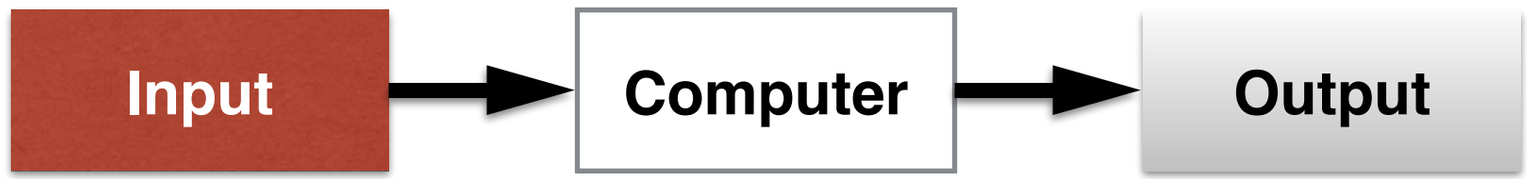
Streaming algorithms

(an introduction)

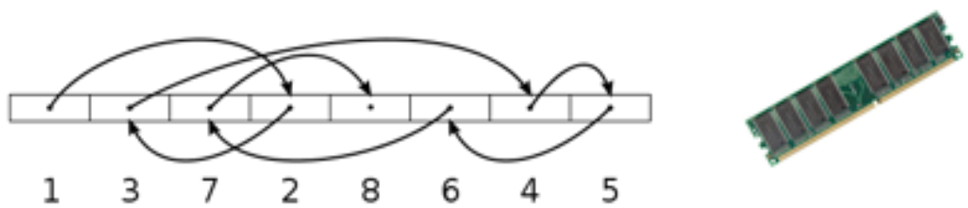
Frédéric Magniez

LIAFA, CNRS, Université Paris Diderot

Data access model

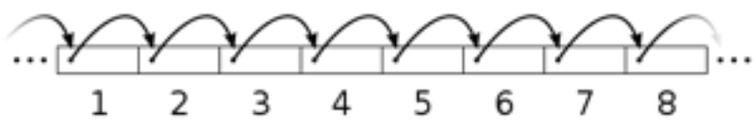


- Random access



  - Input is loaded into the internal memory

- Sequential access



  - Input: data stream that cannot fit into internal memory

Streaming algorithms

Context

- Single pass
  - Network monitoring
  - Online with memory constraints
- Few passes
  - Genome decoding
  - Web databases

Streaming algorithms

- Input
  - Data stream
- Algorithm
  - Number of passes: one, constant or log
  - Internal memory size: polylog, sublinear
  - Processing time per symbol: polylog

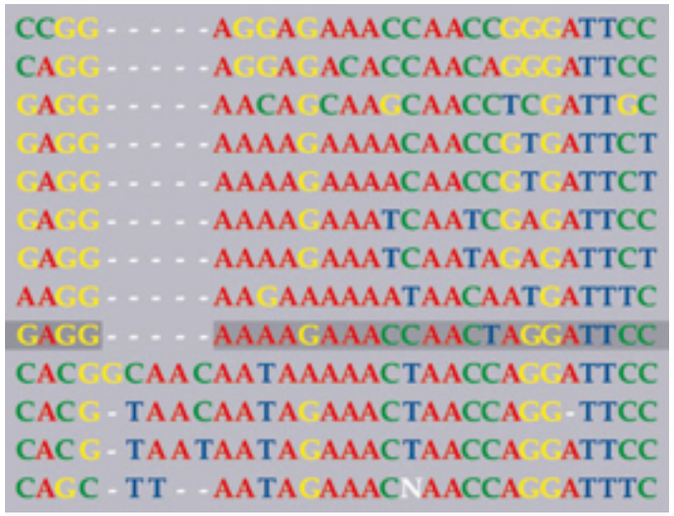
sublinear space
polylog time

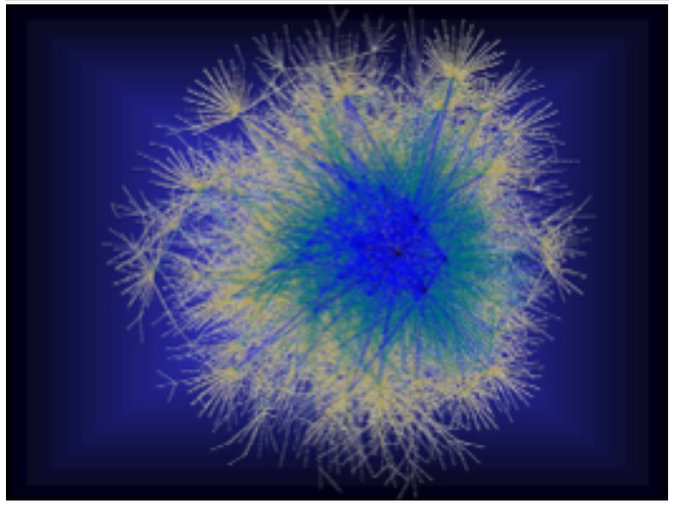
First example

- Missing number
  - Input stream: n numbers in $\{1,2,\ldots,n+1\}$, all pairwise distinct
  - Output: the only missing number among $\{1,2,\ldots,n+1\}$
- Algorithm
  - Deterministic, 1 pass, memory $O(\log n)$

Next examples

- Statistics
  - Frequency moments
- Strings
  - Pattern matching
  - Formal languages
- Graphs
  - Maximum cardinality matching





Statistics

Frequency moments

- Problem
  - Input stream: $a_1, a_2, \ldots, a_n$ in $\{1,2,\ldots,m\}$
  - Output: $F_k = (f_1)^k + (f_2)^k + \ldots + (f_m)^k$ where f_j=multiplicity of j
- Special cases
  - F_0 = # of distinct values, $F_1 = n$, $F_\infty = \max f_j$
  - F_2 = surprise index

F_1: Loglog Counter

- Theorem [Moris'78]
  - 1-pass randomized algorithm with $O(\log\log n)$ memory
- Key idea
  - Boolean random variable X
  - T = # of samples of X in order to get value 1
  - Then $E(T) = 1/\Pr(X=1)$

Algorithm for F_1

```
a := 0
While stream is not empty
    With probability 2^-a
        a := a + 1
Output 2^a-1
```

- Analysis
  - $E(2^a) = n+1$
  - $Var(2^a) = O((E(2^a))^2)$
- Conclusion
  - 1-pass, memory $O((\log\log n) \times \log(1/\delta)/\varepsilon^2)$ s.t.

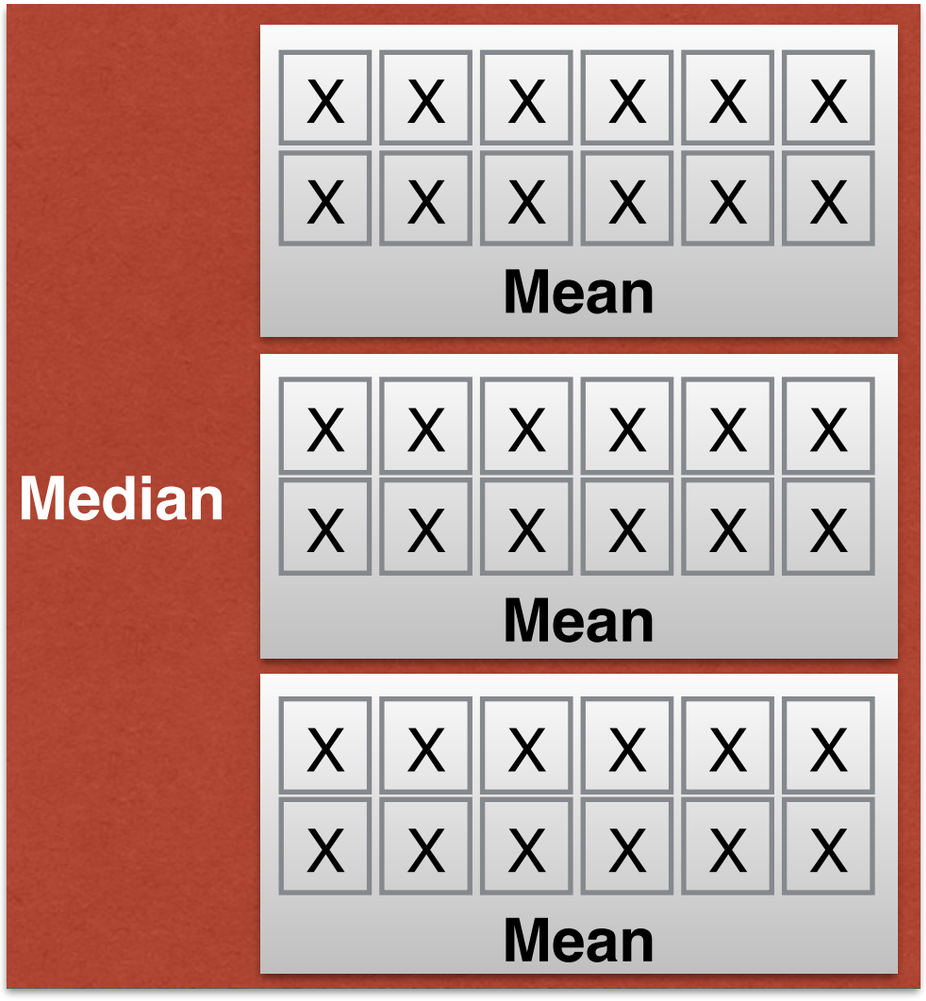
    $Pr(|Output - F_1| > \varepsilon F_1) \leq \delta$

Estimator

- Theorem
  - Variable X s.t. $Var(X) = O(E(X)^2)$
  - Combining $O(\log(1/\delta)/\varepsilon^2)$ samples gives an (ε,δ)-estimator
- Construction
  - Amplifying the precision

    Mean of $O(1/\varepsilon^2)$ samples
  - Amplifying the confidence

    Median of $O(\log(1/\delta))$ means



F_0: Log estimator

- Theorem [Flajolet-Martin'83]
  - 1-pass randomized algorithm with $O(\log m)$ memory
- Key idea
  - Independent random variables $a_1, a_2, \ldots, a_l$ in $\{1,2,\ldots,m\}$
  - $T = \min(a_i)$
  - Then $T \approx l/m$ w.h.p.

Random hash functions

- Our context
  - Values $a_1, a_2, \ldots, a_n$ are fixed (and not random)
  - They takes $l = F_1$ different values

    Say $a_1, a_2, \ldots, a_l$
- Main tool: Random hash function
  - For a random function $h: \{1,2,\ldots,m\} \rightarrow \{1,2,\ldots,m\}$

    Values $h(a_1), h(a_2), \ldots, h(a_l)$ are random and independent

Algorithm for F_0

```
Take random function h:{1,2,…,m}→{1,2,…,m}
min := 0
While stream is not empty
    min := min(h(next element))
Output m/min
```

- Analysis
  - $h(a_1), h(a_2), \ldots, h(a_l)$ are random and independent
  - Therefore $\min \approx l/m$ (and $\text{output} \approx l$) w.h.p.
- Issue
  - Storing h requires $(m \log m)$ bits

k-wise independent hash functions

- Full independency is not required
  - Need $h(a_1),h(a_2),\ldots,h(a_l)$ to be constant-wise independent
  - Even an approximate notion would be enough
- Theorem
  - There exists a family H of functions s.t.

    Generating & storing h from H requires $O(k \log m)$ bits

    Values $h(a_1),h(a_2),\ldots,h(a_l)$ are k-wise independent

F_2

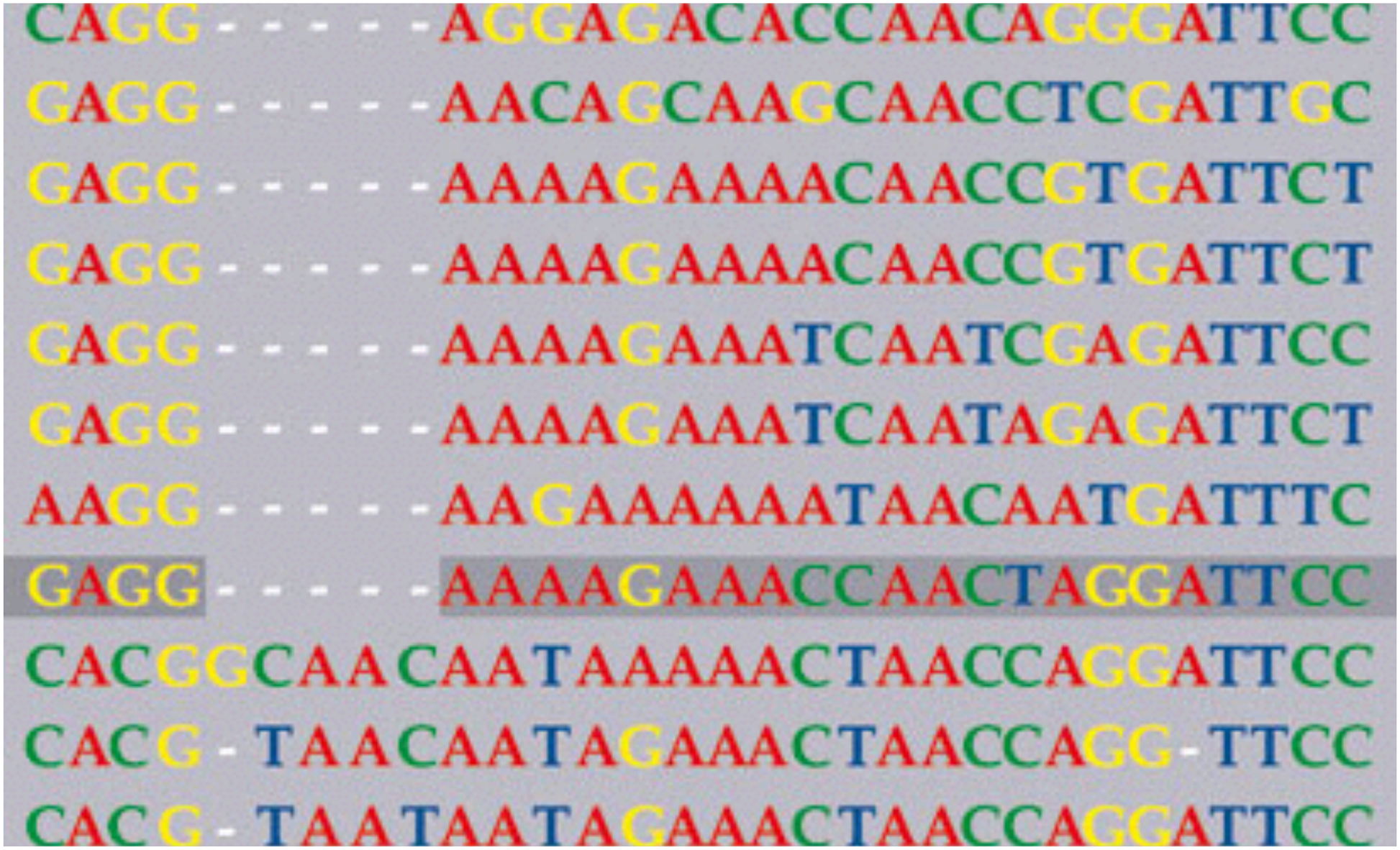
- Theorem [Alon-Matias-Szegedy'96: Godel prize'05]
  - 1-pass randomized algorithm with $O(\log(mn))$ memory
- Key idea
  - Take a random function $h:\{1,2,\ldots,m\}\rightarrow\{-1,1\}$
  - Let $X = h(a_1)+h(a_2)+\ldots+h(a_n) = f_1h(1) + f_2h(2) + \ldots + f_mh(m)$
  - Then $E(X^2) = F_2$

    $E(h(i)^2)=1$ and $E(h(i)h(j))=0$, when $i\neq j$

F_k: conclusion

- Problem
  - Input stream: $a_1, a_2, \ldots, a_n$ in $\{1,2,\ldots,m\}$
  - Output: $F_k = (f_1)^k + (f_2)^k + \ldots + (f_m)^k$ where f_j=multiplicity of j
- Results
  - F_0: $\Theta(\log m)$ space
  - F_1: $\Theta(\log\log n)$ space
  - F_2: $\Theta(\log(mn))$ space
  - F_k, $k \geq 3$: $\Theta(m^{1-(2/k)})$ space
  - F_∞: $\Theta(m)$ space

also valid when items can be deleted (Turnstile model)


CAGG-----AGGAGACACCAACAGGGATTCC
GAGG-----AACAGCAAGCAACCTCGATTGC
GAGG-----AAAAGAAAACAACCGTGATTCT
GAGG-----AAAAGAAAACAACCGTGATTCT
GAGG-----AAAAGAAATCAATCGAGATTCC
GAGG-----AAAAGAAATCAATAGAGATTCT
AAGG-----AAGAAAAAATAACAATGATTTC
GAGG-----AAAAGAAACCAACTAGGATTCC
CACGGCAACAATAAAAACTAACCAGGATTCC
CACG-TAACAATAGAAACTAACCAGG-TTCC
CACG-TAATAATAGAAACTAACCAGGATTCC

Strings

Equality testing

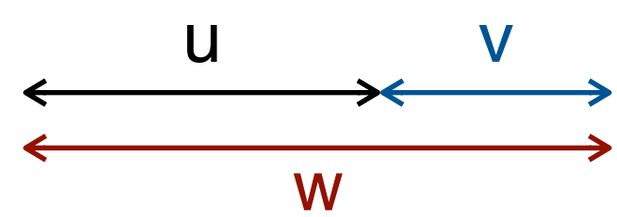
- Problem
  - Input stream: n-bit string u followed by n-bit string v
  - Output: Decide if $u=v$
- Key tool: Fingerprint
  - Let $F(u,X)=u_1X^{n-1}+u_2X^{n-2}+\ldots+u_n$ and $F(v,X)=\ldots$
  - Let $q \in [n^3,2n^3]$ be a prime number
  - Then for a random $a \in \{0,1,\ldots,q-1\}$

    If $u=v$, $F(u,a)=F(v,a) \bmod q$ always

    If $u\neq v$, $\Pr_a(F(u,a)=F(v,a) \bmod q) \leq 1/n^2$

Property of Fingerprint

- Memory space
  - $O(\log(n^3)) = O(\log n)$
- Linearity
  - Assume $w=uv$



    Given 2 fingerprints, one can deduce the 3rd one

    using $O(\log n)$ arithmetic operations modulo q

- Application
  - Can be computed in 1-pass and memory $O(\log n)$

    even if letters arrive in arbitrarily order

Fingerprint in streaming

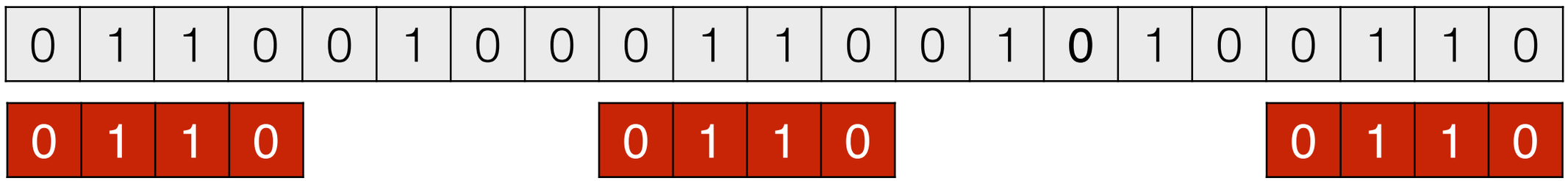
```
Fu := 0 and Fv := 0
Find a prime q ∈ [n³,2n³]
Take at random a ∈ {0,1,…,q-1}
While stream u is not empty
    Fu := a × Fu + (next bit) mod q
While stream v is not empty
    Fv := a × Fv + (next bit) mod q
Output (Fu=Fv)
```

- Analysis
  - Memory space: $O(\log n)$
  - One sided error

    If $u=v$, output is always correct

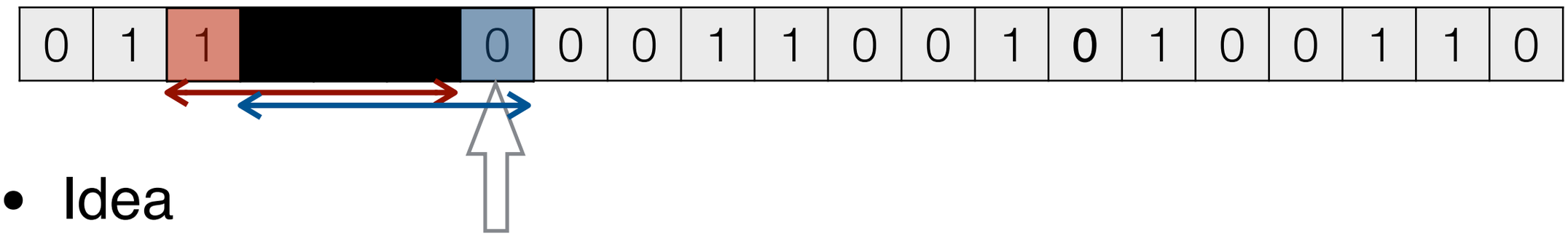
    If $u \neq v$, output is incorrect with probability $\leq 1/n^2$

Pattern matching



- Problem
  - Input stream: m-bit pattern p followed by n-bit text t
  - Output: positions i where p appears in t
- Deterministic approach
  - [Knuth-Morris-Pratt'75]: memory O(m)
- Fingerprint approach
  - [Porat-Porat'09]: memory O(log(n)log(m))

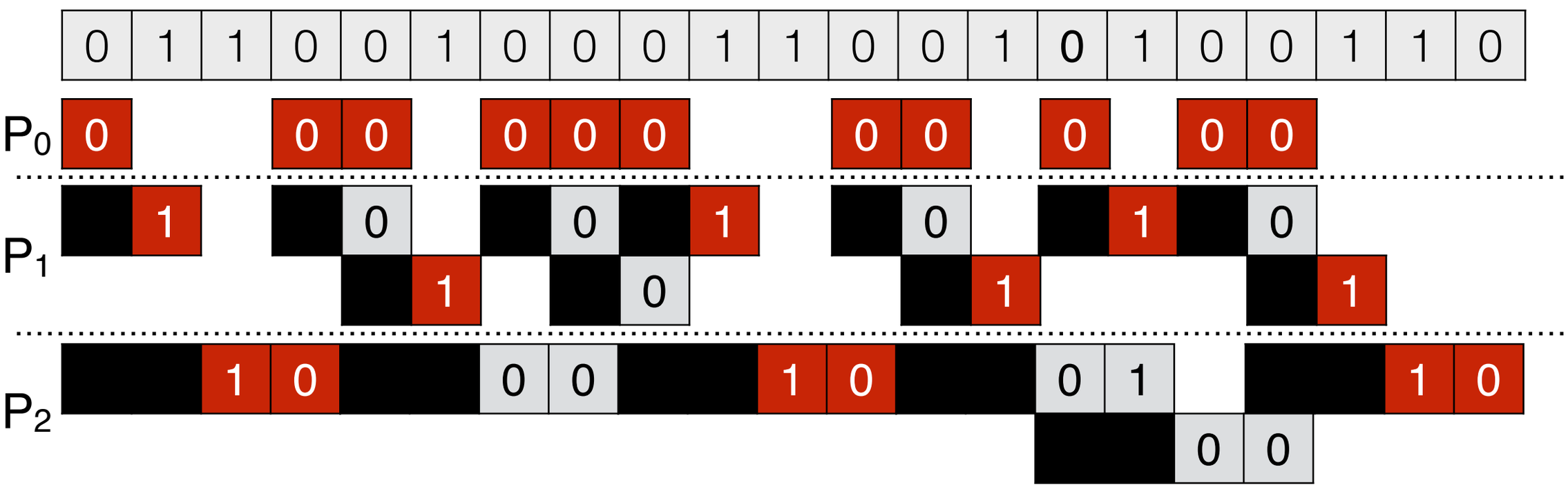
Fingerprint approach



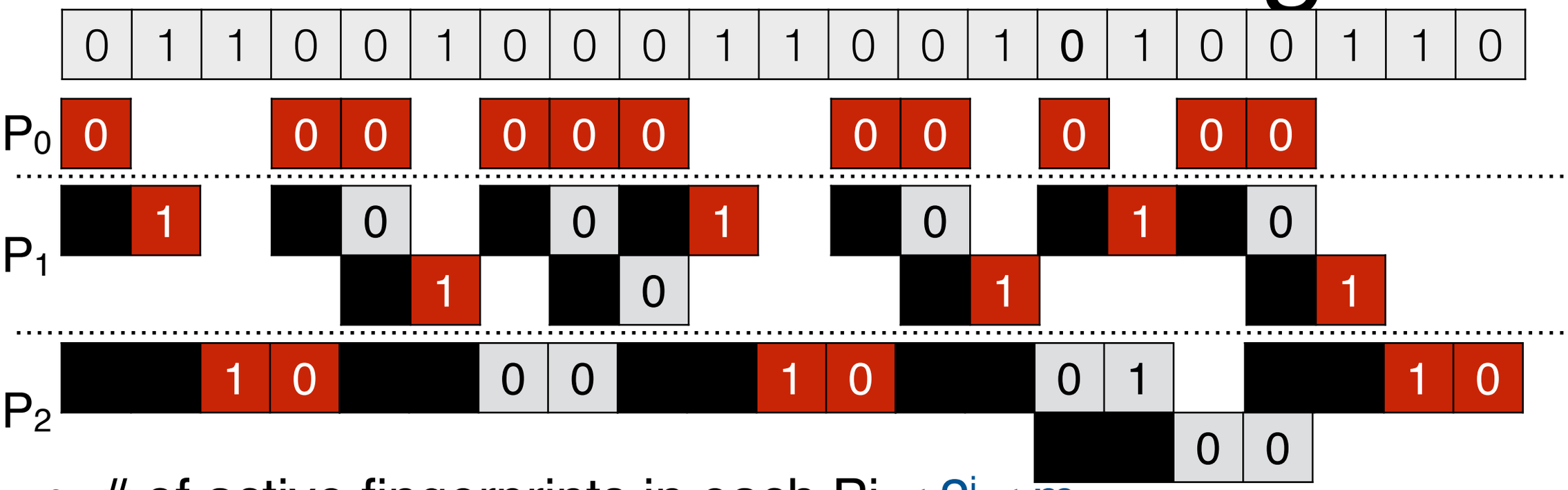
- Idea
  - Compute the fingerprint of pattern p
  - Compare it with fingerprint of each portion t[i,i+m-1]
- Updating the fingerprints while reading t[i+m]
  - Two consecutive fingerprints can be deduced from each other given **t[i]** and t[i+m]
  - Issue: require memory O(m) to remember t[i] since also need to remember t[i+1],…,t[i+m-1]

Recursive matching

- Processus $P_0, P_1, \ldots, P_{\log m}$
  - P_i finds patterns $p[1,2^i]$ and transmit them to P_{i+1}
  - F_i = fingerprint of $p[1,2^i]$
- Example



Recursive matching



- # of active fingerprints in each Pi $\leq 2^i \leq m$
  - At most 2, when $p[1,2^{i-1}]$ is aperiodic
  - Always regularly placed: every period($p[1,2^{i-1}]$) letters
- Each Pi uses memory O(log n)
  - For the first two active fingerprints
  - And the total number of pending fingerprints

0010010010010010010010010
↔ 0010010010
↔ 0010010010
↔ 0010010010
↔ 0010010010

Other applications

- Recognizing well-parenthesized expressions
  - [M-Mathieu-Nayak'10]

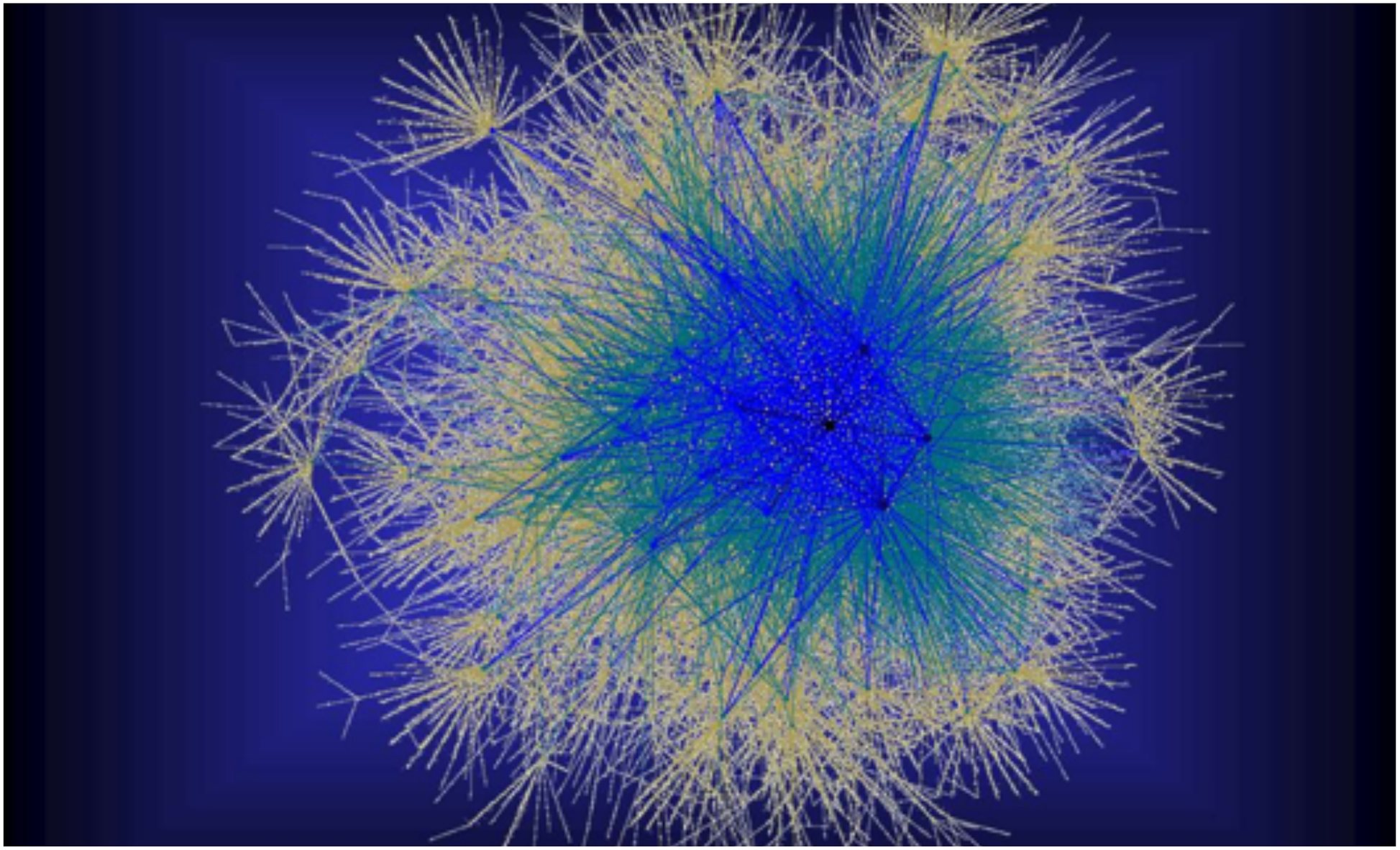
    1 pass: memory space $O(\sqrt{(n \log n)})$

    2 passes (one in each direction): memory space $O((\log n)^2)$

- Checking insert/extract sequences of priority queue
  - Similar results

    1 pass: [Chakrabarti-Cormode-Kondapally-McGregor'10]

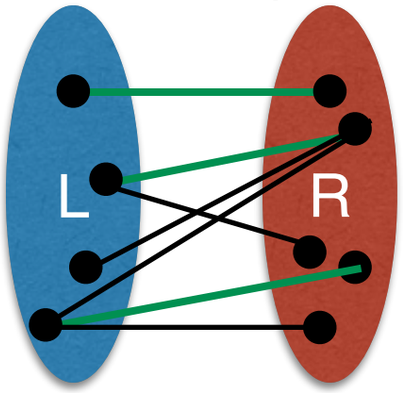
    2 passes: [François-M'13]



Graphs

Graph matching

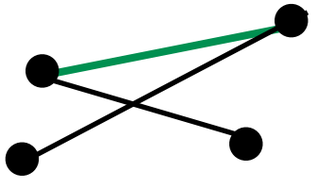
- Problem
  - Input stream: edges of a bipartite graph (with n vertices)
  - Output: a maximum size matching



  - Memory space ≤ n polylog n
- Greedy approach (1 pass)
  - Provides a maximal (for inclusion) matching M…

    …therefore a 0.5-approximation

$|M| \geq 0.5\ |OPT|$ (OPT: maximum size matching)



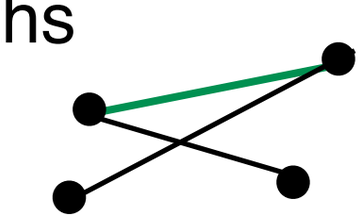
Multiple passes

- Theorem
  - $(1-\varepsilon)$-approximation within $f(\varepsilon)$ passes [McGregor'05]

    $f(\varepsilon) \leq \log\log(1/\varepsilon)/\varepsilon^2$ [Ahn-Guha'11]

- Idea

  At each pass, find larger and larger augmenting paths



Example for 3 passes

Pass 1

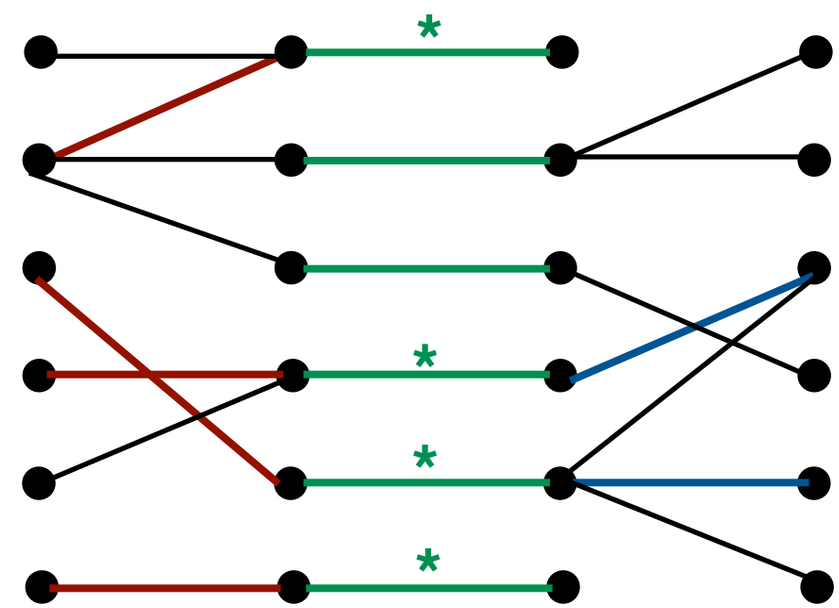
- Compute a maximal matching M between A and B

Pass 2

- Compute a maximal matching M_L between B\M(B) and M(A)
- Call M* the edges of M that intersect M_L

Pass 3

- Compute a maximal matching M_R between A\M(A) and M*(B)
- Augment M using M_L and M_R



Key lemma

If |M| ≤ (0.5+ε) |OPT| then M has ≥ (0.5+ε) |OPT| 3-augmentable edges

Random edge arrivals

- Model
  - Graph is fixed but edges arrive in uniform random order
- Theorem [Konrad-M-Mathieu'12]
  - There is an 0.519-approximation algorithm

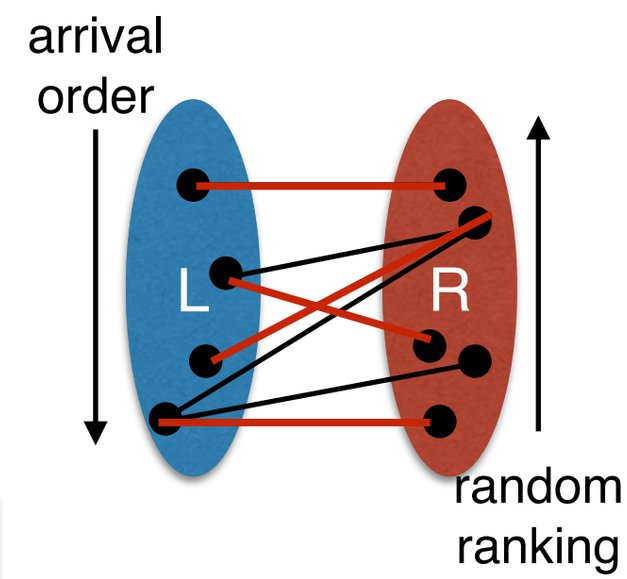
    Idea: Divide the stream in 3 parts. Simulate each pass of the 3-pass algorithm on one part

Compute matching M	Find left wings	Find right wings

1 $|E|/3$ $2|E|/3$ $|E|$

Key lemma: If $|M_{Greedy}| \sim 0.5|OPT|$ for most of random orders, then $|M_{Greedy}|$ converges quickly

Vertex arrival order



- Restriction to specific edge order
  - Each left-node arrives with all its adjacent edges
- Online
  - [Karp-Vazirani-Vazirani'90]

    RANKING algorithm gives an (1-1/e)-approximation (optimal)

    0.63
  - [Devanur-Jain-Kleinberg'13]

    New Primal-Dual analysis
- Streaming
  - [Krapalov'13]: also optimal in streaming
  - [Goel-Krapalov-Khanna'12]: achieved by a **deterministic** pass

Conclusion

Conclusion

- Streaming algorithms for many problems with many tools
  - Randomization is central (often)
  - Tools: sketch, sparsification…
- Many extensions
  - Streams with annotations, sorting procedures…
  - Auxiliary Read-Write streams
  - Distributed streaming algorithms
- Lower bounds can be proven (sometimes)