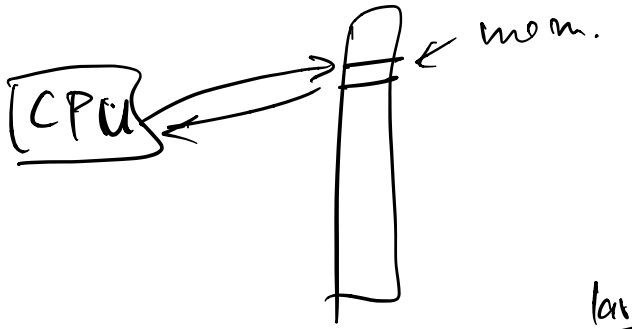
for $T = \frac{\ln m}{\varepsilon^2}$.

output of our algo.

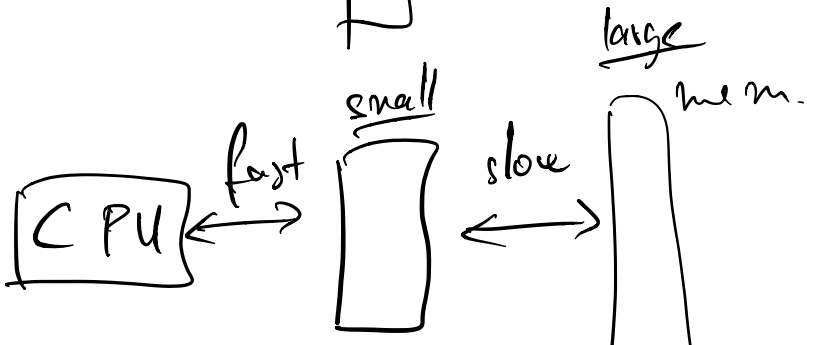
Large-scale Models of Computing.

Computers with cache.

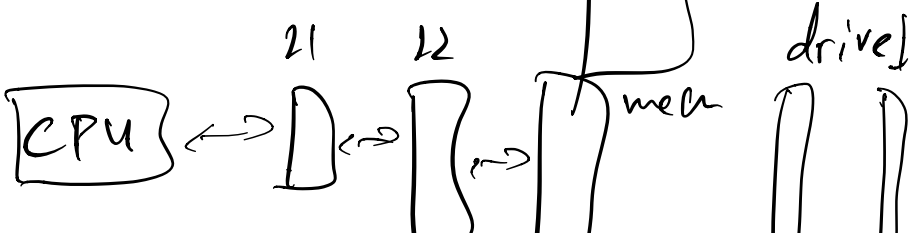
Standard Model: RAM model.



Cache:

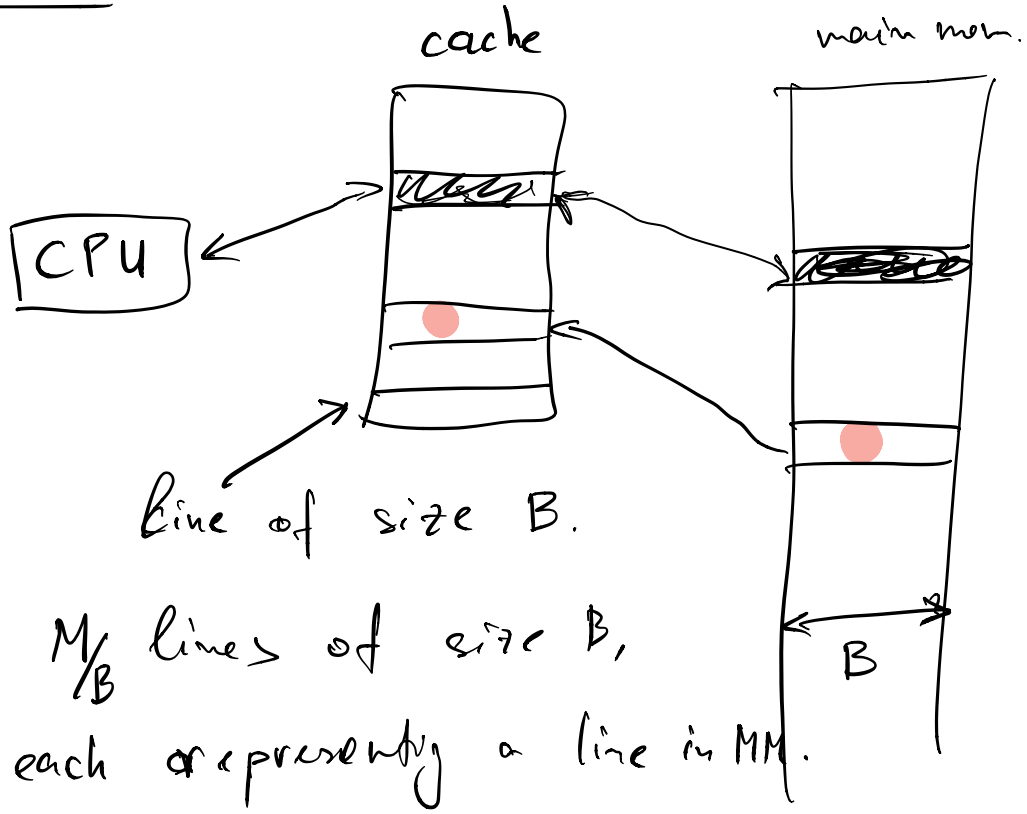


Caches



I/O Model:

Def:



Cache: M/B lines of size B, each representing a line in MM.

Cache miss: when reading MM cell ∉ cache bring-in cache line containing that cell.

B = 2k = 2048 bytes.

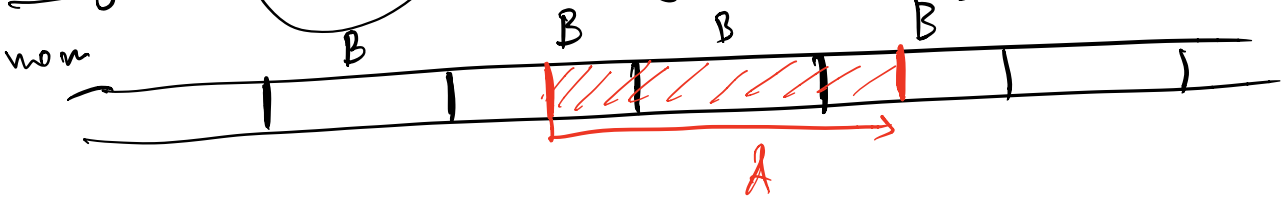
M = a few mega bytes.

Q: which cache line to replace?

- optimal algo.
- automatic "paging" algos
  Least Recently Used (LRU).

Cost: # cache line misses
"time" = # times need to bring in a cache line from MM.

Algo's: **A1:** scanning an array A[1..n]



In general: $n \le B$: Cost ≤ 2.

$n \ge B$: $\le \frac{n}{B} + 1$.

$\Rightarrow$ time $= O\left(\frac{n}{B} + 1\right)$.

A2: reading A[1..n] in a random order.

for n large enough ($n >> |M|$)

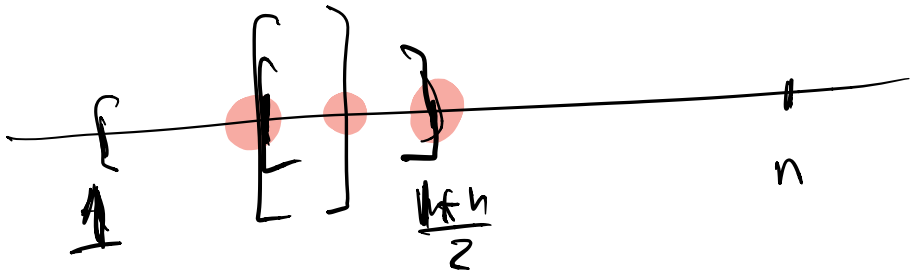
time: $\Omega(n)$.

A3: binary search: search in sorted array.

$n \le B$: cost ≤ 2.

$n >> B$: start $l = 1$, $r = n$.

$m = \frac{l+r}{2}$



at time t: distance from new query vs old query: $\frac{n}{2^t}$.

usually $>> B$.

$\frac{n}{2^t} = B \Rightarrow t = \lg \frac{n}{B}$.

Time: $t+2 = \lg \frac{n}{B} + 2 = O(\lg \frac{n}{B} + 1)$.

e.g. $B < \sqrt{n}$, $\lg \frac{n}{B} = \lg \sqrt{n} = \frac{1}{2} \lg n$.

A3': idea: use B-trees.

time: $O\left(\frac{\lg n}{\lg B}\right) = O(\lg_B n)$.