Streaming Algorithms

Meng-Tsung Tsai

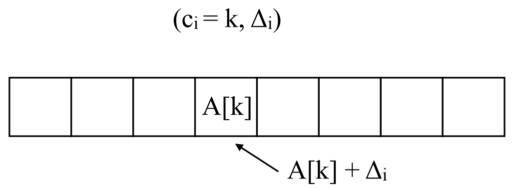
05/11/2018

Uniquess Testing

Input: Let $A[1..U]$ be an array of length U. Initialize $A = \{0\}$. Give a sequence of updates (c_i, Δ_i) where the i-th update asks to add Δ_i to $A[c_i]$. Assume that $A \geq 0$ after every update.

Output: "0" if $A = 0$, or "1" if there is exactly one index i in $[U]$ so that $A[i] > 0$, or otherwise "2+".

Goal: $O(1)$ working space.



Algorithm

Initialize three counters C_0, C_1, and C_2 as 0.

```
foreach update (c_i, Δ_i){
    C_0 += Δ_i;
    C_1 += Δ_i c_i;
    C_2 += Δ_i(c_i)^2
}

if((C_2)(C_0) = (C_1)^2){ // Cauchy-Schwarz Ine.
    output (C_0 == 0) ? "0" : "1"; // the non-zero coordinate is C_1/C_0
}else{
    output "2+";
}
```

Sample a Coordinate i with A[i] > 0

Let $||A||_0$ be the zero norm of A, i.e. $\sum_i 1[A[i] \neq 0]$.

Run the Uniquess-Test and we get $||A||_0$.

```
Switch(||A||_0){
    Case 0: // ||A||_0 == 0
        return "A = 0";
    Case 1:
        return C_1/C_0;
    Case 2:
        return ?;
    ...
}
```

Sample a Coordinate i with A[i] > 0

Idea: assign "a random bit" to each coordinate, and ignore those updates whose coordinate isn't associated with a 0-bit.

If $||A||_0 = 2$, then with probability 1/2, the Uniqueness-Test can return one of them.

If we run k independent copies of the above, then with probability $1-1/2^k$, some Uniqueness-Test can return one of them.

To store these random bits, we need space of n bits.

Sample a Coordinate i with A[i] > 0

2-wise independent randomness:

Step1. choose a prime $p \gg U$.

Step2. pick two independent random numbers $a, b \in [U]$.

Step3. associate the x-th coordinate with $(h_{a,b}(x) = ax + b \pmod p) \pmod 2$ for each x in $\{1, 2, ..., U\}$.

// for each $x \neq y \in U$, $h_{a,b}(x)$ and $h_{a,b}(y)$ are independent.
// note that $h_{a,b}(1), h_{a,b}(2), ..., h_{a,b}(U)$ are pairwise independent rather
// than mutually independent.

To store these random bits, we need O(log U) bits.

Sample a Coordinate i with A[i] > 0

Remark. For $||A||_0 = 2$, there exists a data structure that can sample a coordinate i with A[i] > 0 with probability 1/2 using O(log U) bits.

Remark. For $||A||_0 = 2$, there exists a data structure that can sample a coordinate i with A[i] > 0 with probability $1-1/2^k$ using O(k log U) bits.

What if $||A_0||$ is large?

Sample a Coordinate i with A[i] > 0

2-wise independent randomness:

Step1. choose a prime $p \gg U$.

Step2. pick two independent random numbers $a, b \in [U]$.

Step3. associate the x-th coordinate with $(h_{a,b}(x) = ax + b \pmod p) \pmod r$ for each x in $\{1, 2, ..., U\}$.

// with probability 1/r, the updates with coordinate i are not ignored.

Sample a Coordinate i with A[i] > 0

Let S be the set of cooridnate i whose A[i] > 0 and isn't ignored given $h_{a,b}$ and r.

Expectation.

$$\mu = E[|S|] = ||A||_0/r.$$

Variance.

$$\sigma^2 = Var[|S|] = ||A||_0 ((1-1/r)(1/r)) < E[|S|].$$

Chebyshev Inequality.

$$Pr[||S|-\mu| \geq \lambda\sigma] \leq 1/\lambda^2.$$

Remark.

$$Pr[|S| \in [\mu-2\mu^{1/2}, \mu+2\mu^{1/2}]] \geq 3/4.$$

Sample a Coordinate i with A[i] > 0

```
ℓ0-Sampler(){

  pick a prime p ≫ U;
  let h(x) ← h_{a,b}(x) (mod r) with random a, b in {1, 2, ..., p};

  foreach update (c_i, Δ_i){
    if(h(c_i) == 0){ // S = {x ∈ [U]: h(x) = 0}
      update "a data structure" for S;
    }
  }
  if(|S| ∈ [μ-2μ^{1/2}, μ+2μ^{1/2}]){ // happens w.p. ≥ 3/4
    return a random element e ∈ S;
  }
}
```

For each i with A[i] > 0,
$Pr[sample = i] \geq (1/r)*(3/4)*(1/|S|) = \Omega(1/||A||_0)$.

The Data Structure D

Let D be k copies of Uniquess-Test components.

Each x in S will be used to update exactly one random copy $C_{d(x)}$,

$$\text{where } d(x) = d_{a,b}(x) = (ax + b \pmod p) \pmod k).$$

If all elements in S don't collide, we can return a random element from S, so we need to pick k to be large enough.

Pr[a certain pair e_1, e_2 collide] = 1/k.

Pr[some pair e_1, e_2 collide] $\leq |S|^2/k$.

Pr[this procedure fails] $\leq 1/4 + (\mu+2\mu^{1/2})^2/k \leq 1/2$ if $k = 4(\mu+2\mu^{1/2})^2$.

Sample a Coordinate i with A[i] > 0

```
ℓ0-Sampler(){

  pick a prime p ≫ U;
  let h(x) ← h_{a,b}(x) (mod r) with random a, b in {1, 2, ..., p};

  foreach update (c_i, Δ_i){
    if(h(c_i) == 0){ // S = {x ∈ [U]: h(x) = 0}
      update "a data structure" for S;
    }
  }
  if(|S| ∈ [μ-2μ^{1/2}, μ+2μ^{1/2}]){ // happens w.p. ≥ 3/4
    return a random element e ∈ S;
  }
}
```

For each i with A[i] > 0,
$Pr[sample = i] \geq (1/r)*(1/2)*(1/(\mu+2\mu^{1/2})) = \Omega(1/||A||_0)$.

Sample a Coordinate i with A[i] > 0

Remark. There exists (explicitly given) a $(4(\mu+2\mu^{1/2})^2)$-space data structure so that

with probability $\geq 1/2$

it returns a coordinate i with $A[i] > 0$,
noting that each coordinate i with $A[i] > 0$ is returned with probability $\Omega(1/\|A\|_0)$

or otherwise

it returns "Fails."

Issues. We don't know what μ is.

Sample a Coordinate i with A[i] > 0

Set $\mu = 100$, and thus D has $4*(100+20)^2$ copies of Uniqueness-Test components.

Run in parallel with r = 1, 2, 4, 8, ..., log U. There exists some r so that the corresponding $\mu = \|A\|_0/r \leq 100$, so the above works.

Programming Assignment #2

Input: Given a sequence of edge insertions and deletions of an n-node graph and it is asserted that the final graph has $\leq 2n$ edges.

Output: the final graph with high probability.

Run O(n log n) ℓ_0-samplers in parallel.