Paging

Setting

- n items, cache size k.
- For $i=1,\dots,m$, i^{th} request $\sigma_i \in [n]$ comes
  * if it's in the cache, "hit" — pay 0
  * if not, "miss" — **evict an item** from the cache, bring σ_i in, and pay 1.
- How to evict items to minimize (# misses)?
- If entire σ was known, "evict the item used furthest in the future" is optimal!
  * Of course, not implementable without knowing the future.
- So, will design online algos with small competitive ratio.
- For simplicity, assume that
  * the cache is full in the beginning
  * first request is a miss

} WLOG, since we can just consider the process from first miss.

Lower Bounds for Det. Algos.

__Claim.__ Every deterministic algorithm's competitive ratio is at least k.

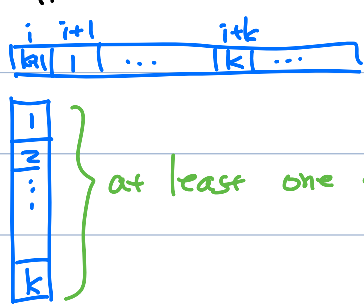
__Pf.__ Let $n = k+1$. At each stage, σ_i is the item not in the cache. So $ALG = m$.

(For a lower bound against det. algos, the adversary can choose σ_i based on what ALG did so far)

For OPT, since it evicts the item used furthest in the future and there are $n = k+1$ items, if OPT misses on σ_i, the next miss will happen no earlier than $i+k$.

requests =

cache at time i =



at least one of them won't be requested until $i+k$.

So, $OPT \le m/k$ $\square$

1-bit LRU (Least Recently Used)

Algorithm

- Works in phases.
- Maintains a single bit for each item i in the cache. — marked/unmarked
  (all unmarked in the beginning of a phase)
- Given σ_i,
  1. hit: mark i.
  2. miss: (i) $\exists$ unmarked item j in the cache: evict j, bring i in and mark i.
     — arbitrarily (deterministically)
     (ii) all items in the cache are marked: unmark all items and begin a new phase.

Example, $k=3$, $n=5$

requests

4	5	1	4	1	2	5	...

cache (right before request)

1	4	4	4	4	4	2	...
2	2	5	5	5	5	5	...
3	3	3	1	1	1	1	...

Phase 1. Phase 2.

Facts about phase i. call them $T_i \in [n]$.

- there are exactly k distinct requests in the phase, which is different from the request right after the phase.
- for each $j \in T_i$, only first request in the phase "matters"; second, third, ..., request for j in the phase are all hit.

Claim. ALG is k-competitive.

Pf. ALG misses at most k times in a phase.

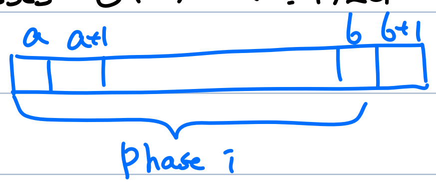
For OPT, consider phase $i = (\sigma_a \dots \sigma_b)$

By definition, there are $k+1$ distinct items in $(\sigma_a \dots \sigma_{b+1})$.

Then OPT must miss at least one of $\sigma_{a+1} \dots \sigma_{b+1}$.

$\Rightarrow$ OPT makes at least one miss in phase i shifted by 1 to right.

(And OPT misses σ_1) $\therefore$ ALG $\leq k \cdot$ OPT



$\square$

Randomized 1-bit LRU

Algorithm

- Works in phases.
- Maintains a single bit (marked/unmarked) for each item i in the cache.
  (all unmarked in the beginning of a phase)
- Given σ_i,
  ① hit: mark i.
  ② miss: (i) $\exists$ unmarked item j (randomly choose among all unmarked cache items.) in the cache: evict j, bring i in and mark i.
  (ii) all items in the cache are marked: unmark all items and begin a new phase.

Example, $k=3$, $n=5$

requests

4	5	1	4	1	2	5	...

cache (right before request)

1	**4**	**4**	**4**	**4**	4	**2**	...
2	2	**5**	**5**	**5**	5	5	...
3	3	3	**1**	**1**	1	1	...

Phase 1. Phase 2.

Facts about phase i. (unchanged!), call them $T_i \in [n]$.

- there are exactly k distinct requests in the phase, which is different from the request right after the phase.
- for each $j \in T_i$, only first request in the phase "matters"; second, third, ..., request for j in the phase are all hit.
- Phases do not depend on random choices of ALG!.

Claim. ALG is $O(\log k)$-competitive

Pf. Let $S_i \subseteq [n]$ be the cache at the beginning of phase i.
(S_0, $S_i = T_{i-1}$ $\forall i > 1$.) Let $\Delta_i = |S_{i+1} \setminus S_i| = |S_i \setminus S_{i+1}|$
Let $S_i^* \subseteq [n]$ be the cache of OPT at the beginning of phase i. Let $\phi_i = |S_i \setminus S_i^*| = |S_i^* \setminus S_i|$

Upper bounding ALG in Phase i.

(i) For request of $j \in S_{i+1} \setminus S_i$, we miss once (only the first one)

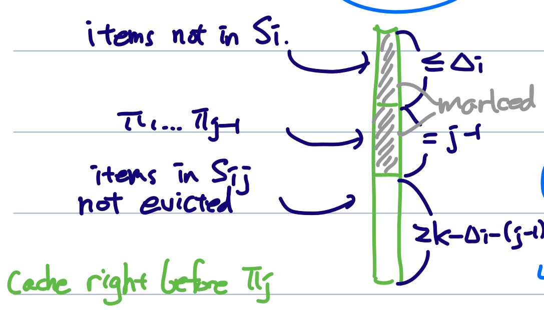
(ii) For items in S_i requested in Phase i, call them $\pi_1 \ldots \pi_\ell$ in increasing order of first request time in phase i.

$$\Pr[\pi_j \text{ hits}] \geq \frac{k - \Delta_i - (j-1)}{k - (j-1)}$$

right before π_j, consider $S_{ij} := S_i \setminus \{\pi_1 \ldots \pi_{j-1}\}$.

① the set of unmarked items in the cache is a subset of S_{ij}. (size $\geq k - \Delta_i - (j-1)$)

② each element of S_{ij} has same probability of remaining in the cache.



③ So $\Pr[\pi_j \text{ hits}] = \Pr[\pi_j \text{ remains in the cache}]$
$\geq (k-\Delta_i-(j-1))/(k-(j-1))$.

$1+1/2+\dots+1/k$.

$\therefore \Pr[\pi_j \text{ misses}] \leq \frac{\Delta_i}{k-j+1}$ and $\mathbb{E}[(\#\text{ misses of type }(ii))] \leq \Delta_i H_k$

$\therefore \mathbb{E}[(\#\text{ misses in Phase } i)] \leq \Delta_i(H_k+1)$.

Lower bounding OPT in Phase i.

Let $OPT_i = (\#$ of misses of OPT in Phase $i)$

Then $OPT_i \geq |S_{i+1} \setminus S_i^*|$. (S_{i+1} is items requested in phase i, so OPT will miss items in $S_{i+1}\setminus S_i^*$ in the phase!)

① $|S_{i+1} \setminus S_i^*| \geq |S_{i+1} \setminus S_i| - |S_i^* \setminus S_i| = \Delta_i - \phi_i$.

(exercise: if A, B, C are sets of size k and $d(A,B) := |A\setminus B| = |B \setminus A|$, then
$d(A,B) + d(B,C) \geq d(A,C)$.)

② $|S_{i+1} \setminus S_i^*| = |S_i^* \setminus S_{i+1}| \geq |S_{i+1}^* \setminus S_{i+1}| = \phi_{i+1}$

in Phase i, only items in S_{i+1} are requested

so S_{i+1}^* must be "closer" to S_{i+1} than S_i^*.

So, $OPT_i \geq \frac{1}{2}(\phi_{i+1} + \Delta_i - \phi_i)$ and summing over i
and using $\phi_1 = 0$ yields $OPT \geq \frac{1}{2}\sum_i \Delta_i$.

$\therefore ALG/OPT \leq 2(H_k+1) = O(\log k)$ □