SARAH CLAROFF
V00836385

CSC 425 A#3

#1 a) Show there is a deterministic alg. that will give you an expected offer, y, w/in a factor $\sqrt{B}$ of the best offer m.

→ That is, $y \leq \sqrt{B}\, m$ ⇒ want to show an alg w/ $c = \sqrt{B}$

OFFLINE: accept offer m.

→ Variation on marriage problem:
- Look @ first $\frac{n}{2}$ offers, find best offer of these.
- Accept the next offer that is better than the previous best.

→ Here, however, we know the best possible is B.
→ But, we don't know for sure that someone will offer the max, B.
→ There will be a variance, so can assume that the best offer will be w/in one standard deviation of the max.

→ Assuming a uniform distribution:

Standard dev = $\sigma = \sqrt{(max - min)/12} = \sqrt{\frac{B}{12}} < \sqrt{B}$

So we will accept the first offer $\geq B - \sqrt{B}$. It will be w/in a factor of $\sqrt{B}$ of m since in the worst case, $m = B$, and we accept $B - \sqrt{B}$.

mace does $O(n \log^c n)$

12.04.2022

CSC 425 A#3

Grad

#2

a) Using LRU deterministic marking alg. for an initially empty cache of size 4. w/ cache accesses (2,3,4,1,2,5,1,3,5,4,1,2,3). What are the cache misses?

EPOCH 1:

2, 3, 4, 1, 2

Miss (first occurrence)

EPOCH 2:

5, 1, 3, 5, 4, 1

miss (unmark all, evict 2)

EPOCH 3:

2, 3

miss (unmark all, evict 3)

miss (evict 4)

MARKING:

→ Initially, all blocks unmarked.

→ When requested, if in cache, mark it.

If not in cache, then miss, check if all blocks marked (if yes unmark) ~~evic~~ then evict an arbitrary unmarked block (here, arbitrary choice is the leftmost block).

→ Then place requested block in cache + mark.

CACHE: [~~2~~, ~~3~~, ~~4~~ ~~1~~, 5, 2, 3]

$\boxed{\text{⇒ IN TOTAL, there will be 7 MISSES: 4 from accessing the first 4 elements, and then 3 from requesting an element not in the cache.}}$

12.04.2022

CSC 425 A#3

GOOD

#2 b)

What is expected # of misses by the randomized marking alg if there is initially an empty cache w/ k pages and there are page requests $1,2,3,\dots,k+1,1,2,3,\dots,k+1$ for a total of $n >> k$ page requests?

* each epoch will have exactly one new request!

For example, $K=4$:

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

→ each epoch contains exactly K unique requests, where the first request is the only new one.

→ # epochs $= \lceil \frac{n}{K} \rceil$

FOLLOWING CLRS ANALYSIS OF THRM 27.5 [Intro to Algorithms, 4th Ed, Section 27.3]

- Take any epoch $i > 1$ with K unique requests and $r_i = 1$ new request, (so $K-1$ old requests)
- Prob that j^{th} old request results in a miss has upper bound: $\frac{r_i}{(K-j+1)} = \frac{1}{(K-j+1)}$
- Let $Y_{ij} = I\{\text{the } j^{th} \text{ old request in epoch } i \text{ results in miss}\}$

$Z_{ij} = I\{j^{th} \text{ new request " " " "}\} = 1$ Since new request is always a miss.

X_i = # misses in epoch i

$$E[X_i] = E\left[\sum_{j=1}^{K-r_i} Y_{ij} + \sum_{j=1}^{r_i} Z_{ij}\right] = E\left[\sum_{j=1}^{K-1} Y_{ij}\right] + 1$$

Since one new request per epoch.

$$\leq \sum_{j=1}^{K-1} \frac{1}{K-j+1} + 1 = H_K$$

⟹ Expected # misses in one epoch i (H_K = k^{th} Harmonic number).

Over $\frac{n}{K}$ epochs, $n >> K$:

X = random var, # total cache misses

$$E[X] = E\left[\sum_{i=1}^{n/K} X_i\right] = \sum_{i=1}^{n/K} E[X_i] \leq \sum_{i=1}^{n/K} H_K = \boxed{\frac{n}{K} H_K}$$

11,27,2022

CSC 425 A#3
GOOD

#3 STREAMING

a) Suppose we have the stream: $(a_1 2, a_2 2, a_3 2, \dots)$

$a_1 \; x_2 \; a_2 \; x_2 \; a_3 \; x_2 \; a_i x_2$

Taken from elements $X = \{1, 2, \dots, N\}$

$x_1 \; x_2 \; x_i \; x_n$

with weights $A = \{a_1, a_2, \dots, a_N\}$, logn bits in length

Then:

$B = \sum_i a_i i = a_1(2) + a_2(2) + \dots + a_i(2) = 2(a_1 + a_2 + \dots + a_i)$

$C = \sum_i a_i = a_1 + a_2 + \dots + a_i$

$\frac{B}{C} = 2 = i \Rightarrow$ CONSTANT

→ If B/C = constant, then we are only seeing one type of item, and that item's index is equal to B/C.

→ However, if we have no other use for B/C, instead of calculating it for each element, we can just have one bit $= i$ and check if that bit is equal to the new i in each round.

GOOD

#3 b)

Show that if we also maintain $D \sum_i a_i i^2$ then $B^2 = C*D$ iff we've seen exactly one type of element. Show both ways.

$C*D = \sum_i a_i \sum_i a_i i^2 = \sum_i a_i^2 i^2 = B^2$

① SHOWING $(B^2 = C*D) \Rightarrow$ (exactly one element)

Take $D = \sum_i a_i^2 i$. Suppose $B^2 = C*D$, where $i \in X = \{x_1, x_2, \ldots, x_n\}$.

Proof by contradiction:

Assume the statement $(B^2 = CD) \Rightarrow$ (stream contains exactly one type of element) is false.
That is, $B^2 = CD$ AND stream contains >1 or 0 types of elements.
If we have >1 types of element:

$$B^2 = (\sum_i a_i i)^2 = (a_1 x_1 + \ldots + a_n x_n)(a_1 x_1 + \ldots + a_n x_n) = a_1^2 x_1^2 + a_1 x_1 a_2 x_2 + \ldots + a_1 x_1 a_n x_n + a_2 x_2 a_1 x_1 + \ldots + a_n^2 x_n^2$$
$$\neq a_1^2 x_1^2 + a_2^2 x_2^2 + \ldots + a_n^2 x_n^2$$

We will show this via a counter example:

Suppose $n = 4$, so weights are $\lg 4 = 2$ bits in length (i.e. $A = \{0, 1, 2, 3\}$)

Take the stream $(2(2), 3(2), 2(3), 3(1))$

Then $B^2 = (\sum_i a_i i)^2 = (2(2) + 3(2) + 2(3) + 3)^2 = 361$

$C*D = \sum_i a_i \sum_i a_i i^2 = (2+3+3+2)(2(4) + 3(4) + 2(9) + 3) = 410 \neq 361$

→ Contradiction, cannot have $B^2 = C*D$ and >1 type of element.

∴ If $B^2 = CD$, then we've seen exactly one type of element.

② SHOWING (exactly 1 type of element) $\Rightarrow B^2 = CD$

If we have exactly 1 type of element, call it x, then $x_i = x$ for all $a_i x_i$

$B = \sum_i^n a_i i = \sum_i a_i x = x \sum_i a_i$

$D = \sum_i^n a_i i^2 = \sum_i a_i x^2 = x^2 \sum_i a_i$

$B^2 = (x \sum_i a_i)^2 = x^2 \sum_i a_i \sum_i a_i = \sum_i a_i \sum_i a_i x^2 = C*D$

∴ If we have exactly 1 type of element, $B^2 = C*D$.

∴ By ① and ②, $B^2 = C*D$ iff the stream has exactly one type of element. ▪

alg to do this? Note that we need to maintain B, C, T for a tree and we need the sums of the edges in the sample to cancel out when an edge is incident to 2 diff nodes in the same tree. Recall $T = \sum a_i \bmod p$

a) Explain how the Monte Carlo alg needs to be modified. Will XOR work in place of ordinary addition for the non-negative numbers? Why or why not?

REFERENCES:

[1] Gibb, D., Kapron, B., King, V., Thorn, N., Dynamic graph connectivity with improved worst case update time and sublinear space, Sept 2015.

[2] Cormode, G., Firmani, D., A Unifying Framework for l_0-Sampling Algorithms, 2014

Can use 1-sparse recovery to verify the name of the edge returned by Search, instead of an odd hash function.

Once Search(T) has returned $z_i = \sum_{x \in T} s_i(x) = e_i$, we want to verify that it's the name of an edge in the cut, rather than the sum of multiple names. A pairwise ind. function randomly splits the cutset into 2 groups, and then the 1-sparse recovery is applied to test if there is at least one element in each group. This is repeated multiple times, and if it fails every time (meaning one group was empty) then there is likely 1 element/name in the cutset. We can relate to the 1-sparse recovery as follows:

Let p = large prime, $z \in \mathbb{Z}_p$ randomly chosen, $1 \le i \le \binom{n}{2}$ - possible edges?, $a_i = e_i$

$B = \sum_i e_i \, i$, $C = \sum_i e_i$ $\quad T = \sum_i e_i \cdot z^i \bmod p$

If 1-sparce (ie. if there is only one element/name in the cutset), then test

$T = C z^{B/C} \bmod p$ $\quad$ If 1-sparse, then the test will always return a positive #.

If not 1-sparce, the above test will fail (return a neg. #) w/ prob. $1 - \frac{\binom{n}{2}}{p}$

XOR will work in place of addition because we are still concerned about ~~the~~ whether there is only one type of element at the end.

b) Each vector size $O(\log n)$ $\Rightarrow$ all vectors space $O(n \log n)$

prob. of Search(T) returning correct/valid edges: $1-\frac{1}{n^c}$, $c>0$ constant.

To maintain the alg's. sublinear space, do at most n^d updates, $d>0$ constant

For p to be a suitably large prime, $p > n^{c+d+2}$ (for vector size, $p > \binom{n}{2}$, $\binom{n}{2} = \frac{n(n-1)}{2}$)

To represent p in binary, will need $\log(n^{c+d+2}) = (c+d+2)\log n$ bits

$z \in \mathbb{Z}_p$, need $(c+d+2)\log n$ size words to represent each z.

$T = \sum_i^{\binom{n}{2}} e_i z^i \bmod p \Leftrightarrow O(\log n)\, O((c+d+2)^2 \log^2 n)\, O((c+d+2)\log n) = O((c+d+2)^3 \log^4 n)$ space

Over n nodes: space is $O(n(c+d+2)^3 \log^4 n) = \boxed{O(n \log^4 n)}$

c) How much time?

[1], Lemma 3.6: Time is $O(top^2 * t(n) + top * \log n)$ where $t(n)$ = update time, top = time for F_{top}

$top = O(\log n)$

assuming $(\log n)(\log n) = O(\log n)$,

update time: Dominated by T

adding is $O(1)$

$T = \sum_i e_i z^i \bmod p = \binom{n}{2} O(\log n)\, O((c+d+2)^2 \log^2 n)\, O(\log n) = O(\log n)$

$\Rightarrow$ over n nodes, $\log n$ levels: $t(n) = O(n \log^2 n)$

$\therefore$ Time is $O(\log^2 n (n \log^2 n) + \log n * \log n) = \boxed{O(n \log^4 n)}$