Range-Efficient Computation of F_0 over Massive Data Streams

A. Pavan, Iowa State University

Srikanta Tirthapura, Iowa State University

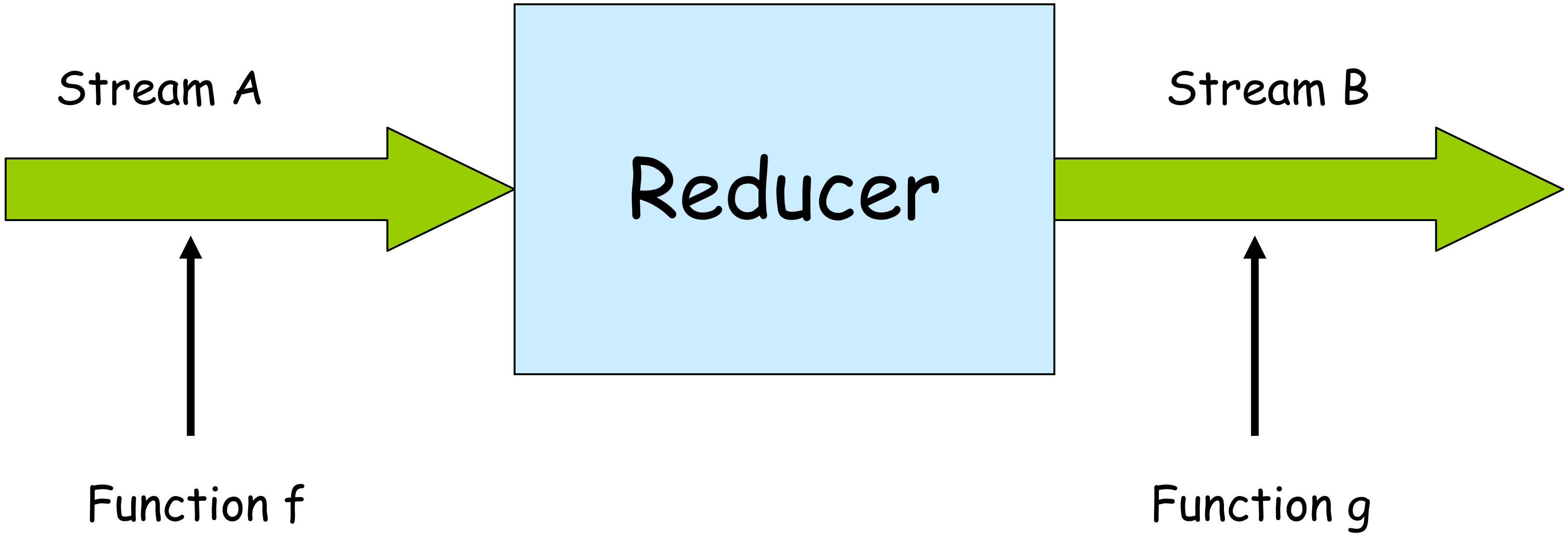
Data Streams

- Network Monitoring
  - All packets on a network link
  - Example Statistics:
    - Average packet size
    - Number of different source-destination pairs
- Sensor Data
  - Average (mean, median) and other aggregates of sensor readings
- Web-Click Streams
  - Frequently requested items
  - Change in request patterns over time
- One-pass algorithm is useful for data stored on disk

Data Stream Characteristics

- Massive Data Sets, One-pass processing
- Limited workspace
  - Much smaller than the size of the data
  - Typically poly-logarithmic in the size of data
- Fast Processing Time per item
  - Constant or logarithmic in data size
- Provide approximate answers to aggregate queries
  - Frequency Moments of Data (F_0, F_2, etc)
  - Quantiles, Distances between streams

Reductions Between Data-Stream Problems

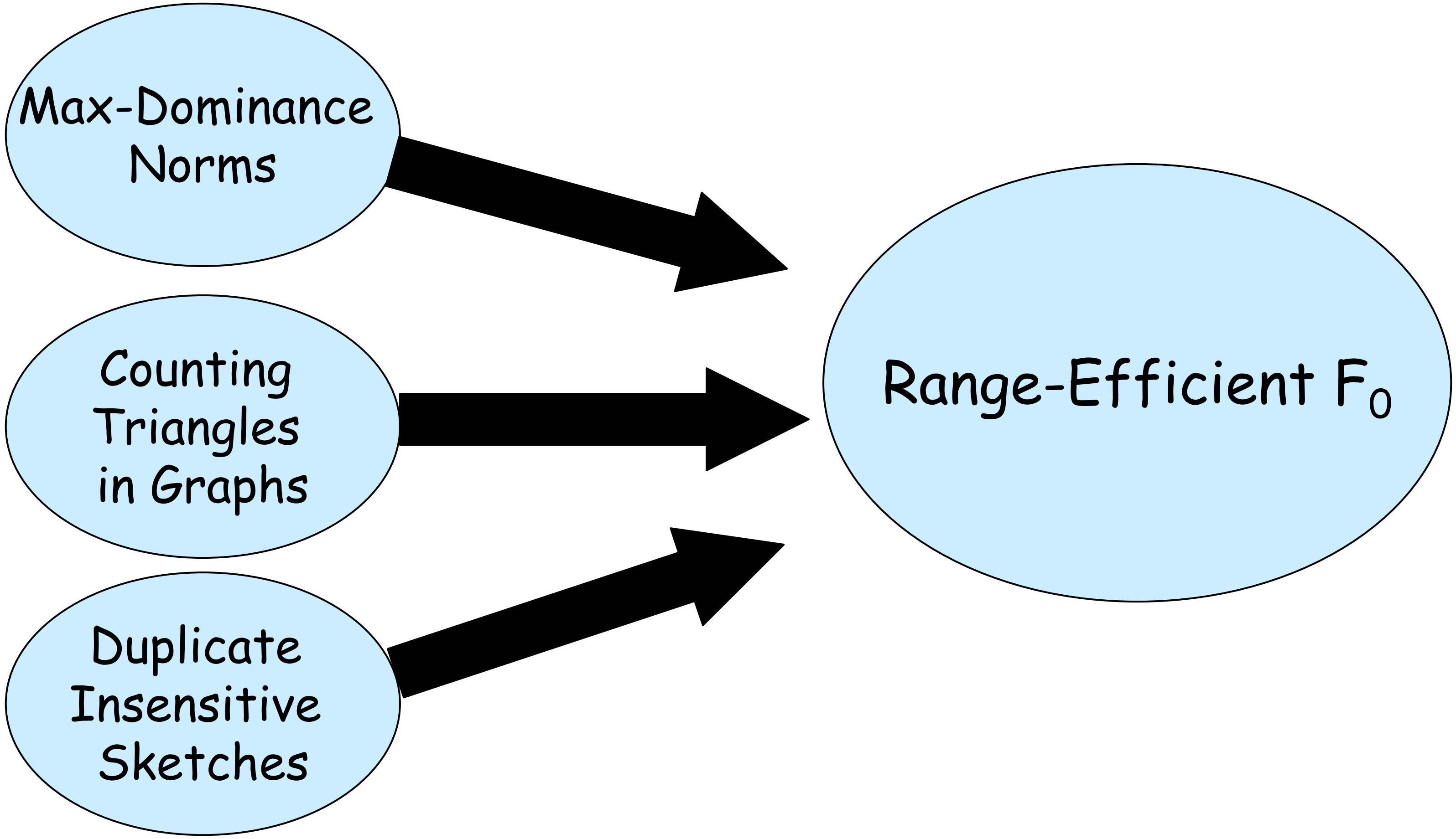


$$g(B) = f(A)$$

Reductions

- Single Element of Stream A may generate a *list of elements* in Stream B
- Algorithm on Stream B
  - Inefficient to process elements of a list one by one
  - List-efficient algorithms process the list quickly
  - Range-efficient algorithms process a range of integers quickly

Reductions to Range-Efficient F_0



Range Efficient F_0

Input Stream

Sequence of ranges $[l_1,r_1], [l_2,r_2] \ldots [l_m,r_m]$

for each i, $0 \leq l_i \leq r_i < n$, and l_i, r_i are integers

Output:

Return $|[l_1,r_1] \cup [l_2,r_2] \cup \ldots \cup [l_m,r_m]|$

i.e. number of distinct elements in the union (F_0)

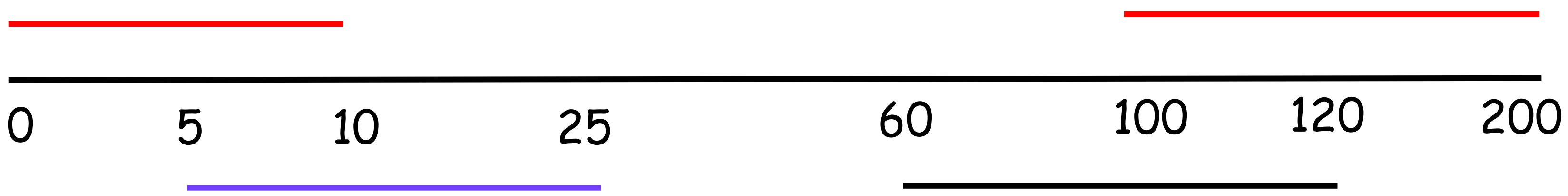
Constraints:

- Single pass through the data
- Small Workspace
- Fast Processing Time

Example

Stream:

[0,10], [100,200], [60, 120], [5,25]



F_0 is:

$|[0,25] \cup [60,200]| = 167$

Approximate Answers

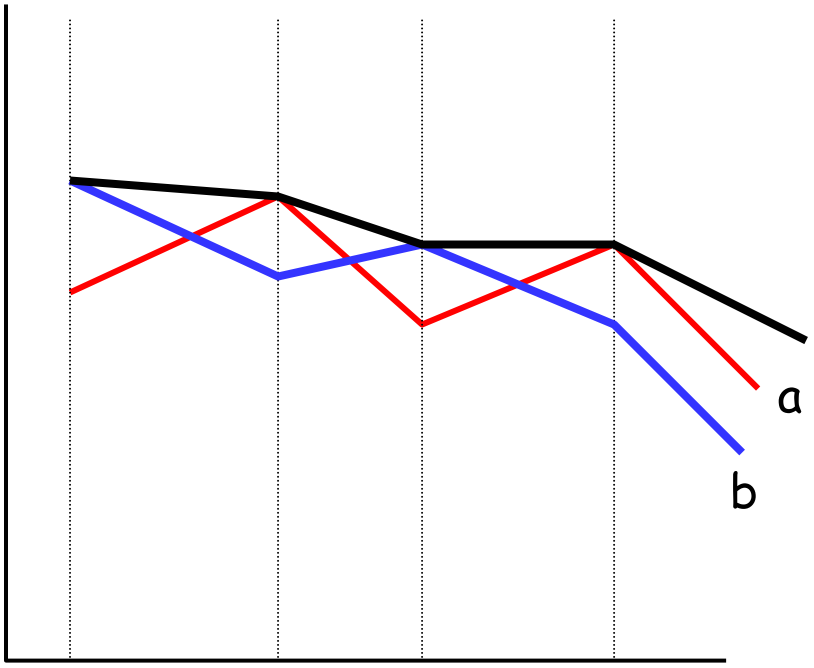
- Known that exact solutions requires too much workspace
- (ε,δ)-Approximation: return a random variable X such that

$$\Pr[|X-F_0| > \varepsilon F_0] < \delta$$

Max-Dominance Norm

Given k streams of m integers each, (the elements of the streams arrive in an arbitrary order), where
$1 \leq a_{i,j} \leq n$

$a_{1,1}\ a_{1,2}\ ..\ \ a_{1,m}$
$a_{2,1}\ a_{2,2}\ \dots\ a_{2,m}$
$\dots$
$a_{k,1}\ a_{k,2}\ \dots\ a_{k,m}$



Return

$\sum_{j=1}^{m} \max_{1 \leq i \leq k} a_{i,j}$

Cormode and Muthukrishnan, ESA 2003

Reduction From Max-Dominance Norm

- Input stream I, output stream O:

  F_0 of Output Stream = Dominance Norm of Input Stream

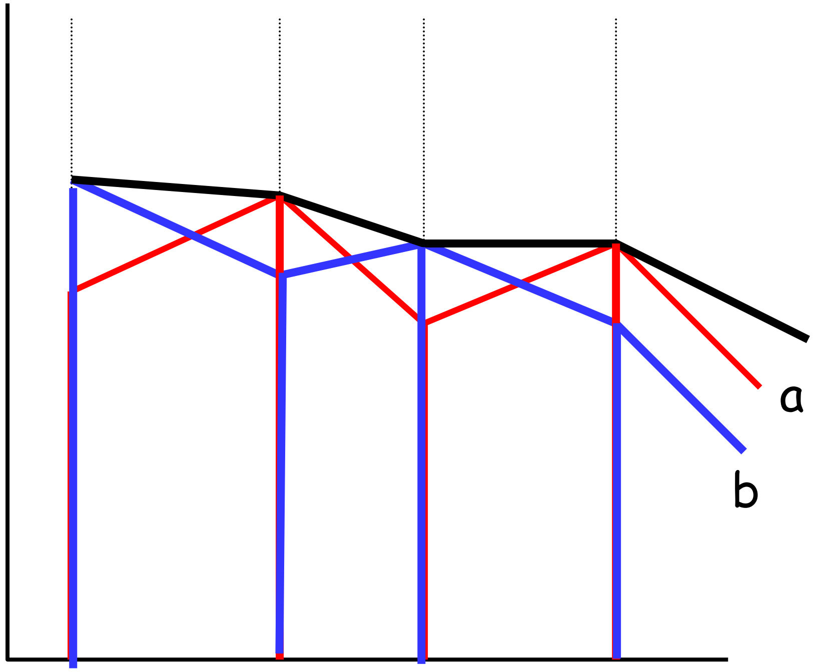
- Assign ranges to the k positions:

  [1,n] [n+1,2n] … [(k-1)n+1, kn]

- When element $a_{i,j}$ is received, generate the range

  $[(j-1)m+1, (j-1)m+1+a_{i,j}]$

- Observation: F_0 of the resulting stream of ranges is the dominance norm of the input stream



Reduction from Sensor Data Aggregation

- Problem: Compute aggregates over sensor observations
  - Sensors transmit "sketch" of data, instead of full data
  - Multi-path routing
  - Duplicate-insensitive sketches

- Duplicate-insensitive sum and average of sensor readings can be reduced to range-efficient F_0
  - Distinct Summation Problem
  - Considine et al. (2004) and Nath et al. (2004)

Counting Triangles in Graphs

- Problem:
  - Graph G=(V,E), where V={1..n}
  - Elements of E arrive as a stream $(i_1,j_1), (i_2,j_2)..$
  - Compute number of triangles in G
- Bar-Yossef et al. (SODA 2002) show a reduction to Range-efficient F_0 and F_2 on a stream of integers

Our Results

Input Stream

Sequence of ranges
$[l_1,r_1], [l_2,r_2] \ldots [l_m,r_m]$

for each i, $0 <= l_i <= r_i < n$, and l_i, r_i are integers

Output:

$| [l_1,r_1] \cup [l_2,r_2] \cup \ldots \cup [l_m,r_m]|$

- Randomized (ε,δ)-Approximation Algorithm for Range-efficient F_0 of a data stream
- Time complexity (n is the size of the universe):
  - Amortized processing time per interval: $O(\log(1/\delta) (\log (n/\varepsilon)))$
  - Time to answer a query for F_0: $O(\log 1/\delta)$
- Space Complexity: $O((1/\varepsilon^2)(\log(1/\delta)) (\log n))$

Comparison to Previous Work

	Previous Work	Our Results
Range-Efficient F_0	Bar-Yossef et al. (2002)	
	Time per item = $O(\log^5 n)(1/\varepsilon^5)(\log 1/\delta)$	Time per item = $O(\log n + \log 1/\varepsilon)(\log 1/\delta)$
	WorkSpace = $O(1/\varepsilon^3)(\log n)(\log 1/\delta)$	Workspace = $O(1/\varepsilon^2)(\log n)(\log 1/\delta)$
Max-Dominance Norms	Cormode, Muthukrishnan (2003)	
	Time per item= $O(1/\varepsilon^4)(\log n)(\log m)(\log 1/\delta)$	Time per item = $O(\log n + \log 1/\varepsilon)(\log 1/\delta)$
	Workspace = $O(1/\varepsilon^2)(\log n + 1/\varepsilon (\log m)(\log \log m))(\log 1/\delta)$	Workspace = $O(1/\varepsilon^2)(\log m + \log n)(\log 1/\delta)$

Related Work

- F_0 of a data stream
  - Flajolet-Martin (JCSS 1985)
  - Alon et al. (JCSS 1999)
  - Gibbons and Tirthapura (SPAA 2001)
  - Bar-Yossef et al. (RANDOM 2002)
  - Lower Bounds, Indyk-Woodruff (FOCS 2003)

- L_1 difference of data streams
  - Feigenbaum et al. (FOCS 1999)
  - Used range-summable hash functions

Algorithm

- Random Sampling

- Two Parts:
  - Adaptive Sampling
    - Change sampling probabilities dynamically
    - Gibbons and Tirthapura, SPAA 2001
  - Range Sampling
    - Quickly sample from a range of integers
    - Novel technical contribution

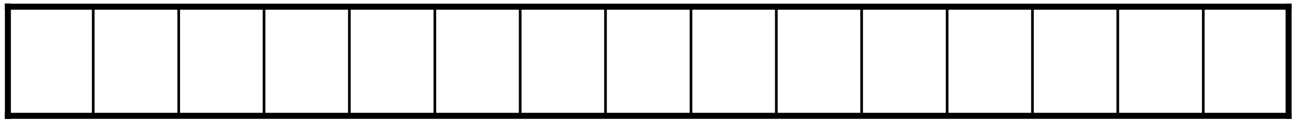
Adaptive Sampling for F_0

- Given a stream of numbers find the number of distinct elements in the stream

- Random Sampling Algorithm
  - Random Sample of distinct elements seen so far
  - Sampling Level i (sampling probability $=1/2^i$)
  - If sample size exceeds threshold, then sub-sample to a smaller probability

- Target Workspace = $O(1/\varepsilon^2)(\log 1/\delta)$ integers

Adaptive Sampling Example



Sample = {}, p = 1

Target Workspace = 4 numbers

Adaptive Sampling

5														

Sample = {5}, p = 1

Target Workspace = 4 numbers

Adaptive Sampling

5	3													

Sample = {5,3}, p = 1

Target Workspace = 4 numbers

Adaptive Sampling

5	3	7												

Sample = {5,3,7}, p = 1

Target Workspace = 4 numbers

Adaptive Sampling

5	3	7	5											

Sample = {5,3,7}, p = 1

Target Workspace = 4 numbers

Adaptive Sampling

5	3	7	5	6										

Sample = {5,3,7,6}, p = 1

Target Workspace = 4 numbers

Adaptive Sampling

5	3	7	5	6	8									

Sample = {5,3,7,6,8}, p = 1

Overflow, sub-sample

Sample = {3,6,8}, p = $\frac{1}{2}$

Target Workspace = 4 numbers

Adaptive Sampling

5	3	7	5	6	8	9								

Sample = {3,6,8,9}, p= $\frac{1}{2}$

Target Workspace = 4 numbers

Adaptive Sampling

5	3	7	5	6	8	9	7							

Sample = {3,6,8,9}, p= $\frac{1}{2}$

Target Workspace = 4 numbers

Adaptive Sampling

5	3	7	5	6	8	9	7	2						

Sample = {3,6,8,9,2}, p= $\frac{1}{2}$

Sample = {6,9}, p=$\frac{1}{4}$

Target Workspace = 4 numbers

Adaptive Sampling

5	3	7	5	6	8	9	7	2	2	7	8	8	3	5

Finally,

Sample = {6,9}, $p=\frac{1}{4}$

Return (Sample Size)(4) = 8

Adaptive Sampling for Range F_0

- Naïve:

  Given [x,y], successively insert
  $\{x, x+1, x+2, \ldots y\}$ into F_0 algorithm

- Problem: Time per range very large

Range Sampling – Time Efficient

Quickly determine how many elements in range [l,r] belong to the sample at current probability

Hash Functions

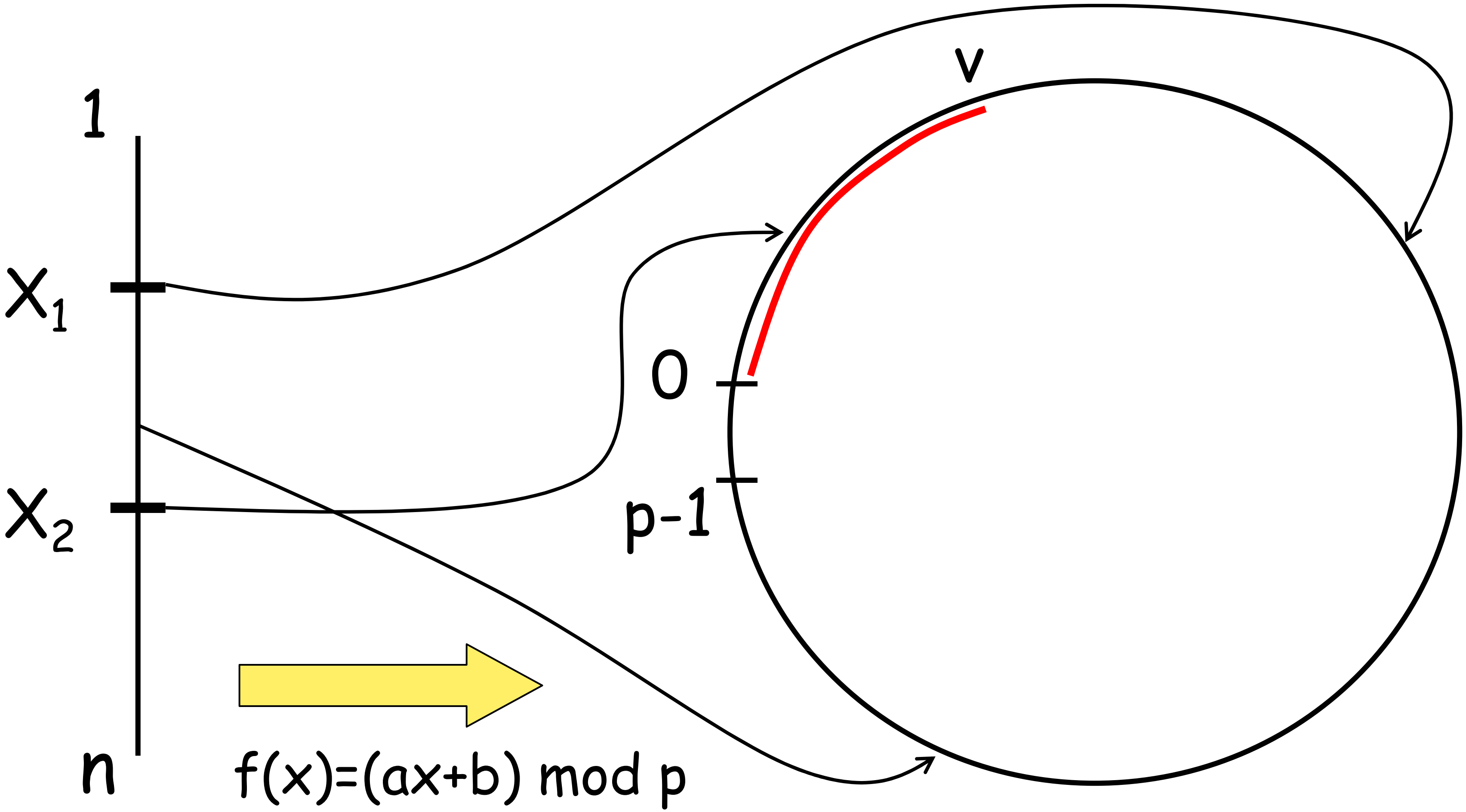
- Random sample through a hash function
  - Consistent decisions for same elements

- Our Hash Function:

  $h: \{1 \ldots n\} \rightarrow \{0,\ldots,p\text{-}1\}$, $h(x) = (ax+b) \bmod p$

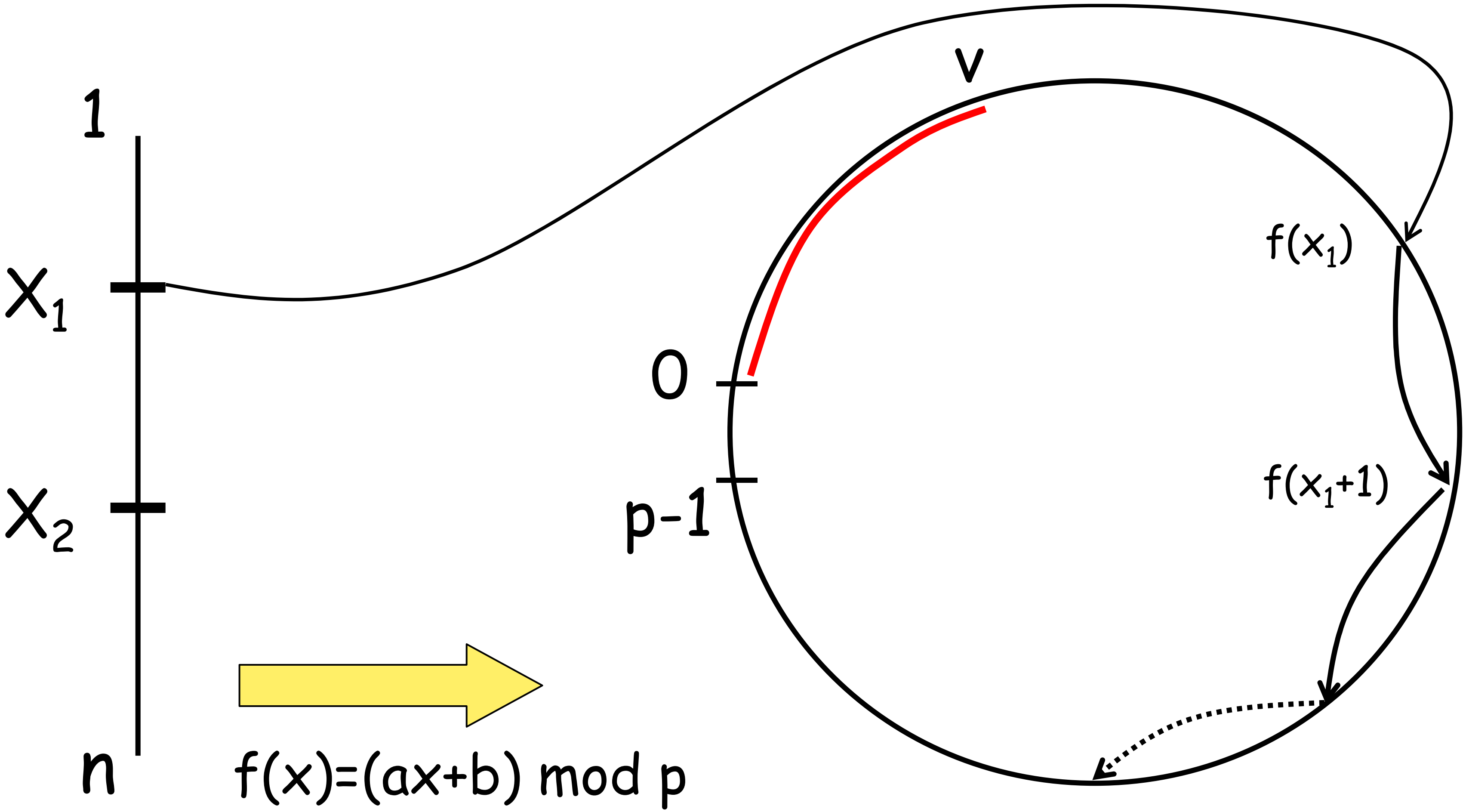
  - Integers a and b chosen randomly from $[0,p\text{-}1]$
  - Element x belongs in sample at level i if $h(x) \in \{0..v_i\}$ for some pre-determined v_i
  - For Range $[l,r]$, if for some $x \in [l,r]$ $h(x) \in \{0\ldots v_i\}$, then the range is "useful"

Range Sampling Problem



Compute $|\{x \in [x1,x2] : f(x) \in [0,v]\}|$

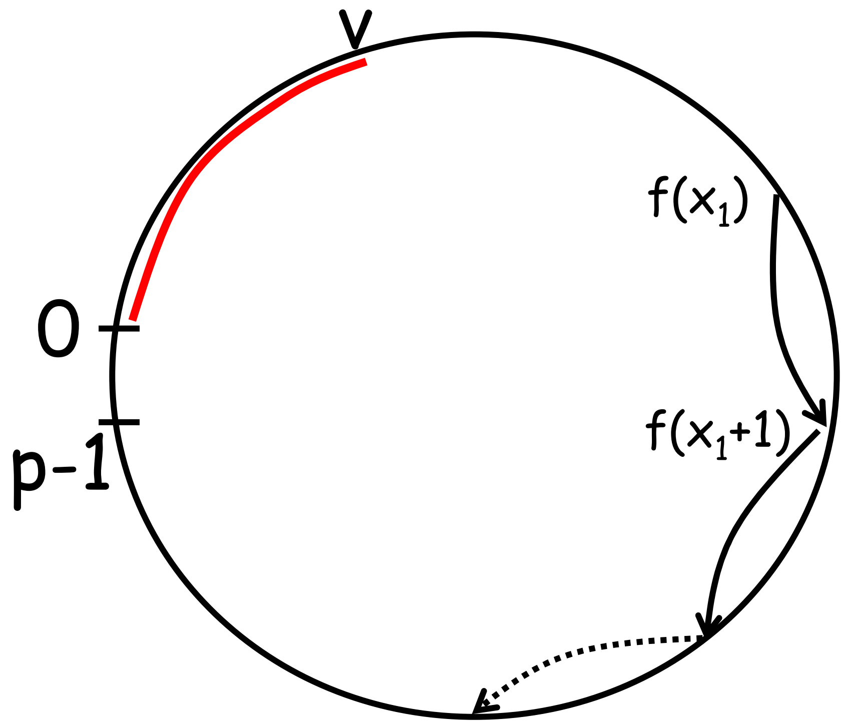
Arithmetic Progression



Common Difference = a

Low and High Revolutions

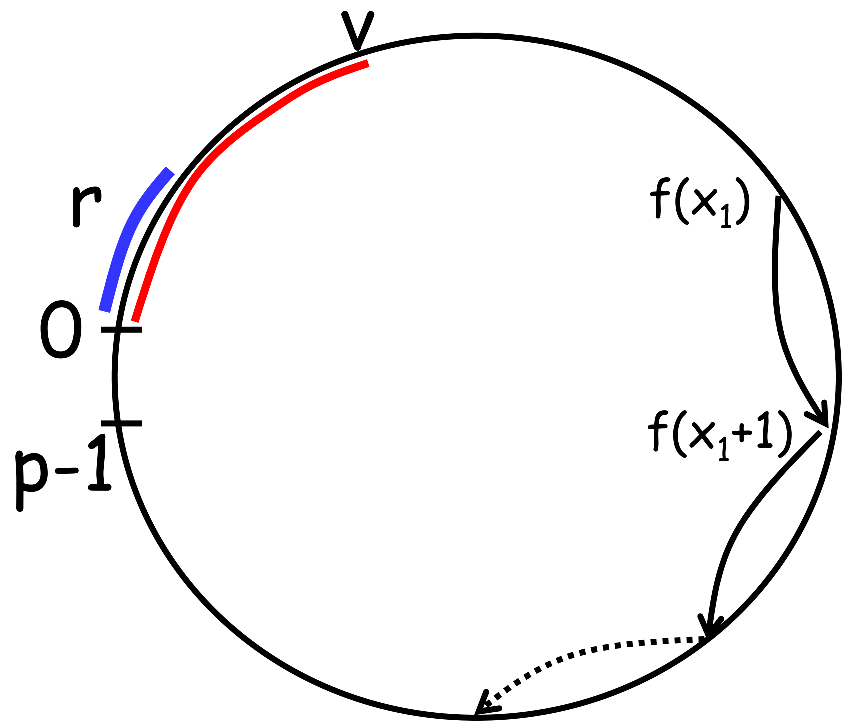
- Each revolution, number of hits on [0,v] is
  - v/a (low rev)
  - v/a +1 (high rev)



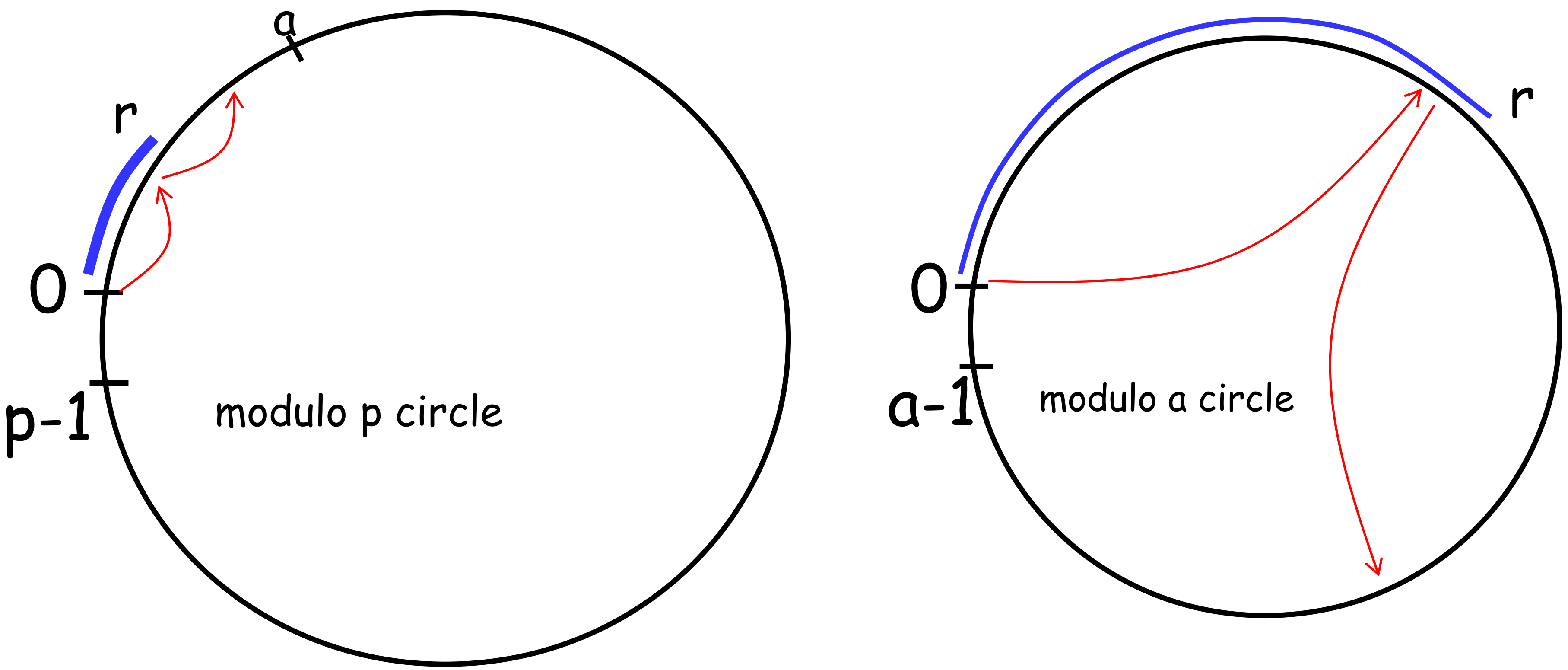
- Task: Count number of low, high revolutions

Starting Points of Revolutions

- Can find $r = (v - v \bmod a)$ such that:
  - If starting point in $[0,r]$, then high revolution
  - Else low revolution
- Task: Count the number of revolutions with starting point in $[0,r]$



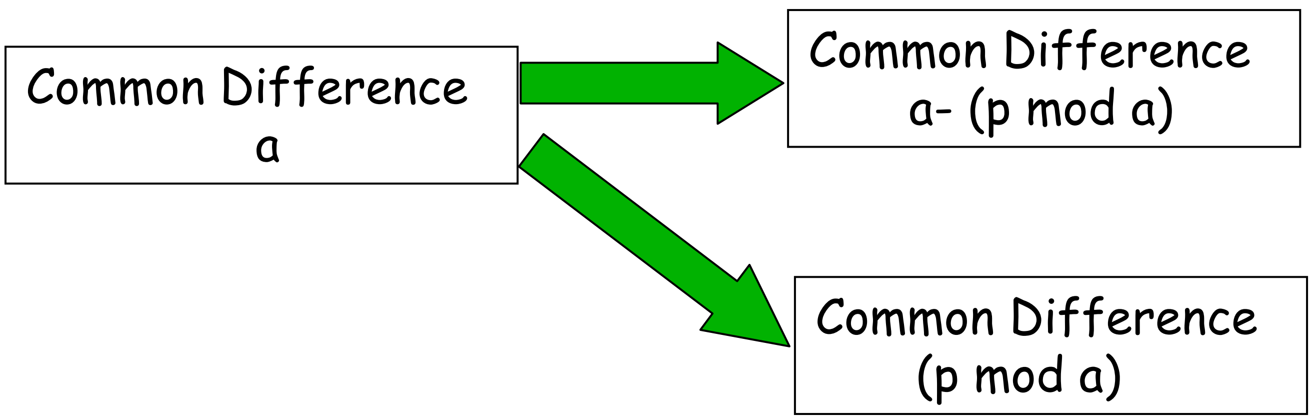
Recursive Algorithm



Observation: Starting Points form an Arithmetic Progression with difference (- p mod a)

Recursive Algorithm

- Focus on common difference
- Two Reductions Possible



At least one of the two common differences is smaller than a/2

Range Sampling

Range [x,y]:

- Time Complexity: O(log (y-x))
- Space Complexity: O(log (y-x) + log m)
- Plug back into adaptive sampling to get range-efficient F_0 algorithm

Extensions

- Distributed Streams
  - Each stream observed by different party
  - Party sends a “sketch” to a referee
  - Estimate F_0 over the union streams, using the sketches
- Multi-dimensional ranges
- Sliding Windows

Open Problems

- Simple Range-Efficient Algorithms for F_k $(k > 1)$?
- Time Lower Bounds for Range-Efficient F_0