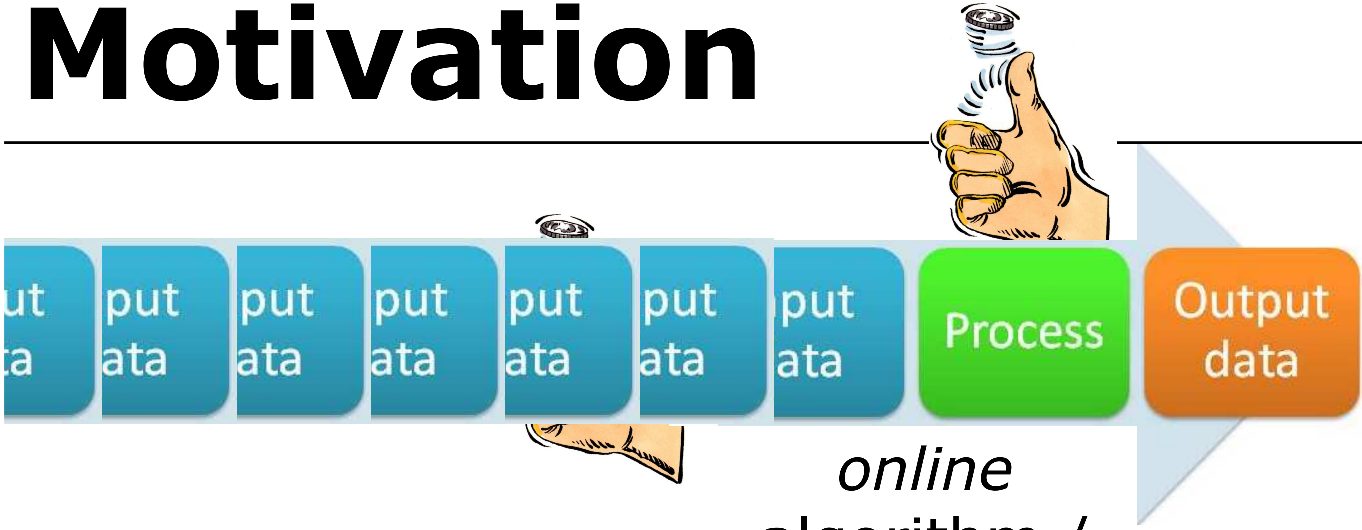
§6 Competitive Analysis of Online Algorithms

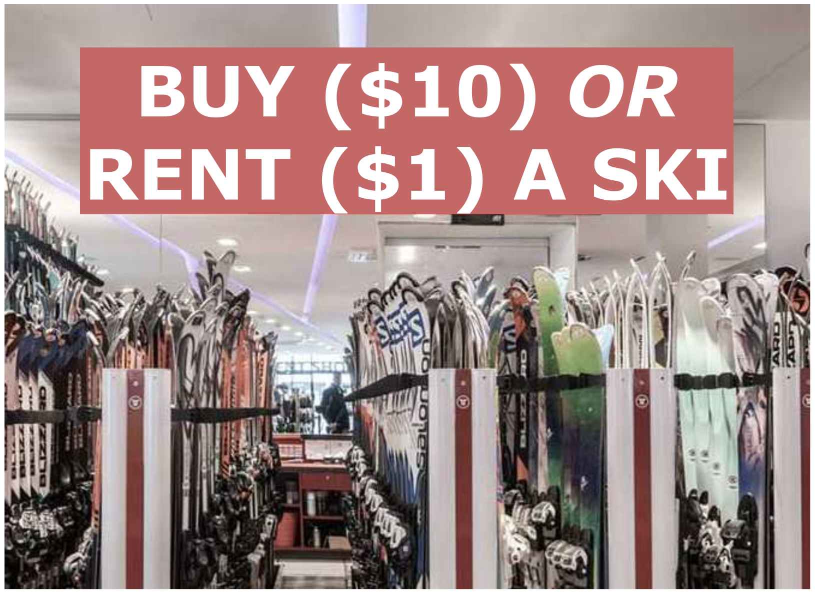
- Motivation: Ski Rental
  - *Break-Even* Algorithm
  - is 2-competitive; optimality
- Online Paging
  - *Least-Recently Used* is k-competitive
  - *Least-Frequently Used* is not competitive
  - LRU is optimal among deterministic online
- Randomization and expected competitiveness
  - 1.84-competitive randomized Ski Rental

Motivation



online algorithm / *competitive* analysis

How much better could *omniscient* **offline** algorithm perform?



Ski Rental Problem

Day by day: weather, decision:
(i) **Rent** at \$1 for <u>another</u> day
or (ii) **buy** <u>once</u> for \$$D$>1

Offline algorithm "knows":
Season will last for L days.
Rent if $L \le D$, **buy** if $L \ge D$.
Offline cost=min(L,D).

deterministic

Fix **any** *online* algorithm $\mathcal{A}$.
Run on season $L \to \infty$.
$\mathcal{A}$ eventually **buys**,
X:=#days **rent**ed.

min(L,2D-1) / min(L,D)
$\le 2-1/D$ (proof?)

Breakeven is
$(2-1/D)$-competitive!

adversary argument

Optimal ???

Breakeven algorithm:
Rent D-1 days, then **buy**.
Online cost: L if $L<D$,
$2D-1$ if $L \ge D$.

Let season last X+1 days; *re*run.

$$\frac{\text{Online cost}}{\text{Offline cost}} \equiv \frac{X+D}{\min(X+1,D)} \ge 2-1/D$$

Competitive ratio = online cost / offline cost

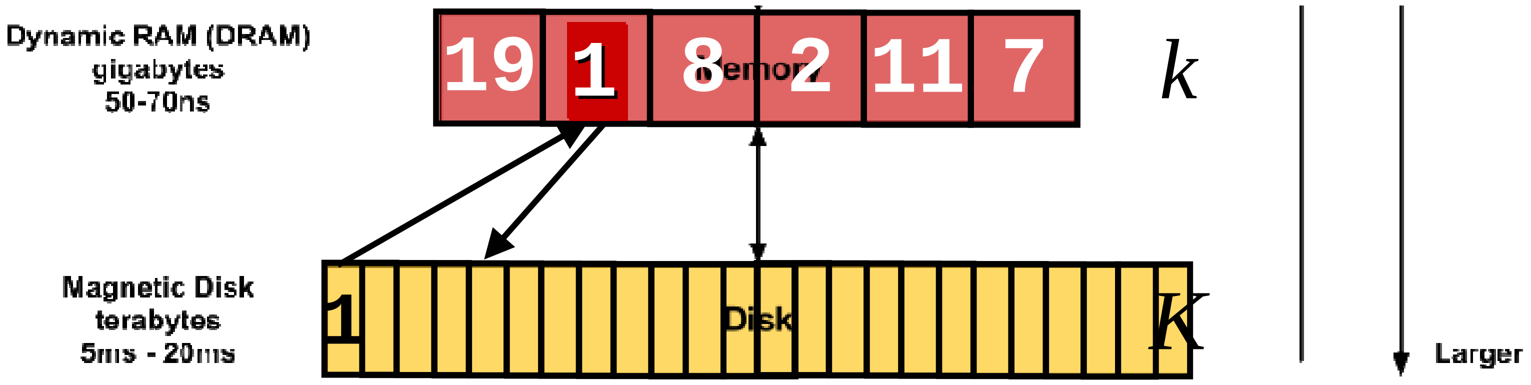
Online Paging

k pages of *fast* memory, caching $K \gg k$ *slow* pages.

For any sequence $\underline{a} = a_1,\ldots,a_N \in \{1,\ldots K\}$ of accesses, minimize the number of cache misses/load/evictions.

$\underline{a} = ($ **8 1** $)$

- **LRU**
- **LFU**
- **FIFO**
- **MIN**



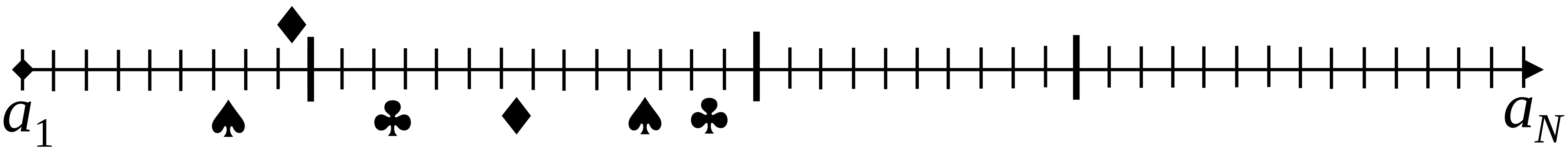
Input revealed gradually; *online* algorithm must makes decisions with <u>partial</u> knowledge.

Analyze *online* algorithm's <u>output</u> in comparison to optimal *offline* algorithm: **competitive ratio**.

Online Paging: <u>LRU</u>

Theorem: LRU has competitive ratio k = #pages

Proof: Compare LRU to optimal offline algorithm $\mathcal{A}$, started with *same* initial cache contents.



Divide $1,\ldots N$ into *rounds* $1<t_0<t_1<\ldots<t_M=N$ s.t. LRU incurs <u>precisely</u> k faults in $(t_{m-1}\ldots t_m]$ and $[1\ldots k]$ faults in $[1\ldots t_1]$.

In each round, $\geq k+1$ pages get accessed; hence $\mathcal{A}$ incurs at least 1 page fault!

fault

Whenever a new page is accessed, evict the one ***L****east* ***R****ecently* ***U****sed.*

Analyze *online* algorithm <u>output</u> in comparison to the *offline* optimum: **competitive ratio**.

Online Paging: LFU

Theorem: LFU has no (finite) competitive ratio!

Proof: Compare LFU to the optimal online algorithm on the following access sequence for $K=k+1$:

$$\overbrace{2,3,\ldots k,}^{\text{repeat }(m+1)\text{-times}}\ 2,3,\ldots k,\ 2,3,\ldots k,\ \ldots\ldots\ldots,\ 2,3,\ldots k,$$

$$\underbrace{1,\ k+1,}_{\text{repeat } m\text{-times}}\ 1,\ k+1,\ 1,\ k+1,\ \ldots\ldots\ldots,\ 1,\ k+1$$

fault

*Whenever a new page is accessed, evict the one **L**east **F**requently **U**sed.*

Analyze *online* algorithm output in comparison to the *offline* optimum: **competitive ratio**.

Optimality in Online Paging

Theorem: Every deterministic online algorithm $\mathcal{A}$ has competitive ratio $\geq k$ = #pages

Proof: Let $K>k$. Simulate $\mathcal{A}$ on initial access sequence $\underline{a}=(1,2,\ldots k)$.

"*adversary*"

Pigeonhole: choose $a_{n+1} \in \{1,\ldots K\}$ not in $\mathcal{A}$'s cache after serving $(a_1,a_2,\ldots,a_n)$.

$\mathcal{A}$ faults on *every* request!

Pigeonhole: Omnicient *offline* algorithm $\mathcal{B}$ (can) choose to evict a page not to be accessed in the next $k-1$ steps.

$\mathcal{B}$ faults only every k-th request!

p_1	p_2	p_3			p_k

$\ldots a_1, \boxed{a_2, a_3, \ldots\ldots, a_k,} a_{k+1},$

Analyze *online* algorithm output in comparison to the *offline* optimum: **competitive ratio**.

Randomized Online Algorithm

Each morning: (i) **Rent** at \$1 for another day
or (ii) **buy** once for \$$D>1$

Breakeven is $(2-1/D)$-competitive, and best possible:

Fix any algorithm $\mathcal{A}$, run on ∞ season, let X=#days it rents before buying. Restart, abort season on day #$X+1$.

adversary

Randomized Ski Rental: Flip a fair coin.

Head: *Breakeven* (≈ rent for D days, then buy)

Tail: Rent for $\tfrac{2}{3}D$ days, then buy.

1.833

$$L \geq D: \mathbb{E}[\text{cost}] = \tfrac{1}{2}\cdot(2D) + \tfrac{1}{2}\cdot(\tfrac{2}{3}+1)\cdot D = \boxed{11/6}\cdot D$$

$$\tfrac{2}{3}D \leq L < D: \mathbb{E}[\text{cost}] = \tfrac{1}{2}\cdot(L) + \tfrac{1}{2}\cdot(\tfrac{2}{3}+1)\cdot D \leq \boxed{(\tfrac{1}{2}+\tfrac{1}{2}\cdot(\tfrac{2}{3}+1)/\tfrac{2}{3})}\cdot L$$

$$L < \tfrac{2}{3}D: \mathbb{E}[\text{cost}] = \tfrac{1}{2}\cdot(L) + \tfrac{1}{2}\cdot(L) = L$$

1.75

"Randomization can beat an adversary!"

§6 Summary

- Motivation: Ski Rental
  - *Break-Even* Algorithm
  - is 2-competitive; optimality
- Online Paging
  - *Least-Recently Used* is k-competitive
  - *Least-Frequently Used* is not competitive
  - LRU is optimal among deterministic online
- Randomization and expected competitiveness
  - 1.84-competitive randomized Ski Rental