ONLINE ALGORITHM

Online Algorithms vs Offline Algorithms

Let x be a sequence of operations (inputs)

- Offline algorithm knows "exactly" what x is, therefore has the optimal solution

$$\underbrace{o_1 \quad o_2 \quad o_3 \quad \dots \quad o_n}_{} \longrightarrow \text{sequence } x$$

OPT(x) is the optimal algorithm for sequence x

- Online algorithm doesn't know in advance what x is, so doesn't have the optimal solution

$$\underbrace{o_1 \quad o_2 \quad o_3}_{} \quad \dots \quad ? \longrightarrow \text{sequence } x$$

$A(x)$ is the online algorithm on sequence x

Goal of Online Algorithm

- Design an algorithm / strategy $A(x)$ s.t. its cost is as close to the optimal algorithm OPT(x) as possible.
- Formally: Minimize "competitive ratio" α in

$$A(x) \leq \alpha \cdot \mathrm{OPT}(x) + \text{Const}$$

where $\begin{cases} \text{sequence } x \text{ is finite} \\ \alpha \text{ can depends on } \mathrm{len}(x) \end{cases}$

↳ This formal setup is called competitive analysis

Example: Paging and Caching

- Cache has "pages", when all pages are full, decide which page to evict (without knowing the future operations)
  → online algorithm
- Some algorithms on this problem: LRU, FIFO, etc...
- OPT algorithm: assume we have the ability to "see into the future" and know which page will be used, then the optimal eviction straty is to "evict page that is furthest in the future".

- Bad news: No good deterministic algorithm is $< k$-competitive
  - → Adversary:
    Entity that generates an access sequence that only consists of item not in cache
    - ⇒ Imaginary deterministic alg will spends n evicts
  - → OPT:
    Recall, OPT evicts item that won't be accessed furthest into the future
    - ⇒ For cache size k, the evicted item won't be accessed again for at least k time units (one cache miss for every k accesses)
    - ⇒ Costs n/k (# cache misses for n accesses)

Conservative Algorithm.

Costs $\leq k$ in sequence $\leq k$ distinct elements

Example: $\underbrace{a \quad b \quad c \quad b}_{\leq k \text{ distinct elements}} \quad e \quad d \quad \ldots$

⇒ LRU and FIFO are conservative algorithms

Claim 1: Any conservative algorithm is k-competitive

Claim 2 (Stronger): Any conservative algorithm, compared to "crippled" offline OPT with h-page cache ($h \leq k$), is $\frac{k}{k-h+1}$ competitive

↳ If OPT cache size is $h = k/2$, then competitive ratio $\approx < 2$ (almost constant)

Proof: $A_k(x) \leq \frac{k}{k-h+1} \cdot OPT_h(x)$

Set up "phases" as maximal subsequence with k distinct elements

$a \; b \; b \;|\; c \; b \;|\; a \; e \;|\; f \; \ldots \qquad k = 2$

1. Show $A_k(x) \leq k \;\; \forall$ phases

   By definition of conservative algorithm, $A(x) \leq k \;\; \forall$ phases

2. Show $OPT_h(x) \geq (k-h+1) \quad \forall$ phases

Setup: accessing the first element in an arbitrary phase

$$\cdots \underbrace{\Big|\; x \;\cdots \;\Big|}_{\text{phase } i} \cdots$$

- Consider $h = k$:

  Case 1: $x \notin$ OPT's cache

  $\Rightarrow$ pays 1

  Case 2: $x \in$ OPT's cache

  $\Rightarrow$ previous phase $i-1$ use only $h-1$ space in cache

  $\Rightarrow$ pays 1

- Consider $h < k$:

  Only $\leq h-1$ items can be stored in OPT's cache (aside from x) out of $k-1$ remaining items in phase i

$$\cdots \underbrace{\Big|\; x \;\overbrace{\underbrace{\cdots}_{h-1} \;\cdots}^{k-1 \text{ items}} \;\Big|}_{\text{phase } i} \cdots$$

  $\Rightarrow$ pays for the $(k-1)-(h-1)$ items that are not stored

  $\Rightarrow$ cost $\geq k-h$

Randomized Online Algorithm

- Main idea : "fool the adversary by flipping coin, instead of deterministic decision".
  - In the caching example :
    - Deterministic Alg vs Adversary : always lose bc the Adversary will always pick that same item to access next.
    - Randomized Alg vs Adversary : kick out a random item
      → higher chance of winning since the Adversary doesn't know "exactly" which item to access next.

- Types of Adversary :
  Designing such Randomized Alg depends on the types of Adversary
  1. Oblivious Adversary :
     Doesn't see your coin flip , know your algorithm
  2. Fully Adaptive Adversary :
     See all the coin flip (deterministic) → not very interesting
  3. Adaptive Adversary :
     know previous coin flip , but not the future coin flips
     ↳ resemble online alg

* Intuition to understand Randomized Online Alg

Consider any zero-sum game , say rock-paper-scissor

player A \ player B	R	P	S
R	–	B	A
P	A	–	B
S	B	A	–

⇒ Best strategy for both players is to uniformly choose at random

⇒ Same can be said for the case with Online Algorithm vs Adversary

⇒ Therefore, we make the Online Algorithm "Random"

Randomized α-competitive

For all inputs x:

$$\mathbb{E}[\text{online\_alg}(x)] \leq \alpha \cdot OPT(x)$$

expected value since we are randomizing

Paging and Caching (with Randomized Alg)

- Marking Alg:

  When page fault (accessing smt not in cache)

  - If $\forall$ pages marked:

    unmark all

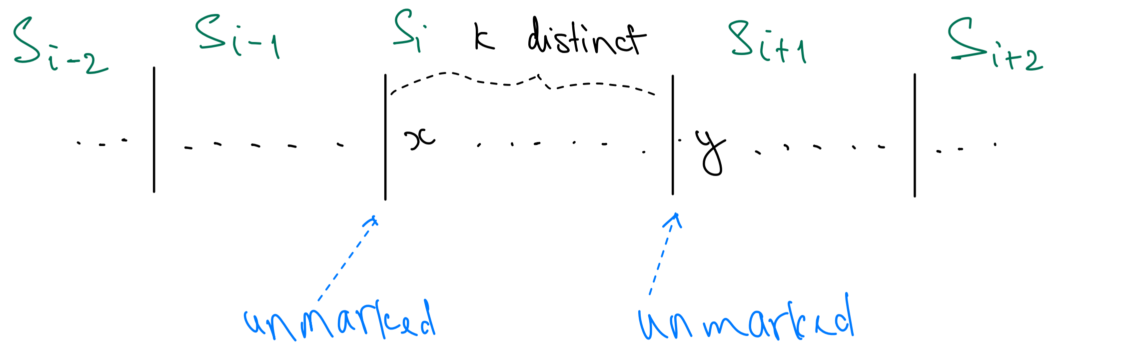
  - Evicts random unmark page
  - New page is marked

  When non-fault (access item already in cache)

  - Mark it

  $\hookrightarrow$ $O(\lg k)$ - competitive

- Effect of Marking Alg on phases:



  - If x is accessed again in the same phase, it is guarantee to be in the cache.

    $\Rightarrow$ pays only for the first time we see an item

  - At the end of a phase, the k distinct items will be in cache.

$$S_i = \{ k \text{ items including } x \}$$
$$S_{i+1} = \{ k \text{ items including } y \}$$
$$\vdots \qquad \vdots$$