Streaming Algorithms

Meng-Tsung Tsai

03/23/2018

Written Assignment #1

It will be announced on e3new.nctu.edu.tw tonight. You need to submit your wrtiteup by 23:59, Apr 10, and the writeup shall be in pdf, generated by latex.

If you didn't use latex before, you may read the guidelines on http://ctan.math.illinois.edu/info/lshort/english/lshort.pdf.

References

- "Pairwise Independence and Derandomization," Luby and Wigderson (2005)
- "Sketch Techniques for Approximate Query Processing," Cormode

Elements in Common

Problem Definition

Input: a sequence of 2n elements $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ where a_i's, b_i's are distinct elements in $[U] = \{1, \ldots, U\}$. Let $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_n\}$.

Output:

$$|A \cap B|$$

In words, how many elements of A and B are in common?

Goal: using $o(n \log |U|)$ bits.

Observation

Let $\boldsymbol{\delta}_x$ be the "smallest" element in $A \cap B$.

Let $\boldsymbol{\delta}_y$ be the "smallest" element in $A \cup B$.

If we order the elements in U by a fully random permutation, what is the probability that $\boldsymbol{\delta}_x = \boldsymbol{\delta}_y$.

$\Pr[\boldsymbol{\delta}_x = \boldsymbol{\delta}_y] = |A \cap B|/|A \cup B|$. (Why?)

Expectation

The expectation of $|A \cap B|$ = ? (in terms of $\Pr[\boldsymbol{\delta}_x = \boldsymbol{\delta}_y]$ and n)

Expectation

The expectation of $|A \cap B|$ = ? (in terms of $\Pr[\boldsymbol{\delta}_x = \boldsymbol{\delta}_y]$ and n)

By PIE, $|A \cup B| = |A| + |B| - |A \cap B| = 2n - |A \cap B|$.

Since $\Pr[\boldsymbol{\delta}_x = \boldsymbol{\delta}_y] = |A \cap B|/|A \cup B|$, we get

$$|A \cap B| = 2n \Pr[\boldsymbol{\delta}_x = \boldsymbol{\delta}_y]/(1+\Pr[\boldsymbol{\delta}_x = \boldsymbol{\delta}_y]).$$

In other words, given an accurate estimate (resp. exact value) of $\Pr[\boldsymbol{\delta}_x = \boldsymbol{\delta}_y]$, we can derive an accurate estimate (resp. exact value) of $|A \cap B|$.

Algorithm

Sample a fully random function f.

let $\boldsymbol{\delta}_1 = \min_i f(a_i)$ and $\boldsymbol{\delta}_2 = \min_i f(b_i)$.

then $\Pr[\boldsymbol{\delta}_1 = \boldsymbol{\delta}_2] = \Pr[\boldsymbol{\delta}_x = \boldsymbol{\delta}_y]$. (Why?)

Can we calculate the exact value of $\Pr[\boldsymbol{\delta}_1 = \boldsymbol{\delta}_2]$?

Issues

It takes $\Omega(|U| \log |U|)$ bits to represent a sample of fully random function.

A Small Sample Space

Let f be a function sampled uniformly at random from the family

F ={
$s_0 : U \to U$ so that $s(x) = x \pmod{|U|}$,
$s_1 : U \to U$ so that $s(x) = x+1 \pmod{|U|}$,
...
$s_{|U|-1} : U \to U$ so that $s(x) = x+|U|-1 \pmod{|U|}$
}

Ordering elements in U by the above random f, $\Pr[\boldsymbol{\delta}_1 = \boldsymbol{\delta}_2]$ remains the same.

A Small Sample Space

Let f be a function sampled uniformly at random from the family

F ={
$s_0 : U \to U$ so that $s(x) = x \pmod{|U|}$,
$s_1 : U \to U$ so that $s(x) = x+1 \pmod{|U|}$,
...
$s_{|U|-1} : U \to U$ so that $s(x) = x+|U|-1 \pmod{|U|}$
}

Moreover, f can be represented in $O(\log |U|)$ bits.

A Small Sample Space

To obtain the exact value of $\Pr[\boldsymbol{\delta}_1 = \boldsymbol{\delta}_2]$, we can simply calcualte

$$\sum_{g \in F} \mathbf{1}[\boldsymbol{\delta}_1 = \boldsymbol{\delta}_2 \mid f = g]/|F|.$$

In this way, we can employ $|U|^{1/2}$ processors and each of them calculates $\mathbf{1}[\boldsymbol{\delta}_1 = \boldsymbol{\delta}_2 \mid f = g]$ for $|U|^{1/2}$ distinct g's using $O(\cancel{|U|} \log |U|)$ bits. Then, sum up the individual counts.

However, $|U|$ may be $\gg n$.

Reduce U to [cn] for some constant c

This can be achieved by any perfect hashing scheme, for example FKS hashing.

Applications

Given an undirected graph $G = (V, E)$, count the number of triangles in G, i.e. # of unordered triples (u, v, w) so that (u, v), (v, w), (w, u) in E.

```
c_Δ ← 0;

foreach edge (u, v) in E{
   c_Δ += |N(u) ∩ N(v)|; // neighbor sets of node u and v
}

output c_Δ;
```

On a RAM, c_Δ can be computed in $O(m^{3/2})$ time.
Now you can have an accurate estimate more efficiently
or calculate the exact value in parallel.

Exercise 1

What if a_i's and b_i's are not all distinct?

We need to estimte |A| and |B|, i.e. # of distinct elements in a set,
a kind of frequency moment.

Maximum Cut

Problem Definition

Input: an undirected graph $G = (V, E)$.

Output: a partition of V into $X \cup Y$ so that

$$|\{(u, v) \in E : u \in X, v \in Y\}|$$

is maximized.

This problem is NP-hard, but we can have an efficient 2-approximation.
(Recall the BB used in the problem of spliting graphs)

BB for Splitting Graphs

Place each node in {Left, Right} uniformly at random. Let X_e be the indicator varaible whether edge e is a crossing edge with respect to the random partition.

Clearly, $E[X_e] = 2 * 1/2 * 1/2 * 1 = 1/2$.

Thus, $E[\sum_{e \in G} X_e] = \sum_{e \in G} E[X_e] = m/2$.

Some random partition of nodes induces at least m/2 crossing edges.

Issues

There are 2^n random paritions, and each of them requires $\Omega(n)$ bits to represent.

A Small Sample Space

Let f be a function sampled uniformly at random from the family

$F = \{a, b \in \mathbf{Z}_p : f(x) = ax+b \pmod{p}\}$ where p is a prime > |V|.

Partition nodes in V by the above random f (mod 2), $E[\sum_{e \in G} X_e]$ remains m/2.

Algorithm

```
foreach function f in F{
  if ∑e∈G Xe ≥ m/2 given f
    output f; // such an f must exist
}
```

The running time is $O(n^2 m)$. Given $O(n^2 m)$ processors, this can be done in O(log n) time.

Observation

Such a partition must have $||X|-|Y|| = O(\log |V|)$, i.e. a balanced partition.

Exercise 2

Can you find a even smaller sample space for the abovementioned derandomization process?

Quasi-Polynomial Time

Quasi-polynomial time algorithms

We say an algorithm quasi-polynomial time if it runs in $2^{O(\text{poly log n})}$ time.

If an algorithm runs in quasi-polynomial time, then people tend to believe that it cannot be NP-hard due to exponetial time hypothesis.

In other words, if you find a problem that can be solved probabilistically using O(poly log n)-wise independence, then it is unlikely to be an NP-hard problem by abovementioed derandomization techniques.