sequence of requests
- alg. must handle each request as it comes
  versus off-line - get to look at whole request sequence first.
- usual scenario for data structures.
- but we will study situations where it makes sense to compare with "full info." solution, (i.e. best off-line).

Examples
- list accessing - e.g. Move to Front heuristic.
- paging - LRU, LFU, CS 341, 350
- splay trees - dynamic optimality conjecture.
- bin packing - First Fit, Best Fit.

Competitive Analysis - compare online algorithm with opt. off-line solution (even if opt is hard to find)
Alg. $A$ is $c$-competitive if exists constant $b$ s.t.
\[
A(\sigma) \leq c \cdot \text{OPT}(\sigma) + b
\]
sequence for minimization.

Note: we allow additive term $b$ always
(unlike for approx. alg.s.)
Paging
- fast memory, "cache", holds k pages
- slow memory, n pages, n >> k
when a page is requested
- if it's in cache, fine
- otherwise "page fault" must read it into cache
cost 1. Which page do we evict?

Goal: minimize cost

Optimum off-line strategy
- evict page whose next request is furthest in future.

Pf: not trivial
- modify any opt. soln. to this one, bit by bit
- without changing cost

Ex.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>B</th>
<th>D</th>
<th>C</th>
<th>E</th>
<th>A</th>
<th>B</th>
<th>E</th>
</tr>
</thead>
</table>

k=3

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>D</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>B</td>
</tr>
</tbody>
</table>

| B  | E  |

Ex. cost 3 evictions

Optimum

On line strategies
- FIFO - first in first out
- LRU - least recently used
- LFU - least frequently used

- keep track of most recent time each page in cache was used
- keep count of number of times each page in cache was used
LRU - least recently used

- like using OPT
  but pretend future = past.

**Example:**

\[
\begin{array}{c}
A & B & C & B & D & C & E & A & B & E & D \\
\frac{1}{k=3} & \frac{1}{B} & \frac{1}{D} & \frac{1}{D} & \frac{1}{A} & \frac{1}{E} & \frac{1}{D} & \frac{1}{E} & \frac{1}{D} & \frac{1}{B}
\end{array}
\]

- same ex. as previous page.

5 evictions

vs. 3 for OPT.

**Theorem [Sleator, Tarjan 1985]**

LRU and FIFO have competitive ratio \( k \).

But LRU is better in practice.

**Proof:** Divide request seq. into phases

\[
\begin{array}{c}
k=3 \\
\text{cache:} \quad \text{ABACBACADADEDDB} \\
\text{phase 1:} \quad \text{ABAC} \\
\text{phase 2:} \quad \text{BACADADEDDB}
\end{array}
\]

- a phase stops just before we see \( k+1 \) different pages.

The algo uses \( \leq k \) swaps per phase.

- because LRU and FIFO will not evict a page used in that phase. (check this out)

Now consider OPT. Let \( p_i \) be first page requested in phase \( i \).

**Claim:** For requests in phase \( i \) \( \cup \) \( \mathcal{E} \) \( p_{i+1} \), OPT uses \( \geq 1 \) swap.

**Proof:** After request \( p_i \), \( p_i \) will be in cache.

- There are \( k+1 \) distinct pages in phase \( i \) \( \cup \mathcal{E} \) \( p_{i+1} \)

and they don’t all fit in the cache.

**Therefore:** \( \text{ALG/OPT} \leq k + \text{term} \),

for partial phase at end.
Any deterministic alg. has competitive ratio $\geq k$. 

If adversary argument:

- $k =$ cache size
- $\#$ pages $= k+1$
- Adversary always asks for page not in cache.
- $n =$ length of sequence.
- $n =$ number of swaps.
- An offline solution that evicts the page with max next request time uses $n/k$ swaps.
- Because each time we evict, we're good for next $k$ requests. So $\text{OPT} \leq n/k$.
- and $\text{Alg}/\text{OPT} \geq k$.

Note that there are only $k+1$ pages!

A Randomized "Marking" Algorithm to serve request for page $P$.

- If $P$ not in cache:
  - If all pages in cache are marked, then unmark all.
  - Choose a random unmarked page to evict & move $P$ in.
  - Mark $P$.

---

Expected competitive ratio is $O(\log k)$.

Proof: As before, divide the request sequence into maximal phases in which $k$ different pages are requested.

- At the beginning of a phase, all pages in cache are unmarked.
- The $k$ requested pages will not be evicted in the phase.
- And will be marked by the end of the phase.

Let $S_i =$ pages in cache at start of phase $i$. 
distinguish request for page $p$ as:
old if $p \in S_i$, new otherwise

Let $n_i = \# new requests in phase $i$
these all cost 1. (i.e. a page is evicted)

What is the expected cost of old requests?
(when we "unluckily" evict a page of $S_i$ and then need it back)

Consider the first old request, say for page $p$.
There are $\leq n_i$ new pages $\Rightarrow \leq n_i$ pages were evicted (at random)

Prob $[p$ was evicted$] \leq \frac{n_i}{k}$

More generally, let $p_1, p_2 \ldots$ be the distinct old page requests (in order)

When $p_{j+1}$ is requested $p_1 \cdots p_j$ are in the cache and are marked. The worst case is that all $n_i$ new page requests happen before $p_{j+1}$ and cause $n_i$ of the remaining cache slots to be marked.

Thus

Prob $[p_{j+1}$ not in cache$] \leq \frac{n_i}{k-j}$

(Pj+1 was evicted)
Sum over $j = 0 \ldots k-n_i-1$. Expected cost of old requests

\[
\leq n_i \left( \frac{1}{k} + \frac{1}{k-1} + \ldots + \frac{1}{k-(k-n_i-1)} \right)
\]

\[
\leq n_i \left( H_k - 1 \right)
\]

Adding the $n_i$ cost of new pages:

To complete the proof we must give a lower bound on $OPT$

Claim $OPT$ costs $\geq \frac{1}{2} \sum n_i$

If in phase $i$ and $i-1$ at least $k+n_i$ distinct pages were requested (the $k$ in the cache at the start of phase $i$ and the $n_i$ new requests)

Thus, except for $i=1$, $OPT$ makes $\geq n_i$ page-faults in phase $i$ and $i-1$

Note: $n_i$ was defined relative to $S_i$ (what the Alg. had in the cache) so we cannot just claim that $OPT$ must use $n_i$ faults in phase $i$)

Also $\#$ faults in phase $i$ is $\geq n_i$ (we charge for reading pages into empty cache)

Thus total cost is $\geq \frac{1}{2} \sum n_i$

Thus $\sum n_i \leq 2 \cdot OPT$

Competitive ratio: Alg. costs $H_k \cdot \sum n_i \leq 2 \cdot H_k \cdot OPT = O(\log k) \cdot OPT$

FACT: There is a (nearly) matching lower bound. It uses an adversary argument where we assume that the adversary does not know the coin tosses.