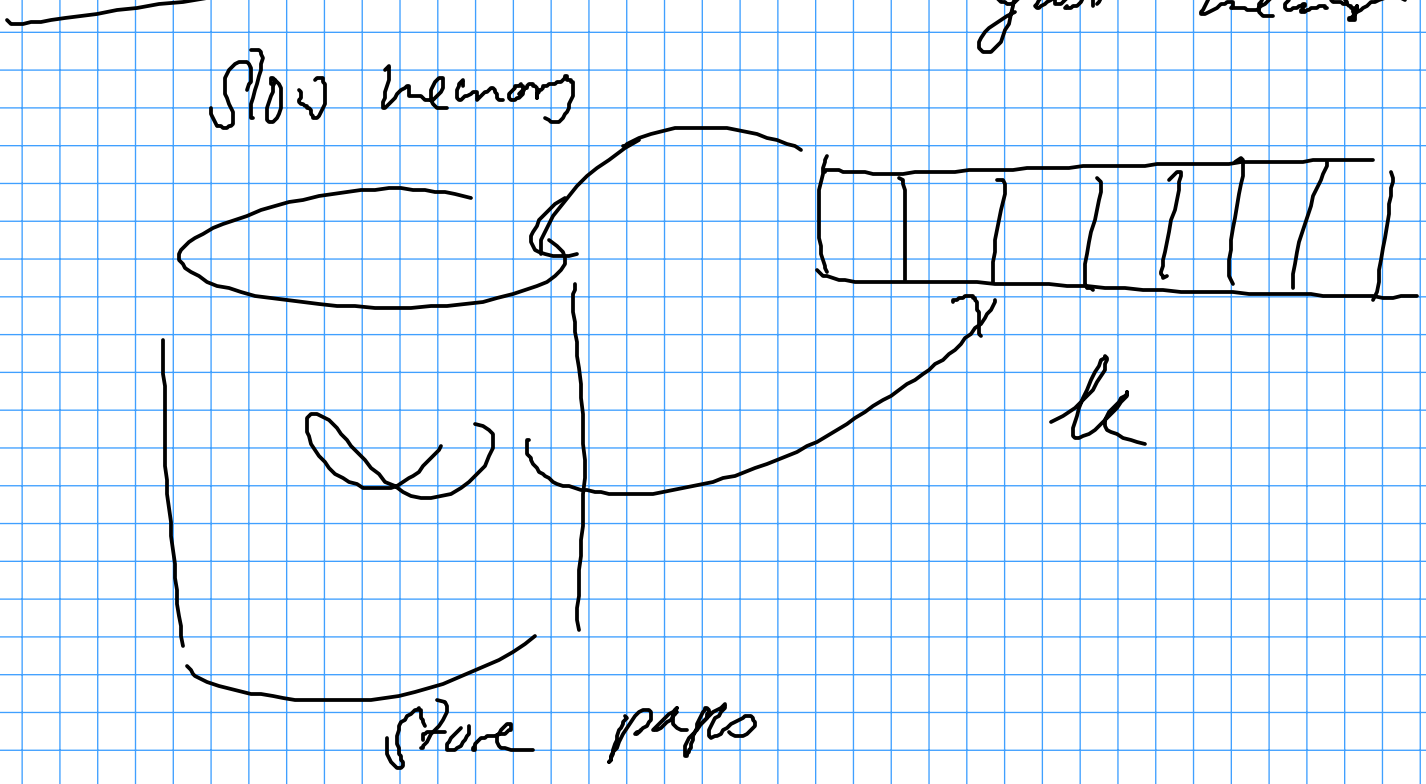
ONLINE ALGORITHMS

request sequence

$$\sigma = \sigma(1), \sigma(2), \ldots, \sigma(n)$$

comes in an online fashion, nothing is known about future requests, have to find strategies which perform good compared to an optimal strategy with 'a look in the future'

The Paging Problem



if a page is in fast memory at the time of request, the request is served immediately, otherwise:

- select page in fast memory
- evict it (put it in slow memory)
- load the desired page in fast memory

LRU

- Last Recently Used – evict the page that was accessed last of all pages in actual fast memory

- Least Frequently Used – evict the page in fast memory which has the lowest access count since it was loaded here

- Random
- FIFO – first in first out

Theorem: LRU and FIFO are k-competitive.

OPTIMAL STRATEGY 'MIN'

⇒ On a fault, evict the page whose next request occurs furthest in the future.

(Randomized alg. can achieve $2 \ln k$ competitiveness, while det. LB says no better than k-comp. possible)

Proof of Theorem for LRU:

Consider an arbitrary sequence of requests $\sigma = \sigma(1), \sigma(2), \ldots, \sigma(n)$.

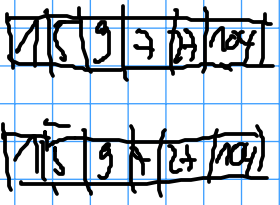
We will prove that $C_{LRU}(\sigma) \leq q \cdot C_{MIN}(\sigma)$.

W.l.o.g. we assume the fast memory to contain the same pages at the start of the algorithms.

We partition σ into phases P_i such that LRU has at most q faults in P_0 and exactly q faults in

every subsequence P_i. In the remainder we will show that the optimal strategy will have at least one fault in every phase.

P_0: The first fault for LRU implies that a page is requested which is not in fast memory. But as LRU and MIN start with the same pages here, this also implies a fault for MIN.



$5, 2, 7, 9$
at fault LRU

P_i: Let $\sigma(t_i)$ be the first request in P_i and $\sigma(t_{i+n-1})$ the last.
Furthermore let p be the last page requested in P_{i-1}.

Want to show: P_i contains requests to k distinct pages that are different from p.

if this is true, then MIN has to have a fault in p_i. (Because at the end of p_{i-1} the fast memory of MIN contains p for sure and any $k-1$ other pages, so k other requests force a fault).

p_i

$\sigma(t_i)\ \sigma(t)$ $\quad\quad$ $\sigma(t')\ \sigma(t_{i+1}-1)$

$\ldots | q | \ldots\ldots | q | \ldots$

Suppose for contradiction that LRU faults twice on the same page (q). Assume that LRU has a fault on $\sigma(t)=q$ and $\sigma(t')=q$ with $t_i \le t < t' \le t_{i+1}-1$

Page q is in fast memory right after $\sigma(t)$ is served. To create a fault at t' it needs to be evicted from fast memory at some time

t^* with $t < t^* < t'$.

When q is evicted, it is the least recently used requested page in fast memory. Therefore the subsequence $\sigma(t), \ldots, \sigma(t^*)$ has to contain two distinct requests to pages of which q must be different from q.

Finally suppose that with p_i LRU does not fault twice on a page but on one of the faults page p was requested. Let $t \geq t_i$ be the first time that p is evicted. Using the same argument as above we obtain a request sequence

$$\sigma(t_i - 1), \sigma(t_i), \ldots, \sigma(t)$$

which must contain k+1 distinct pages. //

⇒) LRU is k-competitive (and FIFO as well with very similar arguments)

Theorem: Let A be some deterministic online paging algorithm. If A is c-competitive, then $c \geq k$.

Proof: Let $S = \{p_1, p_2, \ldots, p_{k+1}\}$ be a set of $k+1$ arbitrary pages. We assume w.l.o.g. that A and OPT initially have $p_1, \ldots, p_k$ in fast memory.

Consider the following request sequence: Request always the page that is not in A's memory right now.

The algorithm A has a page fault on every request. Suppose that MIN has a fault on some request $\sigma(t)$, then it evicts a page that is not requested during the next $k-1$ steps. Thus, on any k consecutive requests, MIN has at most one fault.

From a practical point of view, LRU and FIFO do typically better if k increases. This is exactly the opposite of what is shown in the theoretical analysis.

Randomized Online algorithm

MARKING: The algorithm processes requests in phases, and marks some of them to remember accesses.

→ Whenever a page is requested, it gets marked (at the beginning of a phase everything is unmarked).

→ On a fault, a page is chosen u.a.r. from the unmarked ones to be evicted.

→ if all elements in fast memory are marked, start a new phase

→ and unmark all elements.

Theorem: MARKING is $2H_k \approx 2 \ln k$ competitive.

$$H_k = \sum_{i=1}^{k} \frac{1}{i} \qquad \ln k \leq H_k \leq \ln k + 1$$

Proof: Given a request sequence $\sigma = \sigma(1), \ldots, \sigma(n)$. W.l.o.g. we assume that MARKING al-

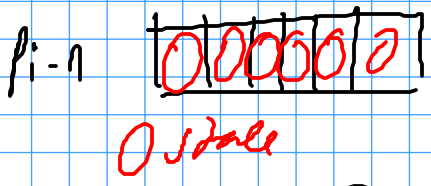
ready faults on $\sigma(n)$.

MARKING divides request in phases. A phase starting with $\sigma(i)$ ends with $\sigma(j)$ where $j \geq i$ and j is the smallest index such that the set

$$\{\sigma(i), \sigma(i+1), \ldots, \sigma(j+n)\}$$

contains $k+1$ distinct pages.

(As at the end of the phase all elements in fast memory need to be marked)



Consider an arbitrary phase P_i

Call a page stale if it is unmarked but was marked in P_{i-1}. Call a page clean if it is either marked or stale.

⇒ Going to show:

Let c be the # of clean pages requested in the phase, then:

- the amortized # faults made by MIN is at least $\frac{C}{2}$.
- the expected number of faults made by MARKING is at most CH_k.

The two statements imply the theorem as $\frac{CH_k}{\frac{C}{2}} = 2H_k$.

MIN

7	10	100	9	8

MARKING

5	100	7	17	1000

We first analyze MIN. Let S_{opt} be the set of pages contained in the fast memory, S_1 be the ones the MARKING stores.

$d_1 := |S_{opt} \setminus S_1|$ number of elements in S_{opt} but not in S_1 at the beginning of a phase

$d_1 = 5 - 2 = 3$

$d_F := |S_{opt} \setminus S_F|$ → at the end

MIN has at least $C - d_1$ faults during the phase.

(Because at least $c - d_I$ of c clean nodes are not in the fast memory of MIN).

Also, MIN has at least d_F fewer during the phase, because d_F pays requested during the phase are not in MIN's fast memory (at the end of the phase)

We conclude that MIN incurs at least

$$\max\{c - d_I, d_F\}$$

$$\geq \frac{1}{2}(c - d_I + d_F)$$

$$= \frac{c}{2} - \frac{d_I}{2} + \frac{d_F}{2}$$

Summing over all phases, the terms $\frac{d_F}{2}$ and $\frac{d_I}{2}$ telescope, except for the first and last term.

Final $\frac{d_I}{2}$ is 0 (because $S_{OPT} = S_0$)

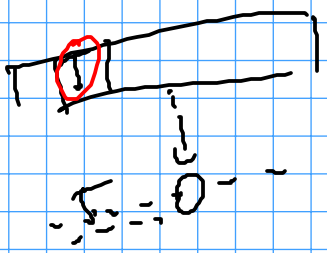
$$\Rightarrow \frac{c}{2} + \frac{d_F}{2} \geq \frac{c}{2}$$

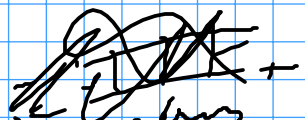
Next we analyze MARKING.

Serving c requests to clean pages obviously incurs c faults.

There are $S = k - c \leq k-1$ requests to stale pages. For $i = 1, \dots, S$ we compute the expected cost of the i-th request to a stale page. Let $c(i)$ be the # of clean pages that were requested in the phase before request #i. Let $s(i)$ denote the # of stale pages that remain before the request i.

$$s(i) = k - (i-1)$$





20 | 8 | 10 | 5

When MARKING serves the i-th request to a stale page, exactly $s(i) - c(i)$ of the $s(i)$ stale pages are in fast memory, and of them with

equal probability.

So the expected cost of the request i^{th}

$$\frac{s(i)-c(i)}{s(i)} \cdot 0 + \frac{c(i)}{s(i)} \cdot 1$$

$$\leq \frac{c}{k-i+1}$$

The total expected cost for serving requests to stale pages is

$$\sum_{i=1}^{k} \frac{c}{k+1-i} \leq \sum_{i=2}^{k} \frac{c}{i} = c(H_k - 1)$$

In total : cost c for clearing

cost $c(H_k - 1)$ for stale pages

$c H_k$ //

$c \log k$