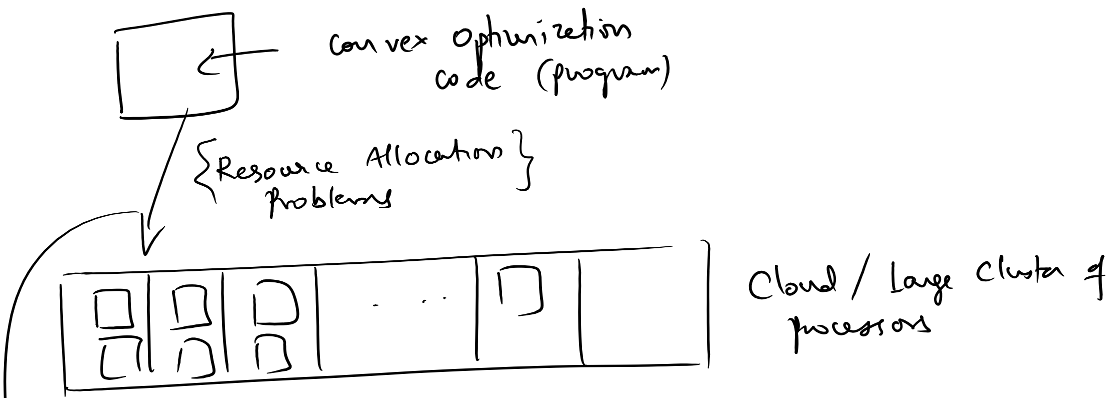
Lecture 23/03

Wednesday, March 23, 2016 1:48 PM

Online Paging + Online Set Cover



Heavily online, Applications arrive & depart freely

- → Algo is unaware of input ahead of time
- → Maintain feasible solutions always
- → Few changes over time (can't entirely reoptimize each time)
- → "competitive", (i.e.) be reasonable for underlying objective function

---

Paging

k cache slots, need policy of what to evict when cache = full?

↳ Well motivated in online setting

3 pages A, B, C

2 cache slots

Offline (traditional) : Know exact page requests

Ex

AA B C A A BBB C A CC

How to maintain cache so that as few cache misses

2 cache slots

A	B

when C arrives, evict B

A	C

when B again comes, evict A

B	C

when A comes, evict B

3 evictions ← OPT!

Goal : Minimize # evictions.

|B|C|
when A comes, evict B
|A|C|

Use flow algorithms to determine optimal eviction policy for a given sequence of request

Opt Alg ≡ "farthest into future"

Model is flawed → don't know future requests !!

Want online algo, minimize # evictions !!

LRU : least recently used. (Intuition is programs have locality of reference)

↓ How good is this algo?

How to measure "goodness" of online algorithms?

↳ Competitive Ratio of Alg ≡ Worst case ratio (over all possible input sequences σ) $\dfrac{E[\#\text{ misses of } Alg(\sigma)]}{\text{Optimal } \#\text{misses}(\sigma)}$

"offline quantity"

"information gap" between NR & DR

What is competitive Ratio of LRU

k cache slots

consider the σ (input sequence) $= 1, 2, 3, \ldots, k, \underbrace{k+1, 1, 2, 3, \ldots, k}_{\text{all cause eviction !!}}$

Cache = empty initially

$|\cancel{1}|\cancel{2}|\cancel{3}| \ldots |\cancel{k}|$; $\cancel{k+1}$, 1, 2, k-1, k

#evictions (LRU on σ) $= k+1$

$Opt(\sigma) = |1|2|3|\ldots|\cancel{k}|$; $\cancel{k+1}$, k $= 2$

$\boxed{\text{Competitive Ratio (LRU)} \geq \frac{k}{2}}$ (since its max over all σ)

$\boxed{\text{Thm :- Competitive Ratio (LRU)} \leq k}$

Randomization of LRU

↓

MARKING ALGORITHM

MARKING ALGORITHM.

Suppose cache has already k pages in it initially, call each of them unmarked

When a request arrives,

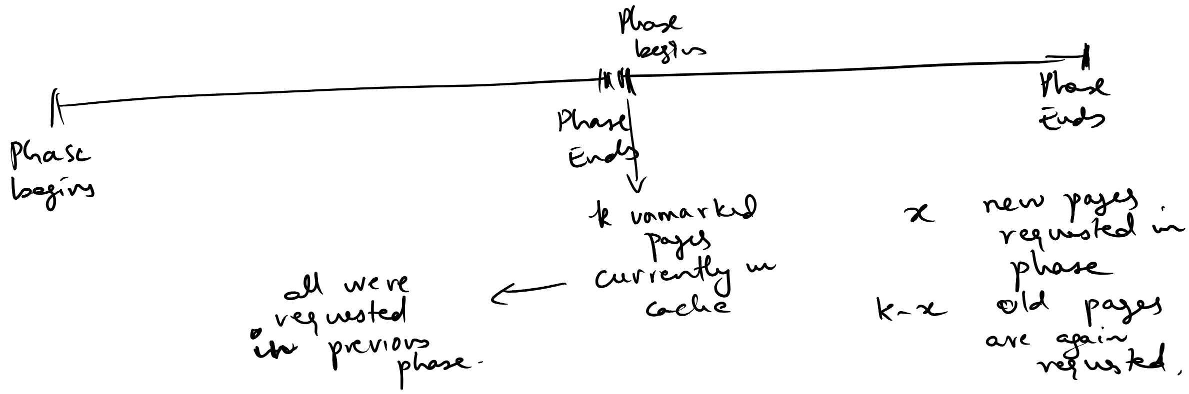
a) If its in cache, serve it, mark that page.

b) if its not in cache, bring it in, mark it, evict a random unmarked page

c) If all pages are marked, reset markers to unmarked.

Whats the competitive ratio of marking? ← $O(\ln k)$ competitive ratio

Proof Sketch



Hand wavy fact } Marking is hurt most when old pages are requested after new pages

Worst Case

Expected # cache Evictions :-

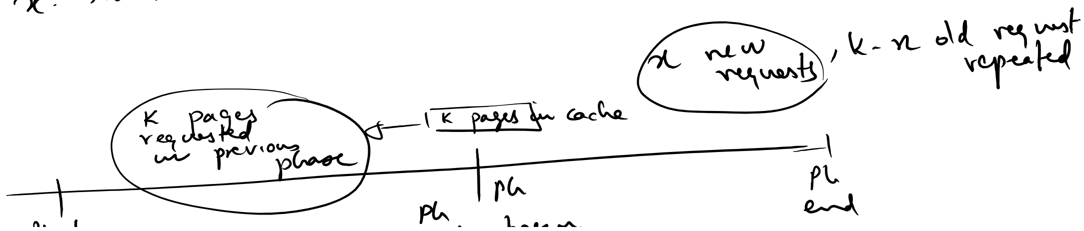
$$\approx x + \left(\frac{x}{k}\right) + \left(\frac{x}{k-1}\right) + \left(\frac{x}{k-2}\right) \cdots + 1$$

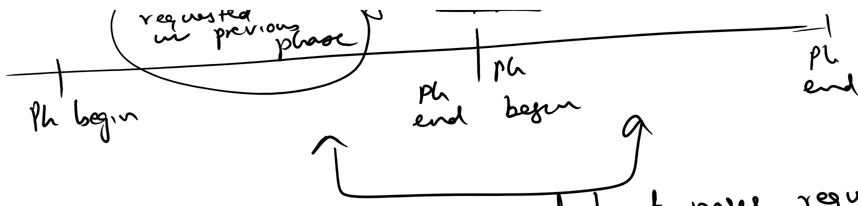
Pr(first old page was thrown out by the x new pages that entered)

Pr(2nd old page was thrown out before it arrived)

Pr(3rd old page was thrown out before)

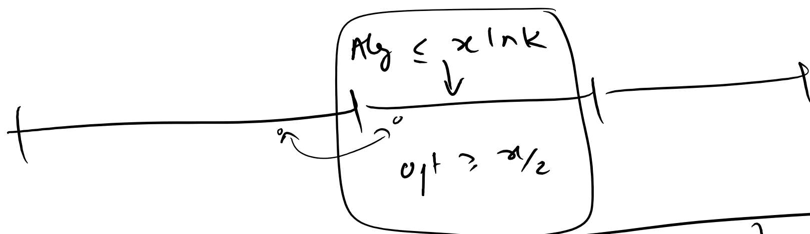
$$\simeq x \cdot \ln k$$





$k + x$ distinct pages requested in sequence

Opt has to evict at least x pages across two phases!



$$\sum_{\text{phases}} E\left[\frac{\#\text{ evictions of Alg in current phase}}{\#\text{evictions of opt in current phase}}\right] \le 2\ln k.$$

Marking outperforms LRU (exponentially better) according to competitive ratio

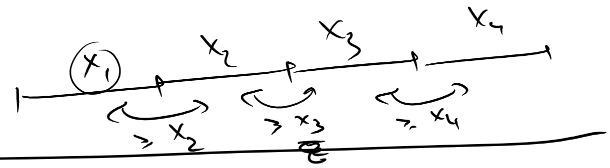
Ex: Any randomized Algo has competitive ratio $\geq \Omega(\ln k)$

{ Hint: Coupon Collector }

x_i be # new requests in phase i

$E[Alg] \le \sum x_i \ln k$

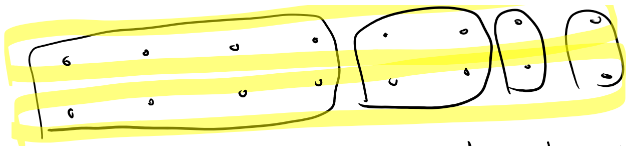
$Opt \geq \frac{\sum x_i}{2}$



Online Set Cover

- Collection of sets (m)
- Collection of Elements (Universe of elts) (n)
- Pick min # sets to cover universe

Alg 1: Greedy (Approx Ratio is $\ln n$)



Opt = 2, Greedy = $\underbrace{1+1+1+\cdots+1}_{=\log_2 n}$ | ratio = $\frac{\log n}{2}$

Alg 2: LP rounding

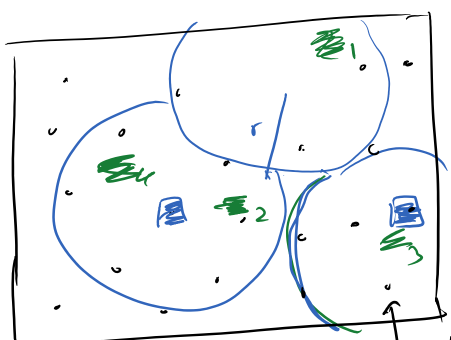
$$\boxed{\begin{aligned} &\text{Min } \sum x_S \\ &\sum_{S:\, e \in S} x_S \geq 1 \\ &x_S \geq 0 \end{aligned}}$$

⟷ compute optimal fractional solution → round it (randomized rounding) → $\boxed{\text{Pick set } S \text{ w.p. } x_S^* \log n}$

Greedy is inherently offline

Q: Online set cover: elements needing connectivity only show up over time!

We need to maintain a good set cover for the elements that have currently shown up.



↑ all potential m locations where Airtel can install cell towers.

Alg knows m possible sets at time 0

→ time 1: element e_1 arrives
Alg picks a set S_1 to cover e_1

time 2: element e_2 arrives
Alg picks new set if existing sets don't cover e_1

time 3: element e_3 arrives
Alg picks new set if previously included sets don't cover e_3

⋮

time T

Incrementally builds a set cover solution

---

time T

How good is an Algo?

Competitive Ratio : $\max\limits_{\sigma} \dfrac{E[\text{\# sets bought by online Alg for } \sigma]}{\text{\# sets which Optimal set cover would buy}}$

Candidate Algo :- (CG)

when new elt arrives, if uncovered, include set which can most elts.

$S_1 = \{e_1, e_{100}, e_{101}, e_{102}, \ldots e_{199}\}$

$S_0 = \{e_1, e_2, e_3, e_4, e_5\}$

$S_2 = \{e_2, e_{200}, e_{201}, e_{202}, \ldots e_{299}\}$

$S_3 = \{e_3, e_{300}, e_{301}, \ldots e_{399}\}$

$S_4 = \{e_4, e_{400}, e_{401} \text{ —— '' —— } e_{499}\}$

$S_5 = \{e_5, e_{500} \text{ —— '' —— } e_{599}\}$

e_1 arrives
↓
CG picks S_1
e_2 arrives
↓
CG picks S_2
↓
$e_3 \rightarrow$ CG picks S_3

CG = 5 for sequence
opt = S_0 = 1 for sequence

Instead alternate Meta Algo

Maintain utility of sets (Confidence that these sets are useful)
↓
every time new elt comes, update the confidence/utility of every set
↓
include the most useful set

Algo

Fractional Algorithm → Rounding Algorithm.

maintain online LP solution

Min $\sum x_s$

$$\sum_{S: e \in S} x_S \geq 1 \quad \forall e$$

$\forall e$ ← currently arrived elements

a) new elt brings new constraint

a) new elt brings new constraint

arrived elements

b) Want the x_S values to be monotonically non-decreasing } ← "relaxation of constraint that a set, once bought, can't be unbought later"

$x_S = 0.1 \rightarrow x_S = 0.2 \rightarrow 0.2 \rightarrow 0.5 \rightarrow 0.9 \rightarrow 1$ ✓

$0.5 \searrow 0.2 \nearrow 1$ ✗

Algo

Min $\sum x_S$

$\sum_{S: e \in S} x_S \geq 1 \quad \forall$ arrived e

Max $\sum_{\text{arrived } e} y_e$

$\sum_{e \in S} y_e \leq 1 \quad \forall$ sets S

Algo

Initialize all $x_S = \frac{1}{m}$

when new element e arrives,

while $\left(\sum_{S: e \in S} x_S < 1\right)$

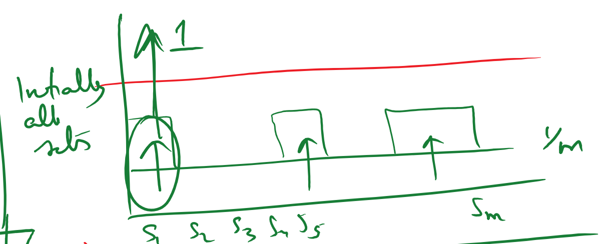
update $x_S \leftarrow 2x_S$ if S covers 'e'.

(update $y_e \leftarrow y_e + 1$ for analysis)

($\sum x_S =$ LP value $= 1$ initially)

Cleverness :-

If a set is repeatedly useful, its x value jumps up to 1 very quickly



---

next class ← recap algo, analyze the competitive ratio of LP

Thm: For any sequence,

$$\frac{\text{LP cost (fractional)}}{\text{Opt \# Sets}} \leq O(\log m)$$

↓

rounding 6

Online Matching