

Online algorithms

Lecture 1

Probability

Events, probabilities,
conditional probabilities,
union bound

Random variables,
expected value,
linearity of expectation
Markov's inequality

Algorithms

Approx. 3-SAT,

Big idea: we only want the average

$$\begin{aligned} \text{Linearity of Expectation} \\ E[X+Y] &= E[X] + E[Y] \\ \text{"average sum = sum of averages"} \end{aligned}$$

let $Y_i = \begin{cases} 1 & \text{if } i\text{th clause satisfied} \\ 0 & \text{otherwise} \end{cases}$ ($i=1, \dots, m$)

$$E[Y_i] = 7/8 \quad (1 \text{ clause case})$$

$$E[\text{\# clauses satisfied}] = E\left[\sum_{i=1}^m Y_i\right] = \sum_{i=1}^m E[Y_i] = 7/8 m$$

let $Y_i = \begin{cases} 1 & \text{if } i\text{th clause satisfied} \\ 0 & \text{otherwise} \end{cases}$

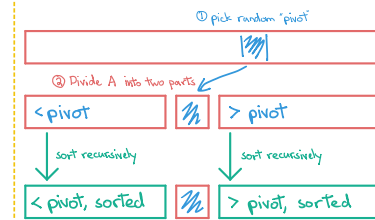
$$Z = \sum_{i=1}^m Y_i = \text{\# clauses satisfied}$$

$$E[Z] = \frac{1}{2} E[Z | x_1 = \text{true}] + \frac{1}{2} E[Z | x_1 = \text{false}]$$

$\Rightarrow$ either $E[Z | x_1 = \text{true}]$ or $E[Z | x_1 = \text{false}]$ is $\geq E[Z]$!

QuickSort,

Random divide and conquer



Expected running time of QuickSort
For each $i, j \in [n]$ w/ $i < j$,
 $X_{ij} = \begin{cases} 1 & \text{if } i\text{th smallest element is compared to } j\text{th smallest} \\ 0 & \text{otherwise} \end{cases}$
 $\sum_{i < j} X_{ij}$ = total # comparisons made by the algorithm
 $\approx$ total running time

$E[\sum_{i < j} X_{ij}]$ = average running time

$$E[\sum_{i < j} X_{ij}] = \sum_{i < j} E[X_{ij}]$$

now analyze $E[X_{ij}]$ in isolation

$$E[X_{ij}] = \Pr[X_{ij} = 1] = \Pr[i\text{th element is compared to } j\text{th}]$$

i th and j th elem. selected before any element in between happens w/ probability $\frac{2}{j-i+1}$

$$\begin{aligned} E[X_{ij}] &= \frac{2}{j-i+1} \\ E[\sum_{i < j} X_{ij}] &= \sum_{i < j} \frac{2}{j-i+1} \\ &\leq 2n \left(\frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n} \right) \\ &\leq O(n \log n) \quad \text{with harmonic series} \end{aligned}$$

Quickselect (HW)

Lecture 2

Probability

Hash Functions,

Universal Hash Functions

$h: [n] \rightarrow k$ s.t. for all pairs $i, j \in [n]$,
 $\Pr[h(i) = h(j)] \leq \frac{1}{k}$ (much weaker than ideal hash)

e.g. $h(x) = (ax + b) \bmod p \bmod k$

where • p prime, $> k$

• a in $\{1, \dots, p-1\}$ unif. random

• b in $\{0, \dots, p-1\}$ unif. random



Markov's Inequality

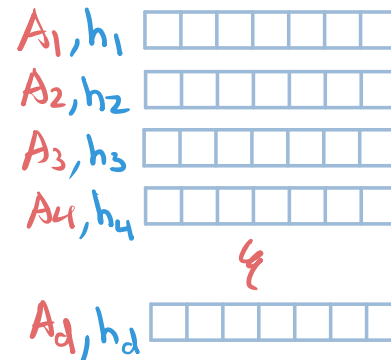
Markov's Inequality (special case)

Let $X \geq 0$ be nonnegative RV

$$\Pr[X \geq 2E[X]] \leq \frac{1}{2}$$

Algorithms

Frequency Estimation, Heavy Hitters (in streams)



estimate f_i w/
 $\min_{j=1, \dots, d} A_j[h_j(i)]$

Lecture 3

Probability

k-wise independence,
higher order moments
(generalizing universal hash
functions and Markov's inequality)

Algorithms

Hash tables w/
chaining +
linear probing

k-wise independent hash functions

$$h: [U] \rightarrow [m]$$

s.t. for any k keys $x_1, x_2, \dots, x_k \in [U]$,
 k values $y_1, y_2, \dots, y_k \in [m]$,

$$P[h(x_1)=y_1, h(x_2)=y_2, \dots, h(x_k)=y_k] = \frac{1}{m^k}$$

eg. $h(x) = a_0 + a_1x + a_2x^2 + \dots + a_{k-1}x^{k-1} \pmod{p}$

where $p \geq U$ prime,

$a_0, a_1, \dots, a_{k-1} \in [p]$ indep. uniformly random

Important lemma

let $X_1, \dots, X_n \in \{0, 1\}$ be 4-wise independent
and $\mu = E[X_1 + \dots + X_n]$. then for $\beta > 0$,

$$P[X_1 + \dots + X_n \geq \mu + \beta] \leq \frac{4 \max\{\mu, \mu^3\}}{\beta^4}$$

k-wise independent random variables

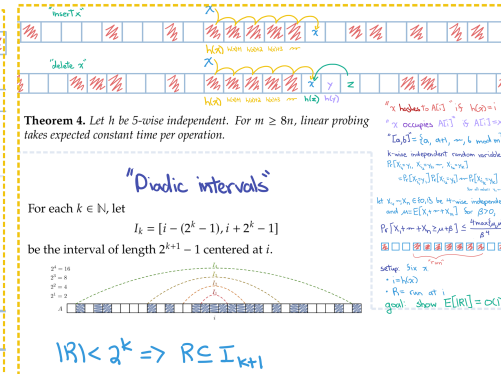
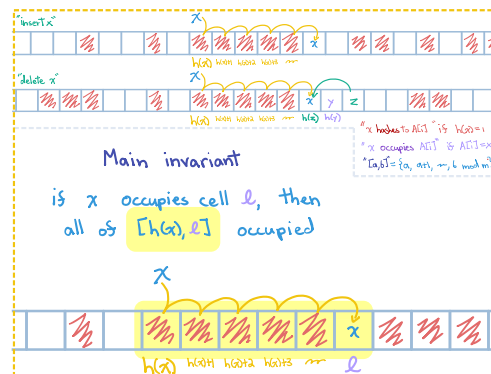
$$P[X_1=y_1, X_2=y_2, \dots, X_k=y_k]$$

$$= P[X_1=y_1] P[X_2=y_2] \dots P[X_k=y_k]$$

for all values $y_1, \dots, y_k$

Main invariant

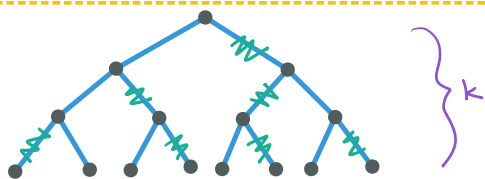
if x occupies cell i , then
all of $[h(x), i]$ occupied



Lecture 4

Probability

Galton-Watson processes



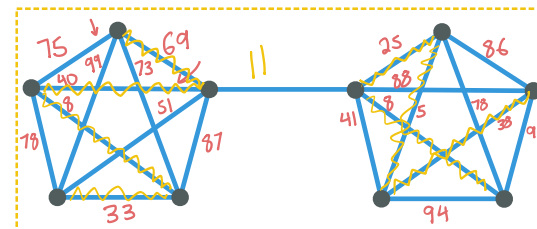
let T = complete binary tree of height k ,
w/ every edge deleted w/ prob $\leq 1/2$

Fact: $P[\text{some leaf connected to root}] \geq \frac{1}{k}$

"Galton-Watson process"

Algorithms

Random Contractions for min cut



Randomized contractions

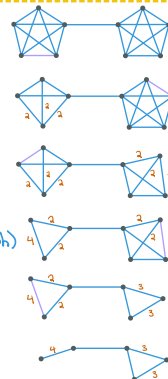
Input: $G=(V,E)$, weights $c \in \mathbb{R}_{\geq 0}^E$
(unweighted)

① while $|E| > 1$

 Ⓐ sample $e \in E$ in proportion to c

 Ⓑ "contract" e

② return set of edges (in original graph)
 contracted to remaining edge



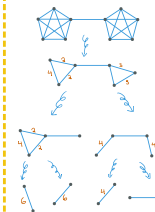
Amplification by branching

Instead of restarting at the beginning, restart partway through

High-level algorithm

① randomly contract until prob.
of ruining the min-cut is $\sim 1/2$

② Branch:
make two copies of current graph
recurse on both copies



Lecture 5

Probability

Multiplicative Chernoff Bound

Multiplicative Chernoff bound

let $X_1, \dots, X_n \in [0, 1]$ be independent

$$\mu = E[X_1 + \dots + X_n]$$

$$\epsilon \in (0, 1)$$

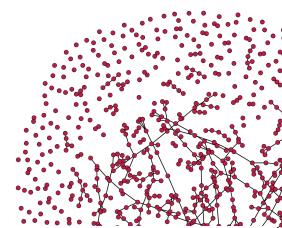
- $P[X_1 + \dots + X_n \geq (1 + \epsilon)\mu] \leq e^{-\epsilon^2 \mu / 3}$
- $P[X_1 + \dots + X_n \leq (1 - \epsilon)\mu] \leq e^{-\epsilon^2 \mu / 2}$

Algorithms (/ results)

Random graphs & the giant component

Erdős-Rényi graphs $G(n, p)$

- n vertices
- each edge sampled independently w/ probability p



Theorem 5.3. Consider a random graph $G \sim \mathcal{G}(n, p)$ for $p = c/n$, where c is a constant.

- If $c > 1$, then with high probability, there is exactly one connected component of G with $\Omega(n)$ vertices, and all other components have size $\leq O(\log n)$.
- For $c < 1$, then with high probability, all connected components of G has size $< O(\log n)$.

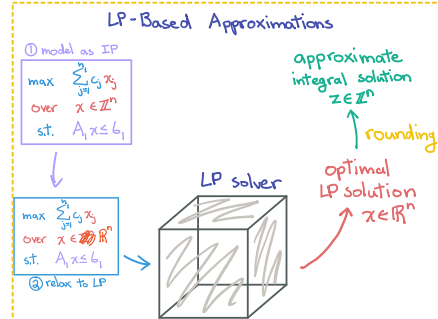
Lecture 6

Probability (& techniques)

Approximations via linear programming & randomized rounding

Linear programs (LPs)
Similar to IPs, except no integrality constraints

min/max $\sum_{j=1}^n c_j x_j$ linear objective
over $x \in \mathbb{R}^n$ linear integral variables
st. $A_1 x \leq b_1$ linear constraints
 $A_2 x = b_2$
 $A_3 x \geq b_3$



Continuous optimization vs Discrete optimization

randomization

Algorithms

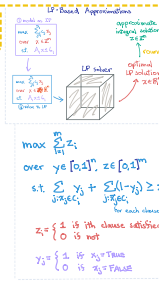
Max Sat Set Cover Covering Integer Programs

For each variable x_j , independently, randomly set $x_j = 1$ with probability p_j , and $x_j = 0$ otherwise.

$$E[\# \text{ sat. clauses}] \geq (1 - 1/e) \text{OPT}$$

k	oblivious	LP	average
1	1/2	$z_i \geq (3/4)z_i$	$\geq (3/4)z_i$
2	3/4	$(3/4)z_i \geq (3/4)z_i$	$\geq (3/4)z_i$
3	7/8	$(1 - 1/e)z_i \geq (3/4)z_i$	$\geq (3/4)z_i$
4	15/16	$(1 - 1/e)z_i \geq (3/4)z_i$	$\geq (3/4)z_i$
5	31/32	$(1 - 1/e)z_i \geq (3/4)z_i$	$\geq (3/4)z_i$
...	...	...	...
k	$1 - 1/2^k$	$(1 - 1/e)z_i \geq (3/4)z_i$	$\geq (3/4)z_i$

Can we get $\frac{3}{4}$ -APX?



Rounding LP solution
 $x \in [0, 1]^n \rightsquigarrow F \subseteq \{S_1, \dots, S_n\}$

For each set S_j , independently, take S_j with probability x_j .

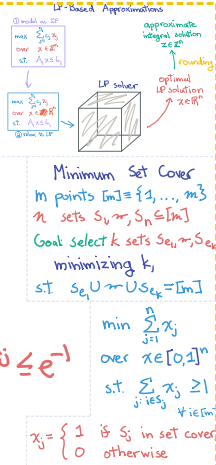
$$E[|F|] = \sum x_j \leq \text{OPT}$$

Fix a point i .

$$E[\# \text{ sets in } F \text{ covering } i] = \sum_{j \in S_i} x_j \geq z_i$$

$$P[F \text{ not cover } i] = \prod_{j \in S_i} (1 - x_j) \leq e^{-\sum_{j \in S_i} x_j} \leq e^{-z_i}$$

$$P[F \text{ cover } i] \geq 1 - e^{-1}$$



Lecture 7

Probability

Continuous Random Variables

Variance, Chebyshev's Variance Reduction

Median trick

Variance

$$\text{Var}(X) = E[(X - E[X])^2] = E[X^2] - (E[X])^2$$

Chebyshev's inequality

$$P[|X - E[X]| \geq d] \leq \frac{\text{Var}(X)}{d^2}$$

"median trick", "median of means"

- ① take mean to reduce variance, get constant prob. of error
- ② take median to amplify success prob. exponentially

Algorithms

Distinct Elements



Goal: estimate # of distinct elements in the stream

Suppose you had an ideal hash function

$$h: [m] \rightarrow [0, 1]$$

and the following facts:

let $Y_1, \dots, Y_k \in [0, 1]$ independent and uniformly random

$$X = \min \{Y_1, \dots, Y_k\}$$

$$\text{then } E[X] = \frac{1}{k+1}$$

$$\textcircled{1} X \leftarrow 1$$

② for each element e

$$\textcircled{a} X \leftarrow \min \{X, h(e)\}$$

$$/* E[X] = \frac{1}{k+1}, k = (\text{\# distinct elements}) */$$

$$\textcircled{3} \text{ return } \frac{1}{X} - 1$$

$$\text{recall } E[X] = \frac{1}{k+1}$$

$$\text{goal: show } |X - E[X]| \leq \frac{\epsilon}{k+1}$$

w/ good prob.

$$P[|X - E[X]| \geq \frac{\epsilon}{k+1}]$$

$$\leq \text{Var}(X) \cdot \left(\frac{k+1}{\epsilon}\right)^2 \leftarrow \text{Chebyshev's inequality}$$

where

$$\text{Var}(X) = \frac{k}{(k+1)^2(k+2)} \approx \frac{1}{k^2}$$

Q: how to reduce $\text{Var}(X)$?

we can get $(1 \pm \epsilon)$ -APX

w/ 10% error prob. in $O(\frac{1}{\epsilon^2})$ space

what about high probability?

① make q copies $Y_1, \dots, Y_q$ of

② return median of $Y_1, \dots, Y_q$

Algo correct $\Leftrightarrow \geq \frac{1}{2}$ of Y_i 's correct

let $Z = \# \text{correct } Y_i$'s $E[Z] = \frac{q}{2}$

$$P[Z \leq (1 - \frac{\epsilon}{4})E[Z]] \leq e^{-\left(\frac{\epsilon}{4}\right)^2 E[Z]/2} \leq \frac{1}{n^{O(n)}} \text{ for } q = O(\log n)$$

ideal hash $h: [m] \rightarrow [0, 1]$

let $Y_1, \dots, Y_k \in [0, 1]$

indep. and unifs. rand.

$$X = \min \{Y_1, \dots, Y_k\}$$

$$\text{then } E[X] = \frac{1}{k+1}$$

$$\text{and } E[X^2] = \frac{2}{(k+1)(k+2)}$$

$$\text{Var}(X) = E[(X - E[X])^2]$$

$$X_1, \dots, X_q \text{ indep. copies of } X$$

$$\bar{X} = \frac{1}{q}(X_1 + \dots + X_q)$$

$$\text{Var}(\bar{X}) = \frac{1}{q} \text{Var}(X)$$

$$P[X_1 + \dots + X_q \leq (1 - \epsilon)q] \leq e^{-\epsilon^2 q/4}$$

Lecture 8

Probability

Gaussian random variables

Gaussian random variables

$$x \sim \mathcal{N}(\mu, \sigma^2)$$

mean variance

Def: the nicest, most pleasant, convenient random variable ever

Fact let $x \sim \mathcal{N}(\mu, \sigma^2)$, $d \in \mathbb{R}$.

$$\text{then } x+d \sim \mathcal{N}(\mu+d, \sigma^2)$$

$$dx \sim \mathcal{N}(d\mu, d^2\sigma^2)$$

Fact let $x_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$, $x_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$

$$\text{then } x_1 + x_2 \sim \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$$

Fact let $x \sim \mathcal{N}(0, 1)$, $t < \frac{1}{2}$

$$\text{then } E[e^{tx^2}] = \frac{1}{\sqrt{1-2t}}$$

Algorithms

Dimensionality reduction

Bag-of-words

literally any sentence is fine

any ↑ is literally sentence English
five encodes text in high dimensions

Lemma 8.8. Let $A \sim \mathcal{N}^{k \times d}$ be a random matrix where each coordinate A_{ij} is an independently drawn sample from $\mathcal{N}$. Let $\epsilon > 0$ be sufficiently small. For any vector x ,

$$\mathbb{P}\left[(1-\epsilon)\|x\|^2 \leq \frac{1}{k}\|Ax\|^2 \leq (1+\epsilon)\|x\|^2\right] \geq 1 - 2e^{-k/8}$$

$$k \left\{ \left[\frac{1}{k} \|Ax\|^2 \right] \leftarrow \alpha_i \right.$$

$$E[\|Ax\|^2] = \sum_{i=1}^k E[(Ax)_i^2] = k\|x\|^2$$

$$E[(\sum_{i=1}^k A_{ij} x_j)^2] = \|x\|^2$$

$$A_{ij} x_j \sim \mathcal{N}(0, x_j^2)$$

$$\sum_{j=1}^n A_{ij} x_j \sim \mathcal{N}(0, \|x\|^2)$$

$$(Ax) \sim \mathcal{N}^k(0, \|x\|^2)$$

$$\Pr[\|Ax\|^2 \geq k\|x\|^2 \cdot (1+\epsilon)] \leq e^{-\epsilon^2 k/8}$$

Dimensionality Reduction - high level

$O(\log n)$ dimensions suffice

where $n = \#$ points

$$\mathbb{R}^{\text{very large}} \xrightarrow{f(x)} \mathbb{R}^{O(\log n)}$$

One can map n points into $\mathbb{R}^{O(\log n/\epsilon^2)}$ while preserving all distances up to $(1 \pm \epsilon)$ -APX

Lecture 9

Probability

Locality-sensitive hashing

Fix $\theta \in [0, \pi]$. given q , either:

(a) output $p \in P$ w/ $\angle(p, q) \leq 4\theta$

(b) declare no $p \in P$ w/ $\angle(p, q) \leq \theta$

define $h: S^{d-1} \rightarrow \{-1, 1\}^k$ by

$$h_i(x) = \text{sign}(\langle g_i, x \rangle) \text{ w/ } g_i \sim \mathcal{N}^d$$

$$k = \pi(\log(n)/2\theta)$$



d-dimensional sphere

when querying y :

- if there is $x \in P$ w/ $\angle(x, y) \leq \theta$, then it collides w/ probability $\geq 1/\sqrt{n}$

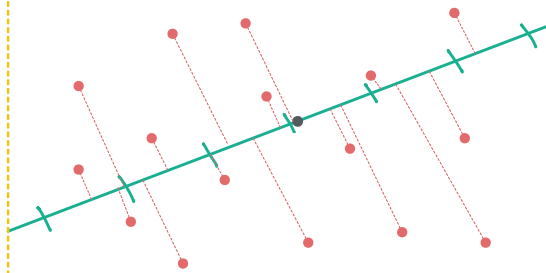
- expect to collide w/ at most one point $x \in P$ w/ $\angle(x, y) > 2\theta$

Lemma let $x, y \in S^{d-1}$
then $g \sim \mathcal{N}^d$

$$P[\text{sign}(\langle g, x \rangle) \neq \text{sign}(\langle g, y \rangle)] = \frac{\angle(x, y)}{\pi}$$

Random line embedding & Buckets

$$h(x) = \lfloor \langle g, x \rangle + d \rfloor \text{ w/ } g \sim \mathcal{N}^d, d \in [0, 1] \text{ unifs. rand.}$$

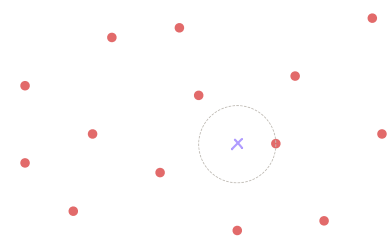


Algorithms

Approximate nearest neighbors

Nearest neighbor search

Input: n points $P \subset \mathbb{R}^d$



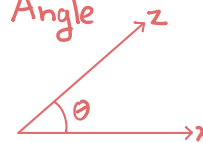
Query: given $q \in \mathbb{R}^d$, find closest point in P to q

Types of distances

- Euclidean distance

$$\|x - z\|_2 = \sqrt{\sum_i (x_i - z_i)^2}$$

- Angle



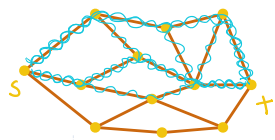
Lecture 10

Probability (and techniques)

LP duality

L_1 -metric embeddings

Max Flow Min Cut



Path Packing LP

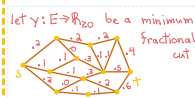
maximize $\sum_{p \in P} x_p$
over $x: P \rightarrow \mathbb{R}_{\geq 0}$
(all s-t paths)

s.t. $\sum_{p \in P} x_p \leq c_e$ for all edges e
"Ax ≤ c" (capacity of e)

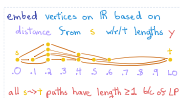
Path Covering LP (i.e. cut)

minimize $\sum_{e \in E} c_e y_e$
over $y: E \rightarrow \mathbb{R}_{\geq 0}$
(total capacity of "fractional cut")

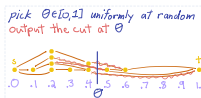
s.t. $\sum_{e \in P} y_e \geq 1$ for all $p \in P$
"A'y ≥ b" (continuous analog of s-t cut)



let $y: E \rightarrow \mathbb{R}_{\geq 0}$ be a minimum fractional cut



embed vertices on $\mathbb{R}$ based on distance from s with lengths y



pick $\theta \in [0, 1]$ uniformly at random output the cut at θ

$E[\text{capacity of cut}]$

$$\stackrel{\text{(linearity of expectation)}}{=} \sum_{e \in E} c_e \Pr[e \text{ cut}] \leq \sum_{e \in E} c_e y_e$$

minimum capacity of any fractional cut

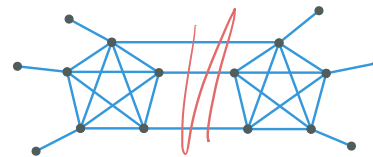


Algorithms

Sparsest Cut

Special case: uniform sparsest cut
w/ uniform demands $b(u,v) = 1 \quad \forall u,v$

$$\text{"sparsity"} = \frac{c(\delta(s))}{|S| |T|}$$



useful primitive for graph partitioning, divide & conquer

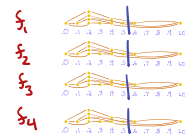
Q: how to round LP to sparse cut?

min $\sum_{e \in E} c_e d(u,v)$ over metrics $d: V \times V \rightarrow \mathbb{R}_{\geq 0}$
s.t. $\sum_{i=1}^k b_i d(s_i, t_i) \geq 1$

Suppose $d(u,v)$ was of the form

$$d(u,v) = \sum_{i=1}^k |f_i(u) - f_i(v)| \quad \text{for } f_i: V \rightarrow [0, 1]^k$$

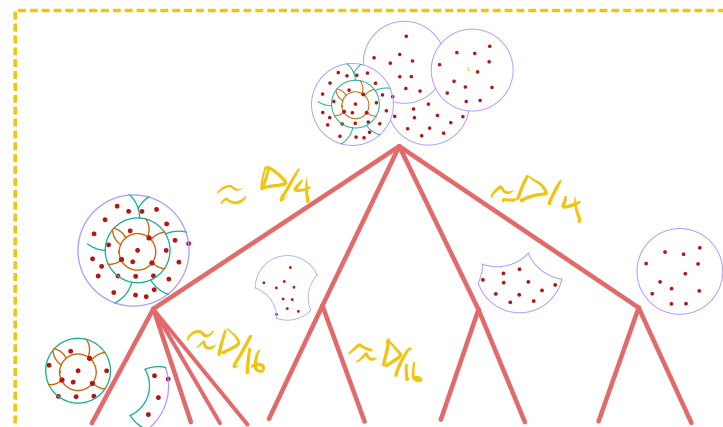
(assume $\min f_i(a) = 0, \max f_i(a) = 1$)



" L_1 -metric"

Sparsest cut
Input: undirected graph $G=(V,E)$, edge capacities $c(e) > 0$ for $e \in E$, demands $b(u,v) > 0$ for $u,v \in V$.
Goal: compute SCF minimizing "sparsity" $\frac{c(\delta(s))}{|S| |T|}$ where S and T are edge cuts.

Algorithms



Lecture 12

Probability (and techniques)

Additive Chernoff Inequality

Double-sampling

VC dimension

Additive Chernoff bound

let $X_1, \dots, X_n \in [0, 1]$ be independent

$$\mu = E[X_1 + \dots + X_n] / n$$

$\mu \in (0, 1)$

- $P[\frac{X_1 + \dots + X_n}{n} \geq \mu + \epsilon] \leq e^{-2\epsilon^2 n}$
- $P[\frac{X_1 + \dots + X_n}{n} \leq \mu - \epsilon] \leq e^{-2\epsilon^2 n}$

Algorithms

ϵ -samples and ϵ -nets

ϵ -sample $|M_Q(r) - \mu(r)| \leq \epsilon \quad \forall r \in R$

let Q sample k points w/ repetition
 $k = O\left(\frac{\ln(g(|P|))}{\epsilon^2}\right) \Rightarrow \epsilon$ -sample $k = O\left(\frac{\ln(g(|P|))}{\epsilon^2}\right) \Rightarrow \epsilon$ -net (w/ constant probability)

ϵ -sample theorem [VC71]
 $k \geq O\left(\frac{\ln(g(2k))}{\epsilon^2}\right) \Rightarrow \epsilon$ -sample (w/ constant probability)

e.g. $g(n) = O(n^d)$
 $k \geq O\left(\frac{d \ln(d/\epsilon)}{\epsilon^2}\right) \Rightarrow \epsilon$ -sample
 ↑ independent of $|P|$ (!)

$P[\frac{X_1 + \dots + X_n}{n} \geq \mu + \epsilon], P[\frac{X_1 + \dots + X_n}{n} \leq \mu - \epsilon] \leq e^{-2\epsilon^2 n}$ $g(n) = \max\{|RAQ| : \text{asf, lalsn}\}$

$\mu(r) = \frac{|P \cap r|}{|P|}$
 $M_Q(r) = \frac{|Q \cap r|}{|Q|}$
 $RAQ = \{r \cap q : r \in R, q \in Q\}$

ϵ -sample $|M_Q(r) - \mu(r)| \leq \epsilon \quad \forall r \in R$

let Q sample k points w/ repetition
 $k = O\left(\frac{\ln(g(|P|))}{\epsilon^2}\right) \Rightarrow \epsilon$ -sample $k = O\left(\frac{\ln(g(|P|))}{\epsilon^2}\right) \Rightarrow \epsilon$ -net (w/ constant probability)

ϵ -net theorem [HW87]
 $k \geq O\left(\frac{\ln(g(2k))}{\epsilon^2}\right) \Rightarrow \epsilon$ -net (w/ constant probability)

e.g. $g(n) = O(n^d)$
 $k \geq O\left(\frac{d \ln(d/\epsilon)}{\epsilon^2}\right) \Rightarrow \epsilon$ -net
 ↑ independent of $|P|$ (!)

$P[\frac{X_1 + \dots + X_n}{n} \geq \mu + \epsilon], P[\frac{X_1 + \dots + X_n}{n} \leq \mu - \epsilon] \leq e^{-2\epsilon^2 n}$ $g(n) = \max\{|RAQ| : \text{asf, lalsn}\}$

$\mu(r) = \frac{|P \cap r|}{|P|}$
 $M_Q(r) = \frac{|Q \cap r|}{|Q|}$
 $RAQ = \{r \cap q : r \in R, q \in Q\}$

Lecture 13

Probability
(and techniques)

Algorithms
(models and applications)

Consistency model

PAC learning model

Generalization error

Occam's Razor

Generalization by growth

Lecture 14

Probability
(and techniques)

Algorithms
(models and applications)

Entropy

(Optimal)
Error-Correcting
Codes

Lecture 15

Probability (and techniques)

Lovász Local Lemma

Algorithms (models and applications)

Moser's resampling
algorithm

Lecture 16

Probability (and techniques)

Algorithms (models and applications)

Random Walks

Stationary Distribution

Perron-Frobenius Theorem

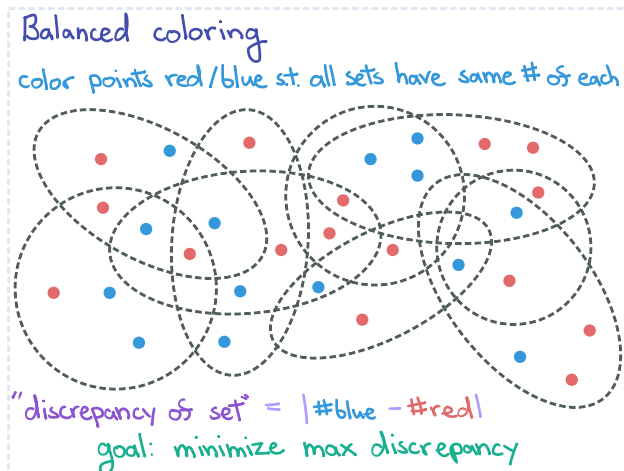
Pagerank

Lecture 17

Probability (and techniques)

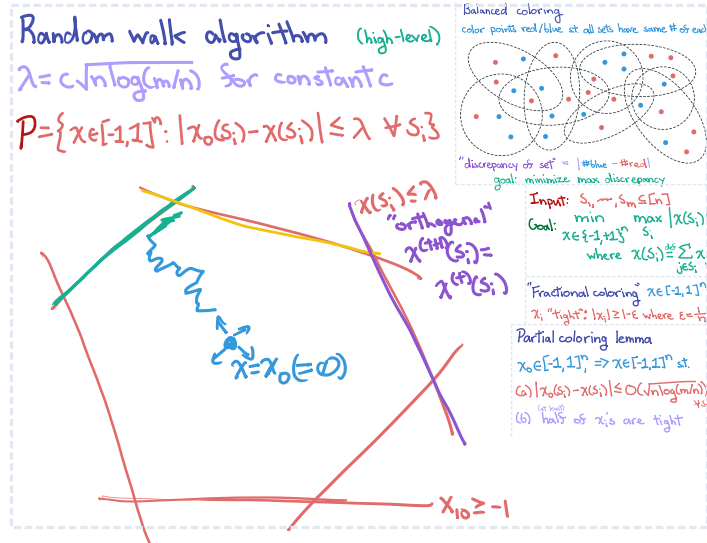
High-dimensional random walks

More fun w/ Gaussians



Algorithms (models and applications)

Discrepancy minimization



Lecture 18

Probability (and techniques)

Algorithms (models and applications)

Hitting times

Cover times

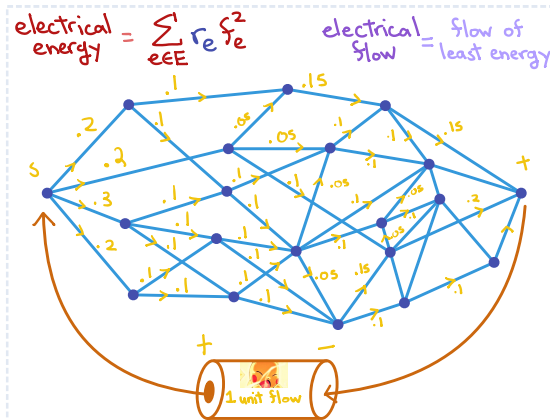
Electrical networks

Connectivity
in $O(\log n)$ space

random walk terminology:

Hitting time $H(s, t)$ = expected # steps until a random walk from s reaches t

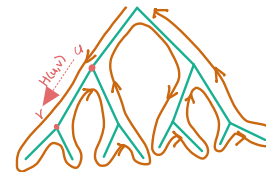
Cover time $C(s)$ = expected # steps until a random walk from s visits every vertex



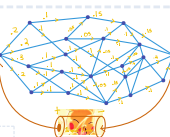
Lemma $H(u, v) + H(v, u) = 2m \cdot (\text{energy of unit flow from } u \text{ to } v)$

Theorem $C(s) \leq 2mn$

proof Six traversal of spanning tree



$$\begin{aligned} \text{total time} &= \sum_{u, v \in V} H(u, v) + H(v, u) \\ &= 2m \sum_{u, v \in V} (\text{energy of unit flow from } u \text{ to } v) \leq 2m(n-1) \end{aligned}$$



electrical energy = $\sum_{e \in E} r_e f_e^2$
electrical flow = least energy

des $B: \mathbb{R}^E \rightarrow \mathbb{R}^V$ by
(BS) $_v$ = net flow at v
 $d_v = \begin{cases} -1 & \text{if } v \text{ is } s \\ 1 & \text{if } v \text{ is } t \\ 0 & \text{otherwise} \end{cases}$

min $\frac{1}{2} \sum_{e \in E} r_e f_e^2$ s.t. $BS = d$
(electrical energy) (is a typical)

effective resistance of d = optimum value/energy
Ohm's law: $f = R^{-1} B^T p$
for electric flow by electric potentials p
 $L = BR^{-1}B^T \in \mathbb{R}^{V \times V}$
(effective resist. of d) $= \langle d, L^{-1}d \rangle$

Lecture 19

Probability (and techniques)

Algorithms (models and applications)

Spectral theorem
for symmetric maps

Spectral analysis of
random walks

Lecture 20

Probability
(and techniques)

Algorithms
(models and applications)

Conductance

Cheeger's inequality

Conductance-based
mixing time

Fiedler's algorithm
for low-conductance
cuts

Lecture 21

Probability
(and techniques)

Algorithms
(models and applications)

Expanders

Zigzag product

tensors

Deterministic

Connectivity in

$O(\log n)$ space

Lecture 22

Probability
(and techniques)

classes RP and BPP

tensors of graphs

Algorithms
(models and applications)

More efficient
amplification

Efficient construction
of exponentially
large expanders

Lecture 23/24

Probability
(and techniques)

Algorithms
(models and applications)

Probabilistically
checkable proofs

Lecture 25

Probability (and techniques)

Discrete Fourier analysis

Algorithms (models and applications)

Property testing:

- Linearity test
- Dictatorship test
- Universal tester

Lecture 26

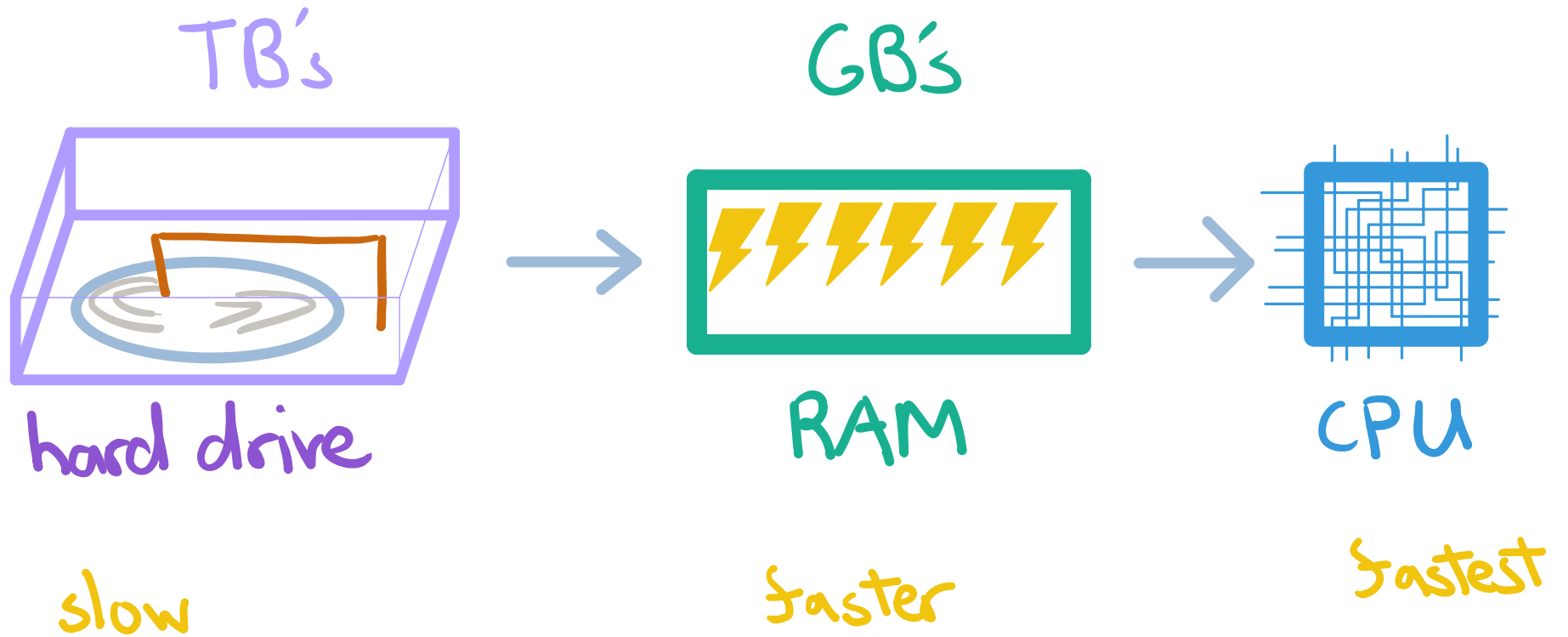
Probability
(and techniques)

Algorithms
(models and applications)

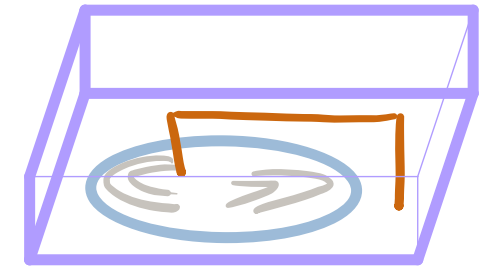
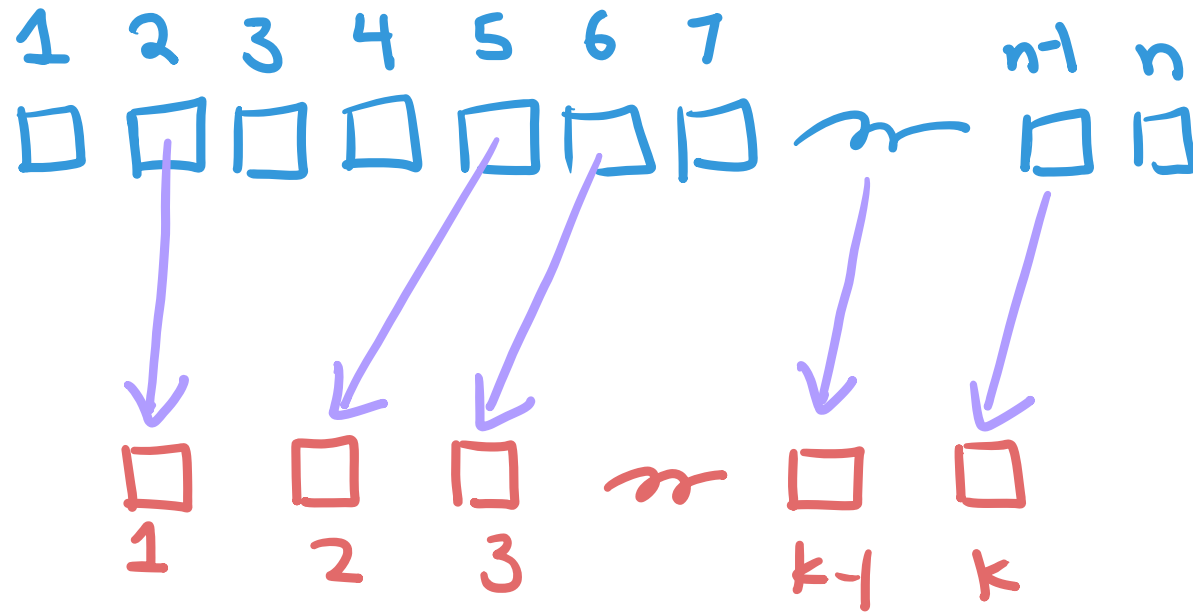
Caching

Buy-or-rent

Caching



Model: n items, k cache slots, $n \gg k$



hard drive



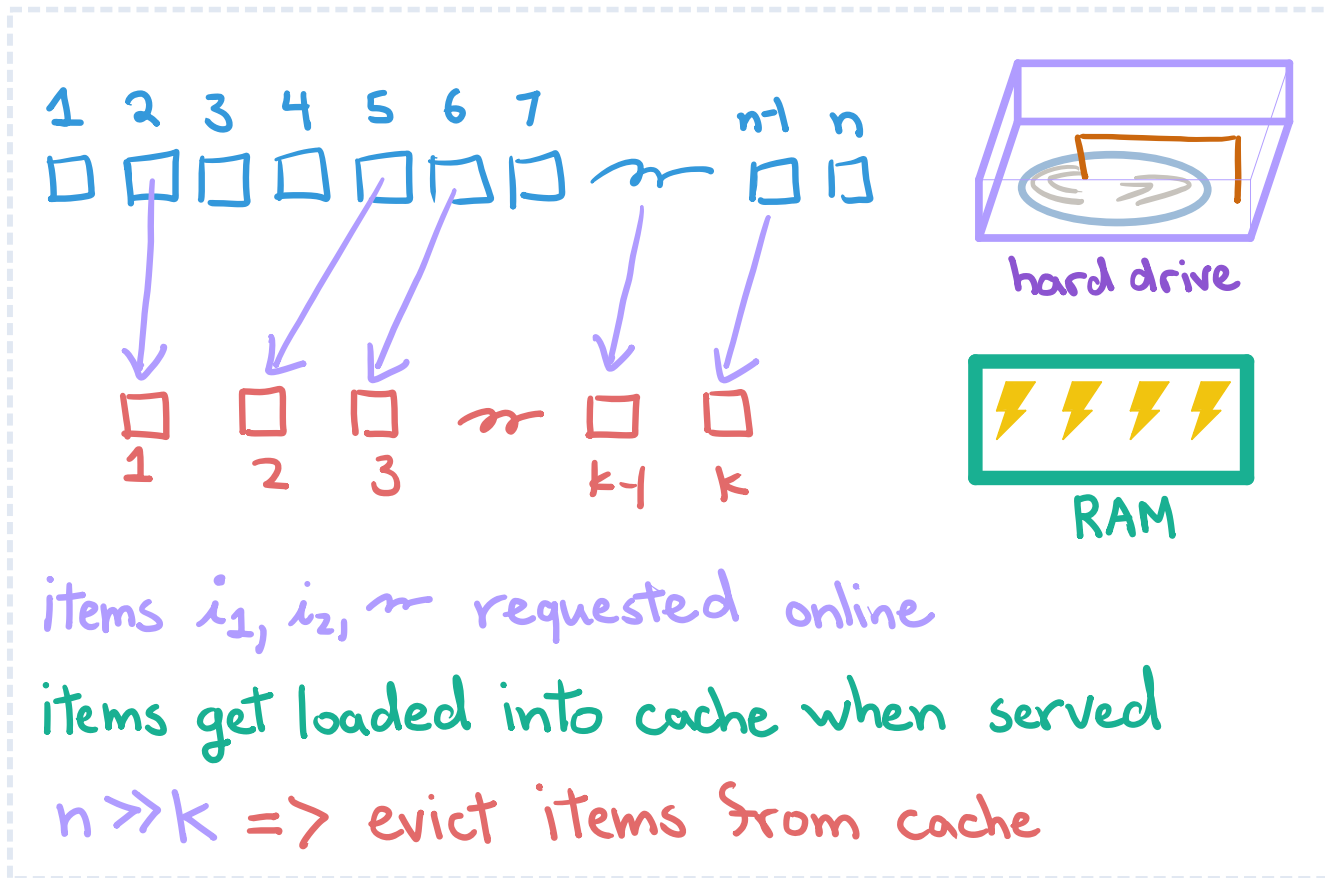
RAM

items $i_1, i_2, \dots$ requested online

items get loaded into cache when served

$n \gg k \Rightarrow$ evict items from cache

cache strategies: who to evict?



good: item already in cache when requested

bad: have to load item into cache when requested, "cache miss"

Requests

1

2

1

4

3

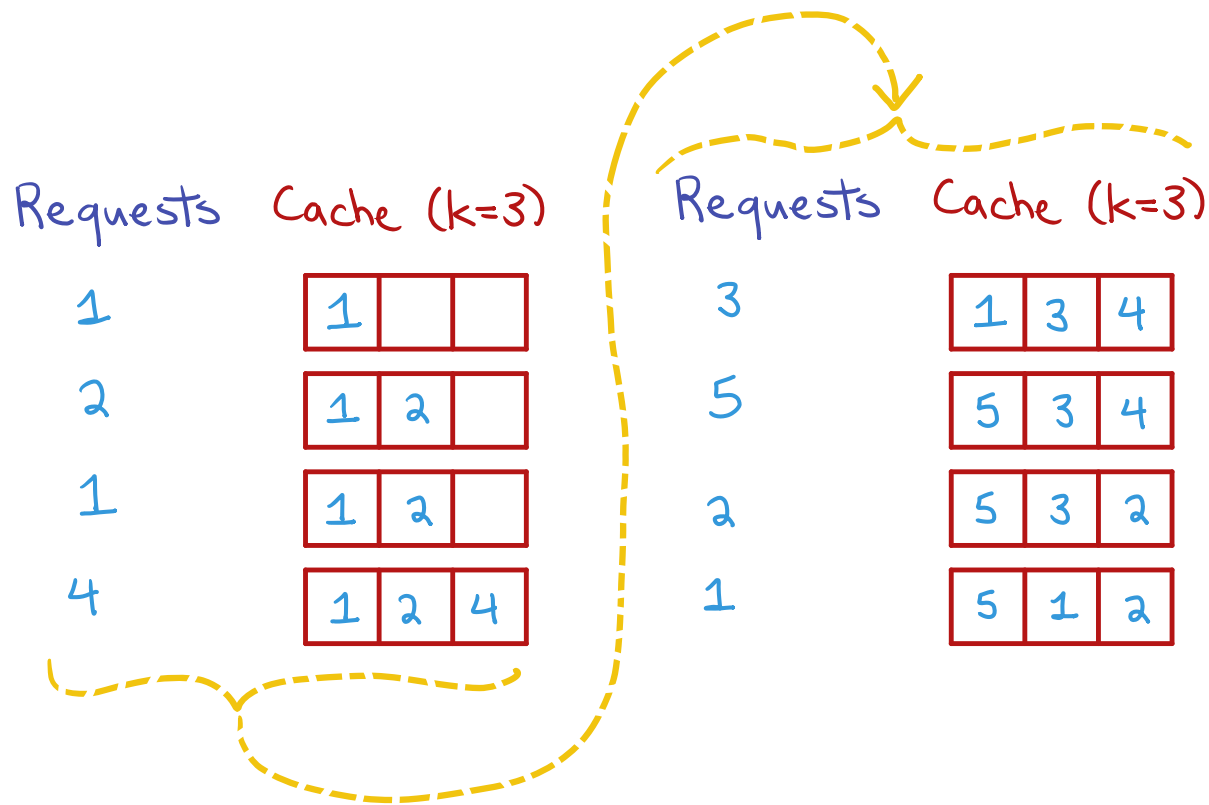
5

2

1

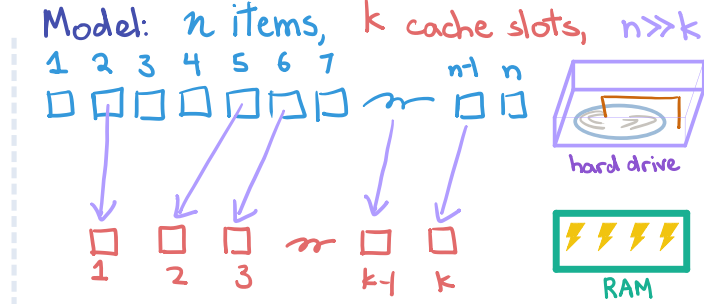
Cache ($k=3$)





Can we come up w/ smart caching strategies and improve performance?

Caching Strategies



items $i_1, i_2, \dots$ requested online

items get loaded into cache when served

$n \gg k \Rightarrow$ also evict items from cache

Goal: minimize cache misses

Least frequently used (LFU)

Least recently used (LRU)

Not recently used (NRU, aka 1-Bit LRU)

"mark" requested items

evict unmarked items

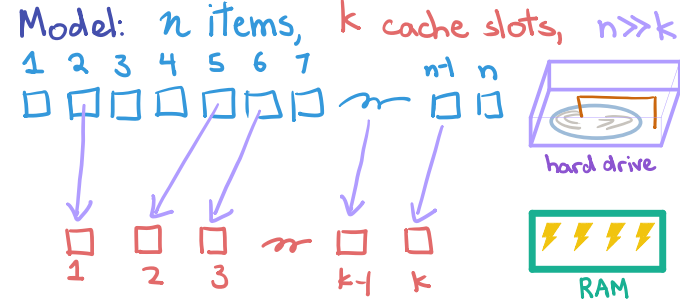
if all cache items marked, then
remove all marks and try again

Worst Case Analysis

count # cache misses
try to minimize

adversary can always request
new item and force cache miss

cache misses unbounded ;)



items $i_1, i_2, \dots$ requested online

items get loaded into cache when served

$n \gg k \Rightarrow$ also evict items from cache

Goal: minimize cache misses

Caching Strategies

Least frequently used (LFU)

Least recently used (LRU)

Not recently used (NRU, aka 1-Bit LRU)

"mark" requested items

evict unmarked items

if all cache items marked, then
remove all marks and try again

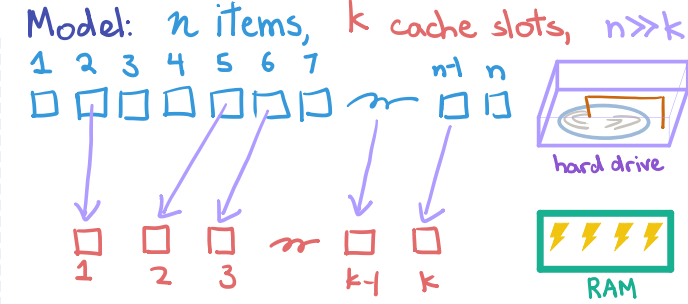
~~Worst Case Analysis~~

Reframe analysis (competitive)

if there is a good cache strategy, then we want to find a comparably good one

"competitive analysis"

minimize ratio $\frac{\text{User misses}}{\text{OPT misses}}$
(in hindsight)



items $i_1, i_2, \dots$ requested online

items get loaded into cache when served

$n \gg k \Rightarrow$ also evict items from cache

Goal: minimize cache misses

Caching Strategies

Least frequently used (LFU)

Least recently used (LRU)

Not recently used (NRU, aka 1-Bit LRU)

"mark" requested items

evict unmarked items

if all cache items marked, then remove all marks and try again

Competitive analysis of LFU

Request		LFU	Optimal
q rounds	1	x	x
	1	1	1
	ℓ	ℓ	ℓ
	1	1	1
		1 x	1 x
	2	1 2 x	1 2 x
	3	1 2 x	1 2 x

29
rounds

2

1	3	x
---	---	---

2	3
---	---

3

1	2	x
---	---	---

2	3
---	---

ℓ

2

1	3	x
---	---	---

2	3
---	---

3

1	2	x
---	---	---

2	3
---	---

cont'd

	Request	LFU	Optimal
q rounds	1	x	x
	1	1	1
	⌀	⌀	⌀
	1	1	1

	Request	LFU	Optimal
2q rounds	2	1 x	1 x
	3	1 2 x	1 2 x
	2	1 3 x	2 3
	3	1 2 x	2 3
	⌀	⌀	⌀
	2	1 3 x	2 3
	3	1 2 x	2 3

Competitive analysis of LFU ^(least frequently used)

3 items {1, 2, 3}

cache size $k=2$

To fool LFU:

load up on first item for q rounds

alternate other items for $2q$ rounds

OPT has 3 cache misses

LFU has $2q+1$ misses

$$\text{Competitive ratio} = \frac{2q+1}{3} \xrightarrow{q \rightarrow \infty} +\infty$$

	Request	LFU	Optimal
q rounds	1		×
	1	1	1
	⋮	⋮	⋮
	1	1	1
$2q$ rounds	2	1 ×	1 ×
	3	1 2 ×	1 2 ×
	2	1 3 ×	2 3
	3	1 2 ×	2 3
	⋮	⋮	⋮
	2	1 3 ×	2 3
	3	1 2 ×	2 3
	⋮	⋮	⋮

Theorem

LFU has unbounded competitive ratio

	Request	LFU		Optimal
q rounds	1		x	x
	1	1		1
	$\varnothing$	$\varnothing$		$\varnothing$
	1	1		1
2q rounds	2	1	x	1 x
	3	1 2	x	1 2 x
	2	1 3	x	2 3
	3	1 2	x	2 3
	$\varnothing$	$\varnothing$		$\varnothing$
	2	1 3	x	2 3
	3	1 2	x	2 3

Competitive Analysis of NRU^(not recently used)

Recall

NRU (1-Bit LRU)

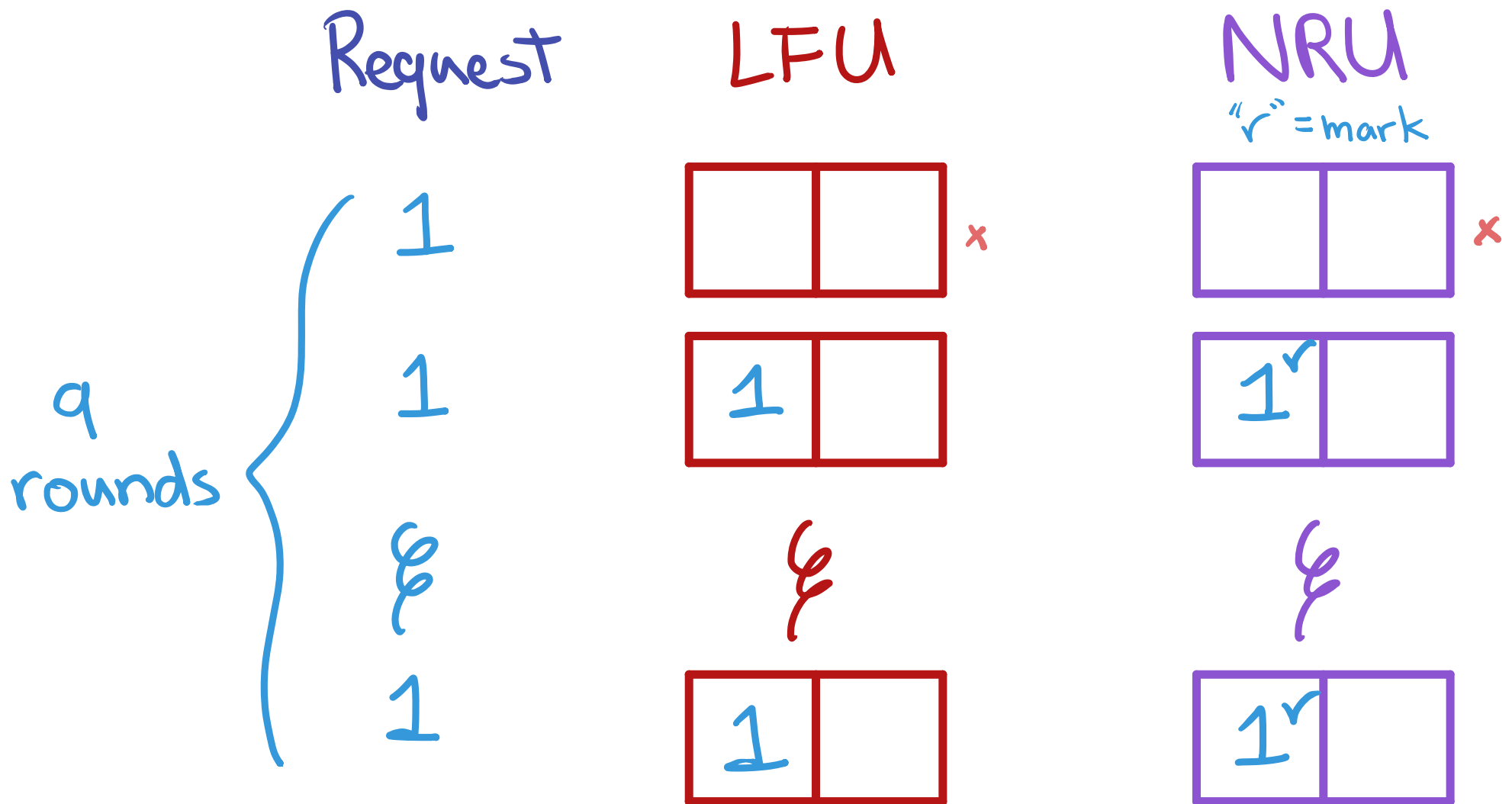
"mark" requested items

evict unmarked items

if all cache items marked, then
remove all marks and try again

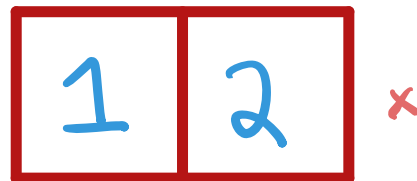
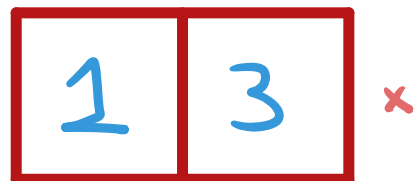
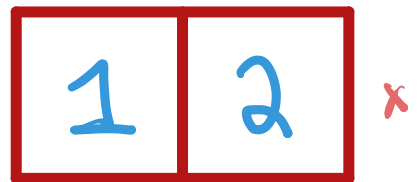
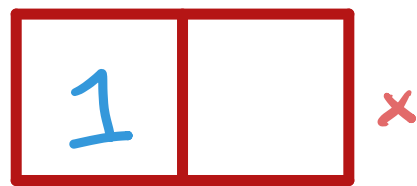
Claim: NRU has competitive ratio $\leq k$

$$\frac{\text{\#NRU misses}}{\text{\#OPT misses}} \leq k$$

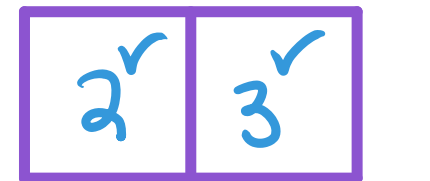
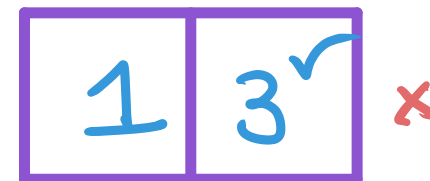
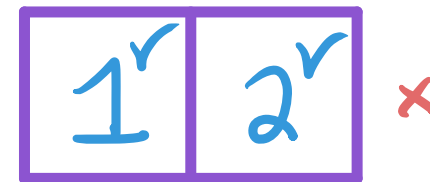
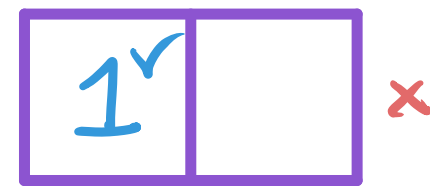
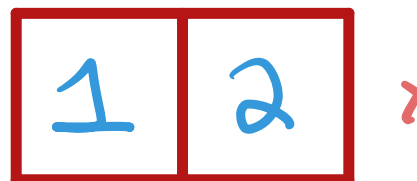
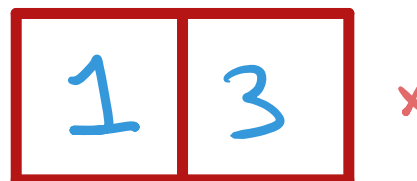


$2q$
 rounds

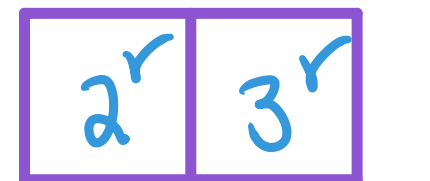
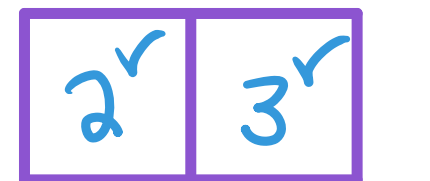
{
 2
 3
 2
 3
 2
 3



⌋



⌋



Claim: $\frac{\# \text{NRU misses}}{\# \text{OPT misses}} \leq k$

NRU (1-Bit LRU)

"mark" requested items

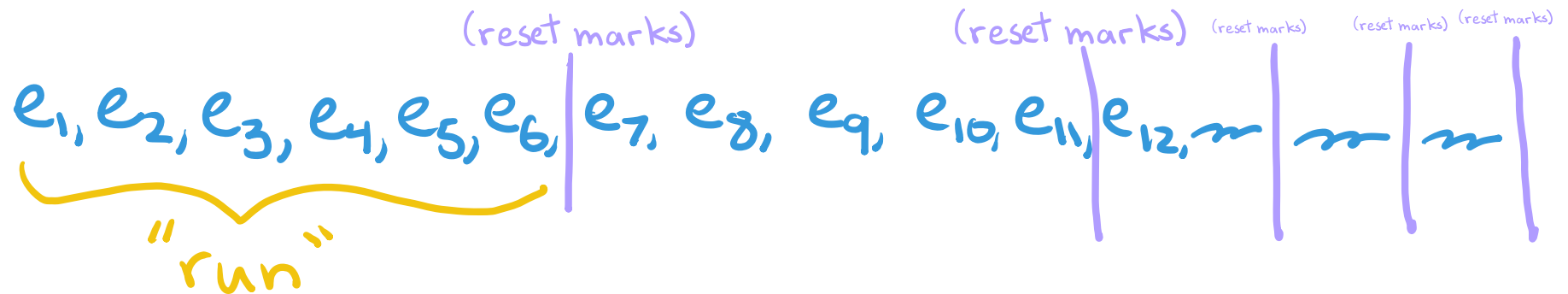
evict unmarked items

if all cache items marked, then
remove all marks and try again

Proof: fix sequence of requests

$e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{13}, e_{14}, \dots$

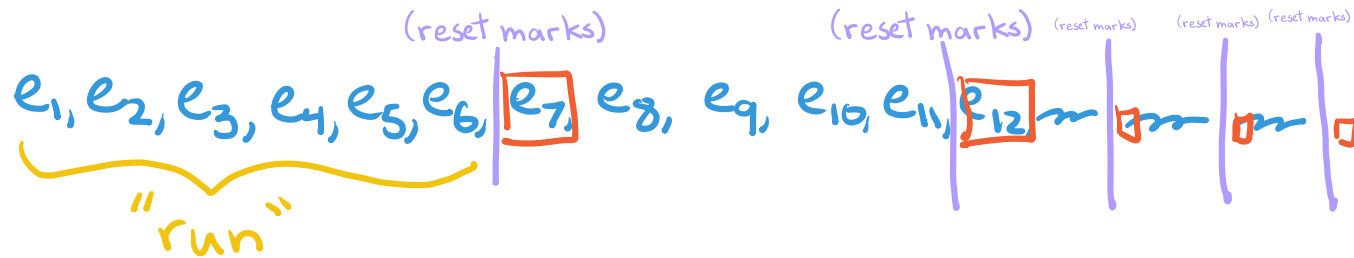
split into "runs" whenever we reset marks



each run (between resets) has k distinct items

NRU has $\leq k$ misses per run

each **run** (between resets) has k distinct items



First item after **run** is distinct from the k distinct items in the run

Any cache must miss either:

- One of the last $k-1$ items in run
- 1st item in the next run

$\Rightarrow$ OPT has ≥ 1 miss per run

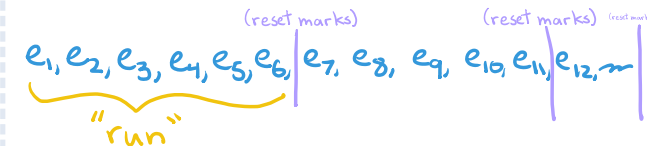
NRU (1-Bit LRU)

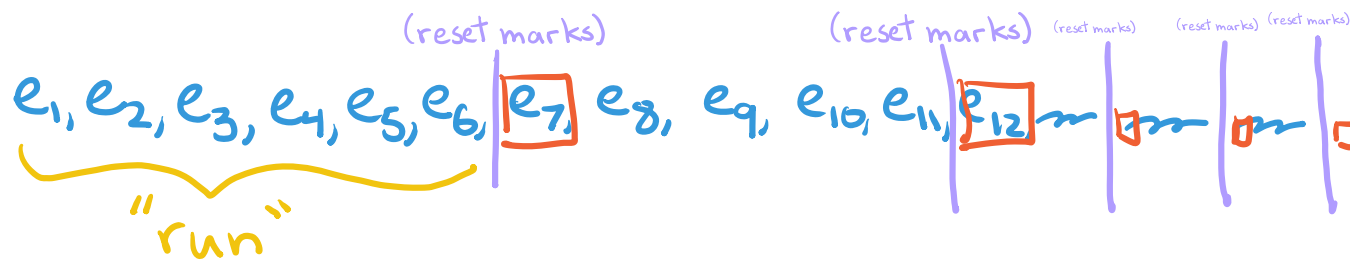
"mark" requested items

evict unmarked items

if all cache items marked, then remove all marks and try again

split into "runs" whenever we reset marks





NRU (1-Bit LRU)

"mark" requested items

evict unmarked items

if all cache items marked, then
remove all marks and try again

each run (between resets) has k distinct items

NRU has $\leq k$ misses per run

first item after run is distinct from
the k distinct items in the run

Any cache must miss either:

- one of the last $k-1$ items in run
- 1st item in the next run

$\Rightarrow$ OPT has ≥ 1 miss per run

$$\frac{\# \text{ NRU misses}}{\# \text{ OPT misses}} \leq \frac{k \cdot (\# \text{ runs})}{1 \cdot (\# \text{ runs})} = k$$

Competitive Analysis of NRU^(not recently used)

Recall

NRU (1-Bit LRU)

"mark" requested items

evict unmarked items

if all cache items marked, then
remove all marks and try again

Theorem: NRU^{*} has competitive ratio $\leq k$

$$\frac{\text{\#NRU misses}}{\text{\#OPT misses}} \leq k$$

^{*} LRU is a special case of NRU

better?

Randomized NRU (1-Bit LRU)

"mark" requested items

evict uniformly random unmarked items

if all cache items marked, then
remove all marks and try again

claim: randomized NRU is

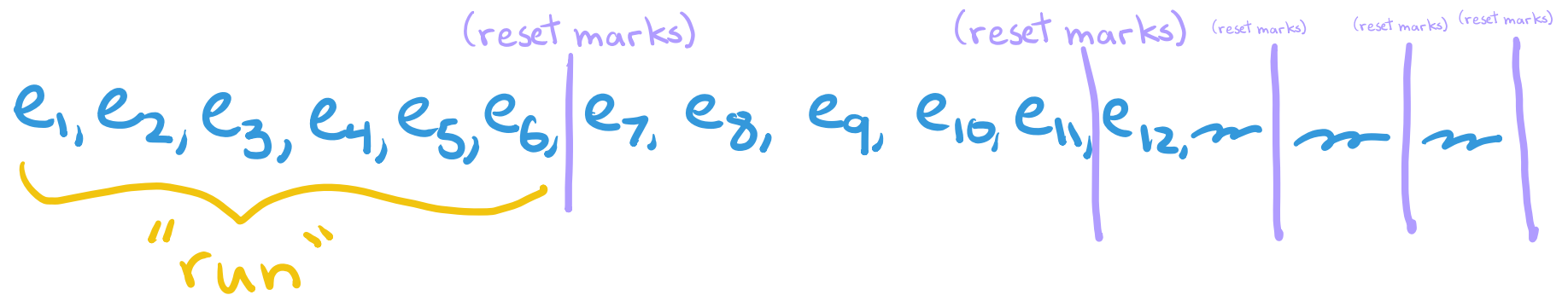
$O(\log k)$ -competitive in expectation

claim: randomized NRU is
 $O(\log k)$ -competitive in expectation

Randomized NRU (1-Bit LRU)
"mark" requested items
evict uniformly random unmarked items
if all cache items marked, then
remove all marks and try again

proof.

split into "runs" whenever we reset marks



each run (between resets) has k distinct items, divided into:

- "new" items that are not in previous run
- "repeat" items also in previous run

potential

$\Phi = \# \text{ items both } \begin{cases} \text{in OPT's cache} \\ \text{not in NRU cache} \end{cases}$

consider a single run $\overset{\text{(reset marks)}}{e_7, e_8, e_9, e_{10}, e_{11}} \mid \overset{\text{(reset marks)}}{e_{12}, \dots}$

let $\Phi_{\text{in}} = \Phi$ before run, $\Phi_{\text{out}} = \Phi$ after run

$\ell = \# \text{ new items in run}$ ($k - \ell$ repeats)

claim: OPT misses $\geq \ell - \Phi_{\text{in}}$

why?

Randomized NRU (1-Bit LRU)

"mark" requested items

evict uniformly random unmarked items

if all cache items marked, then
remove all marks and try again

claim: randomized NRU is

$O(\log k)$ -competitive in expectation

split into "runs" whenever we reset marks

$\overset{\text{(reset marks)}}{e_1, e_2, e_3, e_4, e_5, e_6} \mid \overset{\text{(reset marks)}}{e_7, e_8, e_9, e_{10}, e_{11}} \mid \overset{\text{(reset marks)}}{e_{12}, \dots}$
"run"

each run (between resets) has k distinct items:

- "new" items not in previous run
- "repeat" items also in previous run

$\Phi = \# \text{ items both } \begin{cases} \text{in OPT's cache} \\ \text{not in NRU cache} \end{cases}$

Randomized NRU (1-Bit LRU)

"mark" requested items
evict uniformly random unmarked items
if all cache items marked, then
remove all marks and try again

consider a single run $\begin{matrix} \text{(reset marks)} & & \text{(reset marks)} \\ |e_7, e_8, e_9, e_{10}, e_{11}| \end{matrix}$

let $\Phi_{\text{in}} = \Phi$ before run, $\Phi_{\text{out}} = \Phi$ after run
 $\ell = \# \text{ new items in run}$ ($k - \ell$ repeats)

✓ OPT misses $\geq \ell - \Phi_{\text{IN}}$

claim: OPT misses $\geq \Phi_{\text{OUT}}$

why?

claim: randomized NRU is
 $O(\log k)$ -competitive in expectation

split into "runs" whenever we reset marks

$\begin{matrix} \text{(reset marks)} & & \text{(reset marks)} & \text{(reset marks)} \\ e_1, e_2, e_3, e_4, e_5, e_6, & | & e_7, e_8, e_9, e_{10}, e_{11}, & | & e_{12}, \dots \end{matrix}$
"run"

each run (between resets) has k distinct items:

- "new" items not in previous run
- "repeat" items also in previous run

$\Phi = \# \text{ items both } \begin{cases} \text{in OPT's cache} \\ \text{not in NRU cache} \end{cases}$

Randomized NRU (1-Bit LRU)

"mark" requested items
evict uniformly random unmarked items
if all cache items marked, then
remove all marks and try again

consider a single run $\begin{matrix} \text{(reset marks)} & & \text{(reset marks)} \\ |e_7, e_8, e_9, e_{10}, e_{11}| \end{matrix}$

let $\Phi_{in} = \Phi$ before run, $\Phi_{out} = \Phi$ after run
 $\ell = \# \text{ new items in run } (k-\ell \text{ repeats})$

✓ OPT misses $\geq \ell - \Phi_{in}$

✓ OPT misses $\geq \Phi_{out}$

$$\Rightarrow \text{OPT misses} \geq \frac{\ell + \Phi_{out} - \Phi_{in}}{2}$$

claim: randomized NRU is
 $O(\log k)$ -competitive in expectation

split into "runs" whenever we reset marks

$\begin{matrix} \text{(reset marks)} & & \text{(reset marks)} & \text{(reset marks)} \\ e_1, e_2, e_3, e_4, e_5, e_6, & | & e_7, e_8, e_9, e_{10}, e_{11}, & | & e_{12}, \dots \end{matrix}$
"run"

each run (between resets) has k distinct items:

- "new" items not in previous run
- "repeat" items also in previous run

$\Phi = \# \text{ items both } \begin{cases} \text{in OPT's cache} \\ \text{not in NRU cache} \end{cases}$

Randomized NRU (1-Bit LRU)

"mark" requested items
evict uniformly random unmarked items
if all cache items marked, then
remove all marks and try again

consider a single run $\begin{matrix} \text{(reset marks)} & & \text{(reset marks)} \\ |e_7, e_8, e_9, e_{10}, e_{11}| \end{matrix}$

claim: randomized NRU is
 $O(\log k)$ -competitive in expectation

let $\Phi_{in} = \Phi$ before run, $\Phi_{out} = \Phi$ after run
 $\ell = \# \text{ new items in run } (k - \ell \text{ repeats})$

split into "runs" whenever we reset marks

$\begin{matrix} \text{(reset marks)} & & \text{(reset marks)} & \text{(reset marks)} \\ e_1, e_2, e_3, e_4, e_5, e_6, | e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, \dots \end{matrix}$
"run"

each run (between resets) has k distinct items:

- "new" items not in previous run
- "repeat" items also in previous run

NRU: miss all ℓ new items

$P[\text{NRU misses 1st repeat}] \leq$



$P[\text{NRU misses 2nd repeat}] \leq$



$\vdots$

$P[\text{NRU misses } i\text{th repeat}] \leq$



$$\text{OPT misses} \geq \frac{\ell + \Phi_{out} - \Phi_{in}}{2}$$

$\Phi = \# \text{ items both } \begin{cases} \text{in OPT's cache} \\ \text{not in NRU cache} \end{cases}$

Randomized NRU (1-Bit LRU)

"mark" requested items
evict uniformly random unmarked items
if all cache items marked, then
remove all marks and try again

consider a single run $\begin{matrix} \text{(reset marks)} & & \text{(reset marks)} \\ |e_7, e_8, e_9, e_{10}, e_{11}| \end{matrix}$

let $\Phi_{in} = \Phi$ before run, $\Phi_{out} = \Phi$ after run
 $\ell = \# \text{ new items in run } (k-\ell \text{ repeats})$

claim: randomized NRU is
 $O(\log k)$ -competitive in expectation

split into "runs" whenever we reset marks

$\begin{matrix} \text{(reset marks)} & & \text{(reset marks)} & \text{(reset marks)} \\ e_1, e_2, e_3, e_4, e_5, e_6, | e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, \dots \end{matrix}$
"run" (between resets)

each run (between resets) has k distinct items:

- "new" items not in previous run
- "repeat" items also in previous run

NRU: miss all ℓ new items

$$\checkmark P[\text{NRU misses } i\text{th repeat}] \leq \frac{\ell + i - 1}{k}$$

$$\text{OPT misses} \geq \frac{\ell + \Phi_{out} - \Phi_{in}}{2}$$

$$E[\# \text{ misses on repeats}] \leq$$

$\Phi = \# \text{ items both } \begin{cases} \text{in OPT's cache} \\ \text{not in NRU cache} \end{cases}$

Randomized NRU (1-Bit LRU)

"mark" requested items
evict uniformly random unmarked items
if all cache items marked, then
remove all marks and try again

claim: randomized NRU is

$O(\log k)$ -competitive in expectation

split into "runs" whenever we reset marks

$e_1, e_2, e_3, e_4, e_5, e_6 \mid e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, \dots$
(reset marks) (reset marks) (reset marks)
 "run"

each run (between resets) has k distinct items:

- "new" items not in previous run
- "repeat" items also in previous run

consider a single run $\mid e_7, e_8, e_9, e_{10}, e_{11} \mid$
(reset marks) (reset marks)

let $\Phi_{in} = \Phi$ before run, $\Phi_{out} = \Phi$ after run

$\ell = \#$ new items in run ($k - \ell$ repeats)

$$\# \text{ OPT misses} \geq \frac{\ell + \Phi_{out} - \Phi_{in}}{2} \quad (\text{per run})$$

$$E[\# \text{ NRU misses}] \leq \ell + \ell \ln k$$

let $L = \text{total } \# \text{ new items over all runs}$

(total) $\# \text{ OPT misses} \geq$

$$E[\# \text{ NRU misses}] \leq$$

let L = total # new items over all runs

(total) #OPT misses $\geq L/2$

$$E[\# \text{NRU misses}] \leq (1 + \ln(k))L$$

$$E[\text{Competitive ratio}]$$

$$= \frac{E[\# \text{NRU misses}]}{\# \text{OPT misses}} \leq$$

Randomized NRU (1-Bit LRU)

"mark" requested items

evict uniformly random unmarked items

if all cache items marked, then
remove all marks and try again

claim: randomized NRU is

$O(\log k)$ -competitive in expectation

split into "runs" whenever we reset marks

$e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, \dots$

(reset marks)

(reset marks)

(reset marks)

"run"

each run (between resets) has k distinct items:

- "new" items not in previous run
- "repeat" items also in previous run



Lecture 26

Probability
(and techniques)

Algorithms
(models and applications)

✓ Caching

Buy-or-rent

Skiing seems fun
but you don't own skis
rent or buy?

depends on
how many
times you end
up skiing





Buying a car

or take →
everywhere

Uber

lyft

Keep hard drive
spinning when
not in use?

(even more relevant
in trend towards
heterogeneous systems,
emphasis on power consumption)



model

k dollars to buy
(integer)

1 dollar per day to rent

$l = \#$ days we ski (in hindsight)

tricky: don't know l

each day we decide to buy/rent, online



k to buy, 1 per rent, $l = \#$ ski days

If we knew l :

① if $l \leq k$, rent every day

② if $l \geq k$, buy on first day

$$\text{OPT}_{\text{(in hindsight)}} = \min\{k, l\}$$



k to buy, 1 per rent, $l = \#$ ski days

$$\text{OPT} = \min \{k, l\}$$

(in hindsight)

goal: online strategy
competitive w/ OPT

Question:

(^{hint:}_{simple}) 2-competitive algorithm?



k to buy, 1 per rent, $l = \#$ ski days

$$OPT = \min \{k, l\}$$

(in hindsight)

goal: online strategy competitive w/ OPT

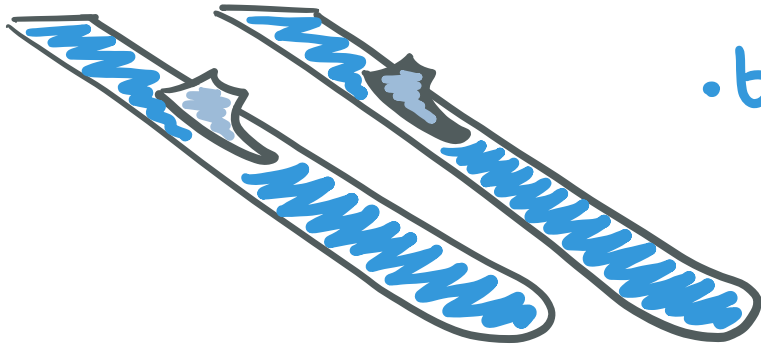
2-competitive algorithm

- First $k-1$ days: rent
- k th day: buy

$$\uparrow \text{cost} < 2 \min \{k, l\} = 2OPT$$



"Half-skis"



Suppose you can

- buy 1 ski for $k/2$ dollars
- rent 1 ski for $1/2$ dollars

(still need 2 skis each day)

OPT (in hindsight): $\min\{k, l\}$ (same as before!)

- $l \leq k \Rightarrow$ rent (both skis) each day
- $l \geq k \Rightarrow$ buy (both skis) each day

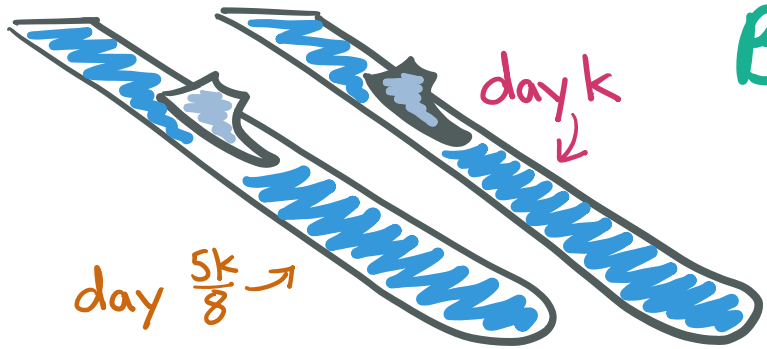
Better than 2 competitive ratio?

Suppose you can

- buy 1 ski for $k/2$ dollars
- rent 1 ski for $1/2$ dollars
(still need 2 skis each day)

OPT (in hindsight): $\min\{k, \ell\}$ (same as before!)

- $\ell \leq k \Rightarrow$ rent (both skis) each day
- $\ell \geq k \Rightarrow$ buy (both skis) each day



Better-than-2-competitive:

- Day $\frac{5k}{8}$: buy one ski
- Day k : buy 2nd ski

for $\ell = \frac{5k}{8}$, pay $\frac{5k}{8} + \frac{k}{2} = \frac{9k}{8} = \frac{9}{5} \text{OPT}$

(rent both) (buy 1)

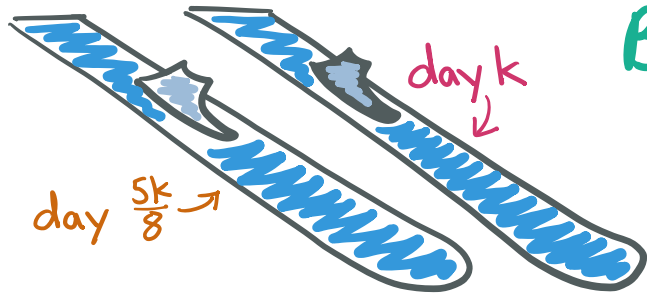
for ℓ in between, better than

for $\ell = k$, pay $\frac{5k}{8} + \frac{1}{2} \cdot \frac{3k}{8} + k = \frac{29}{16} (k = \text{OPT})$

(rent both) (rent 1) (buy both)

"Liquid skis"

- buy 1 ski for $k/2$
- rent 1 ski for $1/2$
- $\text{OPT} = \min\{k, \ell\}$



Better-than-2-competitive:

- Day $\frac{5k}{8}$: buy one ski
- Day k: buy 2nd ski

Similarly:

If you can buy skis by the "fourth",
then competitive ratio would improve more

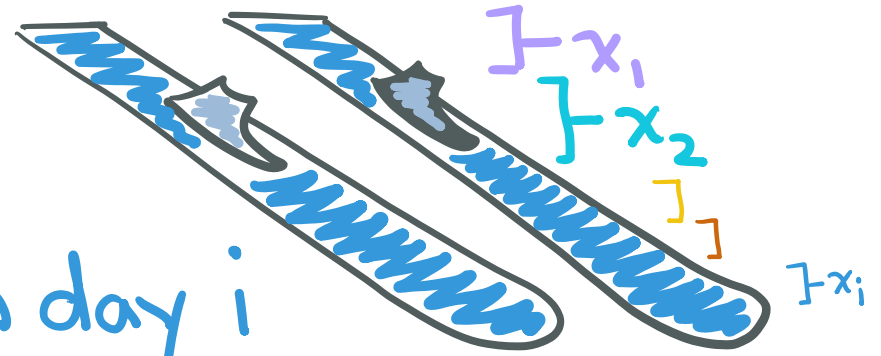
Continued to the limit: arbitrary fractions

x_1 = fractional skis bought on day 1

x_2 = fractional skis bought
on day 2

x_i = fractional skis bought on day i

y_i = fractional skis rented on day i



$$\min k(x_1 + r + x_e) + y_1 + r + y_e$$

$$\text{s.t. } x_1 + r + x_i + y_i \geq 1 \text{ for each day } i$$

x_i = fractional skis bought on day i

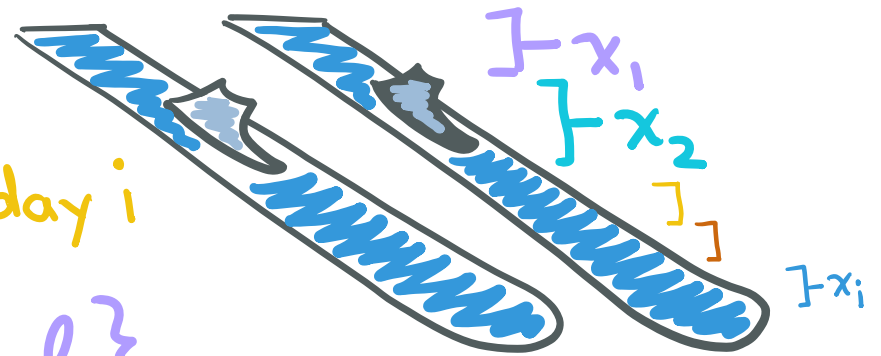
y_i = fractional skis rented on day i

$$\min k(x_1 + \dots + x_\ell) + y_1 + \dots + y_\ell$$

$$\text{s.t. } x_1 + \dots + x_i + y_i \geq 1 \text{ for each day } i$$

$$\text{OPT (in hindsight)} = \min\{k, \ell\}$$

- if $\ell \leq k$, set $x_1 = \dots = x_\ell = 0$ and $y_1 = \dots = y_\ell = 0$
- if $\ell \geq k$, set $x_1 = 1$ and $x_2 = \dots = x_\ell = y_1 = \dots = y_\ell = 0$

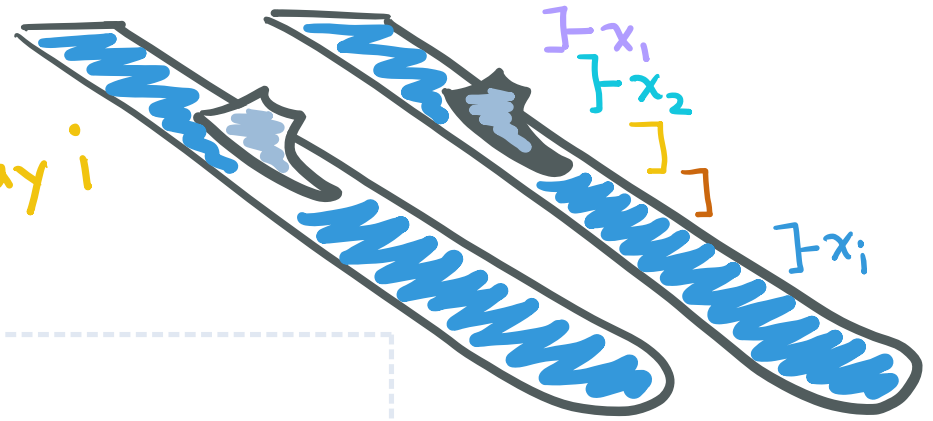


x_i = fractional skis bought on day i y_i = fractional skis rented on day i

$$\min k(x_1 + \dots + x_l) + y_1 + \dots + y_l$$

$$\text{s.t. } x_1 + \dots + x_i + y_i \geq 1 \text{ for each day } i$$

$$\text{OPT (in hindsight)} = \min\{k, l\}$$



Idea: let $\delta > 0$ TBD.

Pay $1+\delta$ every day until we completely buy on day k

(Then competitive ratio = $1+\delta$)

$$\text{day 1: } 1+\delta = kx_1 + (1-x_1) \Rightarrow x_1 =$$

$$\text{day 2: } 1+\delta = kx_2 + (1-x_1-x_2) \Rightarrow x_2 =$$

⋮

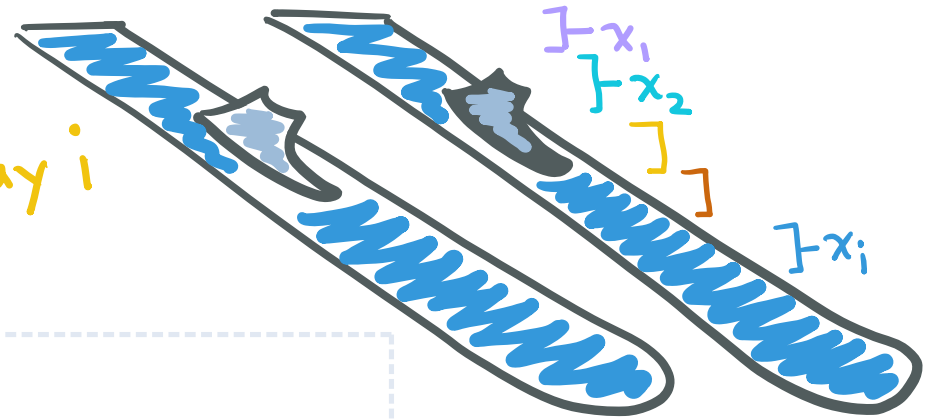
$$\text{day } i: 1+\delta = kx_i + (1-x_1 - \dots - x_i) \Rightarrow x_i =$$

x_i = fractional skis bought on day i y_i = fractional skis rented on day i

$$\min k(x_1 + \dots + x_\ell) + y_1 + \dots + y_\ell$$

$$\text{s.t. } x_1 + \dots + x_i + y_i \geq 1 \text{ for each day } i$$

$$\text{OPT (in hindsight)} = \min\{k, \ell\}$$



Idea: let $\delta > 0$ TBD.

Pay $1 + \delta$ every day until we completely buy on day k

(Then competitive ratio = $1 + \delta$)

$$x_i = \frac{1}{k-1} (\delta + x_1 + \dots + x_{i-1})$$

$$1 = x_1 + \dots + x_k = \frac{\delta}{k-1} + \frac{k}{k-1} (x_1 + \dots + x_{k-1}) = \dots$$

$$= \frac{\delta}{k-1} \left(1 + \frac{k}{k-1} + \left(\frac{k}{k-1} \right)^2 + \dots + \left(\frac{k}{k-1} \right)^{k-1} \right)$$

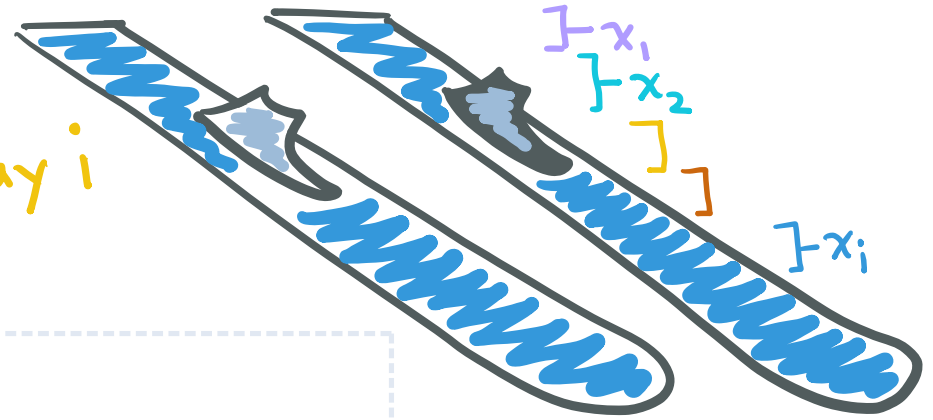
$$= \delta \left(\left(\frac{k}{k-1} \right)^k - \left(\frac{k}{k-1} \right) \right)$$

x_i = fractional skis bought on day i y_i = fractional skis rented on day i

$$\min k(x_1 + \dots + x_\ell) + y_1 + \dots + y_\ell$$

$$\text{s.t. } x_1 + \dots + x_i + y_i \geq 1 \text{ for each day } i$$

$$\text{OPT (in hindsight)} = \min\{k, \ell\}$$



Idea: let $\delta > 0$ TBD.

Pay $1 + \delta$ every day until we completely buy on day k

(then competitive ratio = $1 + \delta$)

$$x_i = \frac{1}{k-1} (\delta + x_1 + \dots + x_{i-1})$$

$$1 = \dots = \delta \left(\left(\frac{k}{k-1} \right)^k - \left(\frac{k}{k-1} \right) \right)$$

$$\delta = \frac{1}{\left(\frac{k}{k-1} \right)^k - \frac{k}{k-1}} \xrightarrow{k \rightarrow \infty} \frac{1}{e-1}$$

Competitive ratio

$$\xrightarrow{k \rightarrow \infty} \frac{e}{e-1} \approx 1.582$$

update rule

$$x_i \leftarrow \frac{1}{k}(x_i + r + x_{i-1}) + \frac{\delta}{k} \quad (\delta \text{ TBD})$$

$$y_i \leftarrow 1 - (x_i + r + x_i)$$

ownership is exponentially increasing

$$\underbrace{x_i + r + x_i}_{\text{total ownership after } i \text{ days}} = (1 + \frac{1}{k}) \underbrace{(x_i + r + x_{i-1})}_{\text{total ownership after } i-1 \text{ days}} + \frac{\delta}{k}$$

cost per day

$$kx_i + y_i = x_i + r + x_{i-1} + \delta + 1 - x_i - r - x_i$$

$$x_i \leftarrow \frac{1}{k}(x_i + m + x_{i-1}) + \frac{\delta}{k}$$

$$y_i \leftarrow 1 - (x_i + m + x_i)$$

$$\leq 1 + \delta$$

meanwhile

OPT pays 1 per day for first k days

