COL758: Advanced Algorithms **Spring 2019**

Lecture 22: April 12

Lecturer: Naveen Garg *Facilitator: Navin Garg*

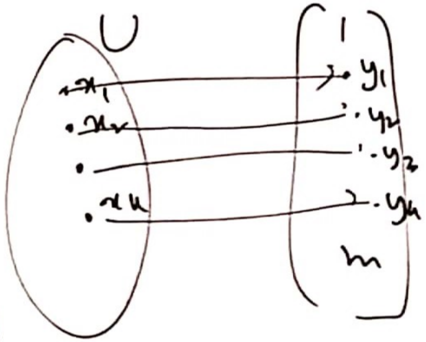
$\mathcal{H}$ is a family of k-independent hash fns.

$\mathcal{H} = \{ h_1, h_2 \dots \}$

$h_i : U \to [m]$

$\mathcal{H}$ is k-independent if for any k elements $x_1, x_2 \dots x_k \in U$ & $y_1, y_2 \dots y_k \in [m]$

$$\Pr_{h \in \mathcal{H}}\left[h(x_1) = y_1 \wedge h(x_2) = y_2 \wedge \dots h(x_k) = y_k\right] = \frac{1}{m^k}$$



$x_1, \dots x_{k-1}, \; y_1 \dots y_{k-1}$

$$\Pr\left[h(x_1) = y_1 \dots h(x_{k-1}) = y_{k-1}\right] = \sum_{y_k=1}^{m} \Pr\left[\begin{matrix} h(x_1) = y_1 \\ h(x_2) = y_2 \\ h(x_k) = y_k \end{matrix}\right]$$

$$= m \cdot \frac{1}{m^k} = \frac{1}{m^{k-1}}$$

is equivalent to the conditions

① $\forall x \in U, y \in [m] \quad \Pr_{h \in \mathcal{H}}[h(x) = y] = \frac{1}{m}$

② $\forall x_1, x_2 \ldots x_k \in U$, h picked randomly from $\mathcal{H}$, $h(x_1), h(x_2) \ldots h(x_k)$ are independent random variables

$p = m$

$h(x_1) = y_1$

$h(x_2) = y_2$

$\vdots$

$h(x_k) = y_k$

$|U| < p$

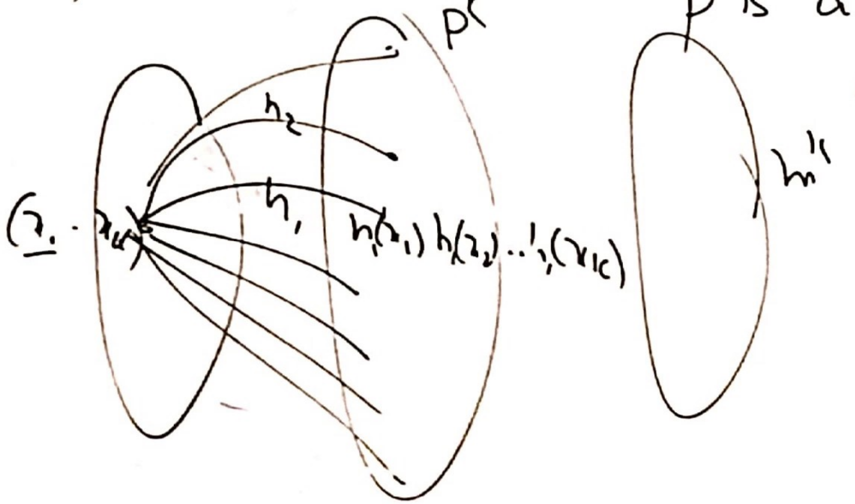
$\Pr[(h(x_1) = y_1) \wedge (h(x_2) = y_2)] = \Pr[h(x_1) = y_1] \Pr[h(x_2) = y_2] = \frac{1}{m^2}$

mod p

$$h_{a_0 a_1 \ldots a_{k-1}}(x) = \left(\sum_{i=0}^{k-1} a_i x^i \mod p\right)$$

p is a large prime

$a_i \in_R [0 \ldots p]$.



Online Algorithms.

Skirental

rent/day = 1 input σ = S S S S S S S N N ✓

buying = B. output R R R B

deterministic algorithms :

Alg 1: rent for B days & then buy (if season continue).

if season lasts for exactly B days then algorithm spends B+B while opt is B.

and for this input the competitive ratio is 2 & this is the worst input

if ski season ran for R days

then $opt = \min(R, B)$

$$\text{Our solu} = \begin{cases} R & R \leq B \\ 2B & R > B \end{cases} \leq 2\min(R, B)$$

is it possible to get a better deterministic algorithm.

Consider a deterministic algorithm which rents for k days & then buys.

adversary chooses input seq. in which ski season ends after k days.

cost of deterministic alg = $k+B$

randomised alg.

cost of opt solu = $\min(k, B)$.

$\frac{e}{e-1}$

Competitive ratio = $\frac{k+B}{\min(k,B)} \geq 2$.

line Algorithms

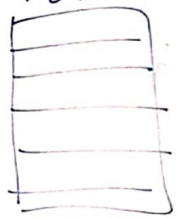
Paging.

n pages in all

$k < n$.

$r_i \in [n]$

main memory



k pages

i^{th} request to pa

M_i set of pages in cache at i^{th} step

input $r_1, r_2 - - r_n$ r

at i^{th} step if $r_i \notin M_i$ then find a page in M_i to evict

$Cost_A(\sigma)$ = # of page faults incurred by A on input σ.

Comp ratio of A $= \max_{\sigma} \dfrac{Cost_A(\sigma)}{Cost_{OPT}(\sigma)}$.

Deterministic

If opt has no fault in Phase 1 then neither does LRU

LRU (Least recently used)

Divide the request sequence into phases. In each phase (except 1st) LRU incurs exactly k page faults

LRU has < k faults

k

$r_1\ r_2 \cdot r \mid r_{p_{i-1}} \quad r_{p_i - 1} \mid r_{p_i} \cdots\cdots r_{n-1}\ r_n$

1st phase

phase i

phase

Claim 1:

phase i, has requests to $\geq$ k distinct pages different from $\bar{r}$

⇓

# of page faults in OPT $\geq$ (# phases − 1)

of ... in LRU = (# phases − 1)k

$\bar{r} = 7$ 1 3 2 4 7 5 7

LRU is k-competitive.

Lower bound on deterministic algorithms. Let A be a deterministic algorithm

$n = k+1 \quad p_1 \ldots p_{k+1}$

$H_k = 1 + \frac{1}{2} + \ldots \frac{1}{k}$

$\sigma =$ at each step request page that is not in algorithms cache.

$cost_A(\sigma) = |\sigma|$

...no of $A \geq k$.

$cost_{OPT}(\sigma) =$

$\sigma = \sigma_1, \sigma_2 \ldots \sigma_k \mid \bar{\sigma}$

Randomised algorithm

MARKING Algorithm $\quad 2H_k \simeq 2 \ln k$

if page is in cache then mark the page
else evict a random unmarked page

Reset marks when all pages are marked.

3, 5, 4