COMS E6998-9: Algorithms for Massive Data (Spring'19) April 11, 2019

Lecture 22: MPC Model: Distinct Elements and Sorting

Instructor: *Alex Andoni* Scribes: *Andrew Lee, Jonathan Zhang*

1 Introduction

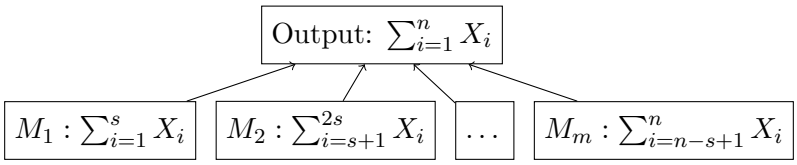
Today's lecture presented some problems and corresponding algorithms in the massively parallel computation (MPC) framework that was inspired by Map-Reduce. In this lecture, we cover MPC algorithms for:

- Sum of n integers
- Number of distinct elements (approximate)
- Prefix-sum
- Sorting
- Number of distinct elements (exact)

From the last class, we refer to the space per machine as s, the number of machines as m, the input size as N. Typically, we want $ms \approx \Theta(N)$. Our cost is measured in the number of rounds required for computation (p-time).

2 Sum of n integers

In this problem, suppose $s = \sqrt{n}$. We have m machines that receive input directly from the stream. Machine 1 will receive stream elements $X_1, ..., X_s$, machine 2 will receive $X_{s+1}, ..., X_{2s}$, and so on. Each machine m_i computes a local sum based on the stream elements it receives, and then passes the sum up to another machine (computation goes bottom up):

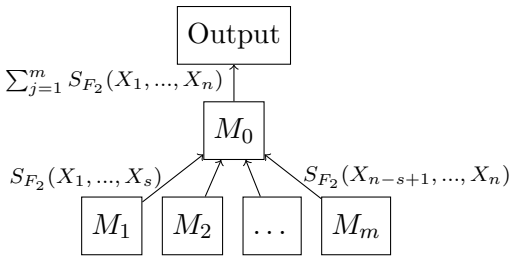


The amount of input to the machine in the second round has input $m = \sqrt{n} = s$. This algorithm thus solves sum of n integers in the case that $s = \sqrt{n}$ in $O(1)$ p-time.

In the case that $s << \sqrt{n}$, we employ a similar strategy, except we build a tree of arity s and depth $\lg_s n = O(1)$. Thus the p-time is still constant since $O(\lg_s n) = O(1)$. Such a tree can be seen as a circuit, as the maximum arity is analogous to an upper bound on the fan-in of a circuit.

3 Counting Distinct Elements

Suppose we have n items from a universe $\{1, ..., U\}$. We can compute the number of distinct elements in $X_1, ..., X_n$ and the 2nd moment of the frequency vector to a factor of $(1 + \epsilon)$ with probability at least 99% in $O(1)$ rounds using linear sketches. Like in the previous problem, machine 1 will receive stream elements $X_1, ..., X_s$, machine 2 will receive $X_{s+1}, ..., X_{2s}$, and so on.



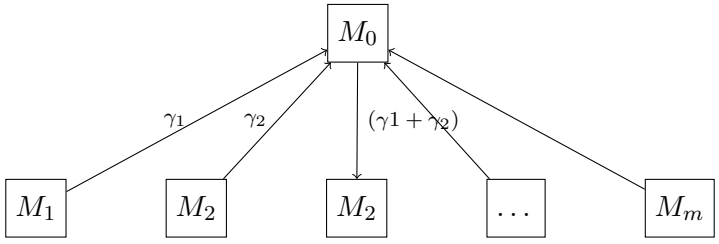
The size of each sketch $S_{F_2}(X_{ks+1}, ..., X_{(k+1)s})$ is $O(\frac{1}{\epsilon^2} \lg n)$. Thus, the input size from each machine in the first round to the machine in the 2nd round is $\leq M * O(\frac{1}{\epsilon^2} \lg n) \leq s$ if $s \geq \sqrt{n} \lg n$.

Again, in the case that $s < \sqrt{n} \lg n$, do the same thing as before, where we construct a tree of arity $\epsilon^2 s / \lg n$, resulting in a depth and p-time of $O(\lg_{\epsilon^2 s / \lg n} n) = O(1)$. Note that this algorithm gives an approximation of factor $(1 + \epsilon)$. The exact algorithm, which occurs at the end of this lecture, requires the sorting algorithm.

4 Prefix-Sum

The input comes as integers $X_1, ..., X_n$. The output will be in the form $\sigma_1, ..., \sigma_n$ where $\sigma_i = \sum_{j=1}^{i} X_j$. One obstacle in the framing of this problem is that the numbers do not necessarily come in order, and come on with indices instead: $(1, X_1), ..., (n, X_n)$. Thus, the right numbers may not be initially stored in the correct machines (using the mapping of elements to machines we used in the previous to problems). Thus, the algorithm will proceed as follows (with $s = \sqrt{n}$ and the M_i responsible for $X_{(i-1)s+1}, ..., X_{is}$):

1. Each M_i sends all its input (i, X_i) To the machines that are responsible for each input.
2. Each M_i computes a local sum $\gamma_i = \sum_{j@M_i} X_j$
3. M_i sends γ_i to M_1
4. M_1 sends to each M_i the quantity $\sigma_{s(i-1)} = \sum_{j=1}^{i-1} \gamma_j$
5. M_i computes all $\sigma_{i(s-1)-j} = \sigma_{i(s-1)} + \sum_{k=1}^{j} X_{i(s-1)+k}$ where $j \in \{1, ..., s\}$.



If $s << \sqrt{n}$, use $s - ary$ trees in steps 2 and 3, where the γ_i values must traverse the entire depth of the s-ary tree (up the tree for step 2 and down the tree for step 3).

5 Sorting

We discuss the problem of sorting integers across multiple machines. In this problem, suppose $s = O(n^{2/3} \log n)$ and $m = O(n^{1/3}/\log n)$. We have the following setup:

input: $x_1, ..., x_n$

output: $(x_1, z_1), ..., (x_n, z_n)$ where z_i is the rank of x_i

We will use Quicksort with m pivots as described below:
Algorithm:

0. Every item is marked as 'pivot' with probability $\delta = \frac{n^{1/3} \log n}{n}$

1. All machines send their chosen pivot candidates to a master machine M_1. M_1 needs to pick a pivot for each each length $n^{2/3}$ interval.

   option 1:

2. M_1 sends all received pivots to all machines. The output size is $O(n^{1/3} \log n)$.

3. For each pivot p received from M_i, every machine M_i computes C_{ip} = number of items less than pivot p on machine M_i.

4. Each machine sends C_{ip} to M_1. M_1 computes the corresponding ranks for each pivot and proceeds to choose a pivot p_i in each range $[(i-1)n^{2/3}, in^{2/3}]$. We claim that each range will have a pivot with high probability.

   $$\mathbb{E}[\text{number of pivots in a range}] = n^{2/3}\delta = \log n$$

   Using Chernoff bounds, we can bound that at least one pivot exists in each range with high probability.

5. M_1 sends final chosen pivots $p_1, ..., p_{n^{1/3}}$ to each machine.

6. Each machine, for input x_i sends x_i to machine M_j for j such that

   $$p_{j-1} \leq x_i \leq p_j$$

   and

   $$p_0 = -\infty$$
   $$p_{n^{1/3}+1} \triangleq \infty$$

   Note that since $s = O(n^{2/3} \log n)$, there is enough space on each machine to store the respective data. Namely the fan in is $2n^{2/3}$.

7. Each M_i then sorts it's data locally and assigns $\text{rank}(x_i) = \text{rank}(p_{j-1}) + \text{local\_rank}(x_i) - 1$ and we are done.

   option 2: compute ranks within pivots

2 M_1 chooses pivots $p_1, ..., p_{n^{2/3}}$ which have ranks $k\frac{i}{n^{1/3}}$ where $k =$ number of pivots. Requires more complicated proof not covered in class

Remark: If $s < n^{2/3} \log n$ then we apply an s-ary tree strategy to sort recursively.
Algorithm:

1. mark $\frac{n}{s} \log n$ items as pivots
2. sort the chosen items recursively
   - $n' = \frac{n}{s} \log n$
   - $n'' = (\frac{n}{s} \log n)(\frac{1}{s} \log n)$

     ...
   - $n^{(j)} = n \left(\frac{\log n}{s}\right)^j$

If $j \approx \log_s n$ then chosen pivots fit on a machine.

6 Number of Distinct Elements (exact)

We can exploit the sorting algorithm
Algorithm:

1. sort $x_1, ..., x_n$
2. Each M_i sends to M_1
   - number of distinct elements (can be found with linear scan)
   - first item
   - last item
3. M_1 then counts the total number of distinct elements, subtracting one when the first and last item of adjacent machines M_i and M_{i+1} match. We need this final step to prevent double counting repeated elements that cross multiple machines (e.g. half the input is 7).