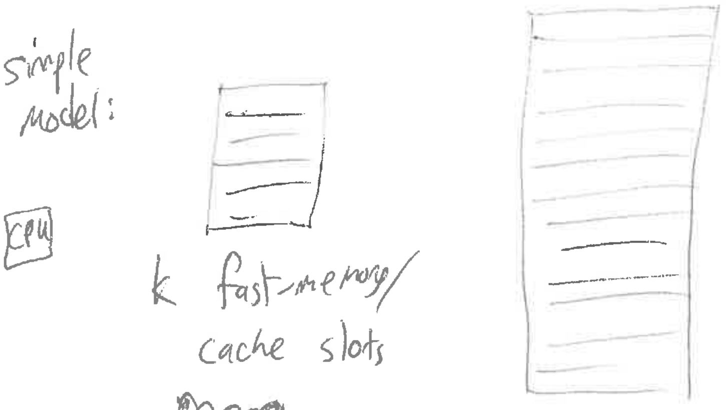
Lecture 18: Online algs Part 2: Paging

[[Today we'll do one long example that illustrates many concepts: competitive ratio, determinism vs randomized algs, amortized analysis...]] [["Paging" is a funny old-fashioned word; it's all about...]] Cache

[[Disk vs memory, mem vs. cache, CPU vs motherboard ... extremely important for practical alg. run time.]].

simple model:



k fast-memory/cache slots

N data/disk/slow-mem items/"pages"

Model: CPU will access/request a seq. $I = r_1, r_2, r_3, \ldots$ ($r \in \{1, \ldots, N\}$)

If item in cache: ☺ [[free]]

Else:

- incur cost 1 [["cache miss"/"page fault"]]
- must move item into cache [[this is a modeling choice; maybe not always appropriate, but let's use it]]
- must "evict" one item from cache

[[maybe at "the beginning of time" cache is not full yet, but we'll ignore, assuming cache starts full]]

Alg = Eviction strategy. [[Ideas?]]

e.g.: $k=3$, $N=4$, requests: $I = 1, 2, 3, 2, 4, 3, 4, 1, 2, 3, 4, \ldots$

"Offline Optimal": IF you psychically knew I, optimal alg is....

Evict page whose next req. is farthest in future!

[[Takes a teeny bit of thought to see this greedy strat is]] indeed offline optimal. But it is.

On e.g.: $1\ 2\ 3\ 2\ \underset{X \to \text{kick out } 2}{4}\ 3\ 4\ 1\ \underset{x \to \text{kick out } 1}{2}\ 3\ 4$

[[But we don't know future, so offline alg?]]

"LRU" (Least Recently Used): evict page that's L.R.U.

On e.g.: $1, 2, 3, 2, \underset{x \to \text{Evict } 1}{4}\ 3\ 4\ \underset{x \to \text{evict } 2}{1}\ \underset{x \to \text{evict } 3}{2}\ \underset{x \to \text{evict } 4}{3}$

[[Hmm... kinda bad if next req. is whatever was just evicted.]]

"Adversary's" worst I for LRU: 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, ... (in k=3 case; always just uses N=k+1).

- LRU has a miss every $\underbrace{1}$ requests
- Offline opt has a " " $\underbrace{3}_{\substack{\downarrow \\ \text{in general } k.}}$ " . $\quad 1\ 2\ 3\ \underset{x \to \text{evict } 3}{4}\ 1\ 2\ \underset{x \text{ evict } 2}{3}\ 4\ 1\ \underset{x}{2} \ldots$

⇒ C.R. for LRU no better than k. :([[pretty bad]]

[[Better alg?]]

"FIFO": evict page that's been in cache longest

ex: [[~~[illegible]~~ same bad example I ~~[illegible]~~]] implies C.R. no better than k :(.

fact: ~~[illegible]~~ No det. algorithm can have C.R. < k.

proof: · Given Alg, [Always use N=k+1.] let I be a long seq. ~~where~~ that always requests the page Alg just evicted. [[Need to know Alg to design I.]]

⤷ miss every 1 reqs.

· Claim: for any I with N=k+1, offline opt. only misses once every ≥ k reqs.

Because when item i evicted, all other k−1 items will subsequently be req'd earlier: ⇒ k−1 succeses per miss.

⟦ There's a twist you can invent to make this situation less sad... ⟧

LRU on input I with cache size k vs. Offline opt. on I with cache size k

↑ might be k times worse ☹

Force it to have cache size $\frac{k}{2}$.

Now. "C.R." is $\lesssim 2$ ☺

⟦ kinda cheating, not apples to apples. But... "RESOURCE AUGMENTATION": maybe okay to imagine you can double your cache... ⟧

Proof sketch: Given any seq. $I = 1, 3, 1, 1, \{4, 7, 2, 2, 2\} 3, 6, 5 \ldots,$ divide into consecutive "phases":

phase = maximal sequence of k distict items

phase boundaries for $k=3$

Fact 1: LRU (& FIFO) have cost $\le k$ in each phase ⟦ evident from def^n ⟧

Fact 2: Offline opt. with cache of size $\frac{k}{2}$ must have $\geq \frac{k}{2}$ misses per phase.

$\rightsquigarrow$ "C.R." of $\le \frac{k}{k/2} = 2$.

ex: For LRU with k vs. offline opt. with $h \le k$, "C.R." is $\le \frac{k}{k-h+1}$ ⟦ so k for $h=k$, but ≤ 10 for $h = 0.9k$, e.g. ⟧

⟦ Back to usual apples to apples model. ⟧

⟦ The sad thing about determinictic algs is "adversary" at beginning always knows what's in your cache, knows what you just evicted, can always choose I so that next req. = last evict. How can we make it so adversary doesn't "know" what you just evicted? Randomness! ⟧

<u>Random model</u>: Adversary <u>knows</u> your randomized eviction policy

" <u>doesn't see</u> " random coin flips/choices, can't see your cache.

[It's like the O.S.'s alg's spec is published, but cache is in "private memory".]

∴ May as well assume Adversary fixes all I in advance

[But now <u>you</u> can't change your alg!]

Motivates def^n: $C.R. = \max_{I} \dfrac{E[\text{cost of Alg on } I]}{\text{Offline Opt}(I)}$

you go next, randomized

Adv. goes first, det'ic

<u>Thm</u>: ① ∃ randomized alg., "Marking", with $C.R. \le O(\log k)$!

② ~~[illegible]~~ Every alg, even randomized, has $C.R. \ge \Omega(\log k)$

[Won't show ②. We'll sketch proof of ①.]

<u>Marking</u>: "1-bit LRU"

- initialize cache to $1, 2, \ldots, k$, & all pages "unmarked"
- when i requested: if in cache, "mark" it
  else, evict a random unmarked page,
  ~~[illegible]~~
  except if all pages marked... <u>unmark</u>-all first
  Then mark i when it's brought to cache.

We'll show $C.R. \le O(\log k)$ assuming $N = k+1$.

[For $N > k+1$, proof is ≈ 25% harder.]

Observation: Unmark-all's happen exactly at "phase" boundaries. $k=4$ eg...

$$\} \; 3 \quad 1 \quad 1 \quad 1 \quad 2 \quad 1 \quad 4 \; \}$$

unmark all

↑ in phase, # marked = # diff. items seen so far in phase.

Q: What is $\mathbb{E}[\text{cost of a phase}]$?

~~Only newly-marked items have cost, others free~~
~~So can just focus on them~~

WLOG, say prev. phase ended with $1, 2, \ldots, k$ in cache.
($k=4$, $N=5$ example)

Let's make a diagram ~~showing all $N=k+1$~~ items, annotated with probability it's in cache, and a dot ● to mean marked.

Start of phase:

1	1	1	1	0
1	2	3	4	5

// 1, 2, 3, 4 are in cache with prob 1, 5 in cache with prob 0, no marks

Because we're starting a new phase, 5 must have just been requested.

requests	1	2	3	4	5	$\mathbb{E}[\text{cost}]$
⑤	3/4	3/4	3/4	3/4	● 1	+1
②	2/3	● 1	2/3	2/3	● 1	$+\frac{1}{4}$
①	● 1	● 1	1/2	1/2	● 1	$+\frac{1}{3}$
④	● 1	● 1	0	● 1	● 1	+1/2

Perhaps 5 now req'd many times. All free, so let's move to next req.

Some more cache hits, then phase ends.

In general, $\mathbb{E}[\text{cost of phase}] = 1 + \frac{1}{k} + \frac{1}{k-1} + \cdots + \frac{1}{2}$ (6)

$$= \text{"}H_k\text{"} \approx \ln k = O(\log k).$$

Claim: Opt offline alg pays $\geq$ 1 per phase

~~first k phases, pays~~

$\{$ 4 1 1 2 3 $\{$ 5 5 2 1 1 1 2 3 $\}$ 4 4 1 1 2 5 $\{$ 3 1 2 1 1 1 4 $\{$ 5 4 3 2 $\}$ 1 …

X who does it evict? 4!

X who does it evict? 3

X evicts 5

X

. . .