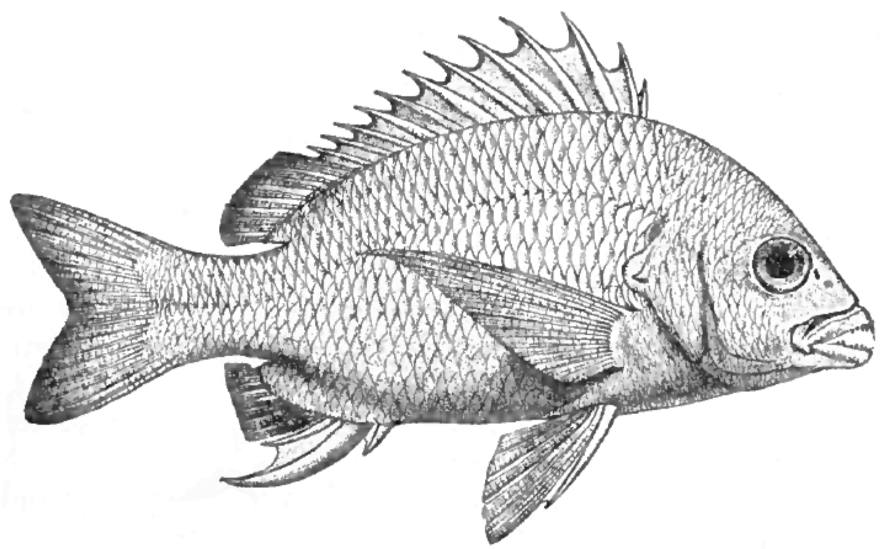
Lower Bounds for Quantile Estimation in Random-Order and Multi-Pass Streams

Sudipto Guha (UPenn)   Andrew McGregor (UCSD)



Data Stream Model

Data Stream Model

- *Stream:* m elements from a universe of size n:

  3,5,3,7,5,4,8,5,3,7,5,4,8,6,3,2,6,4,7,3,4, ...

  e.g., IP packets, search engine queries, data read from external memory, device, sensor readings...

Data Stream Model

- *Stream:* m elements from a universe of size n:

  3,5,3,7,5,4,8,5,3,7,5,4,8,6,3,2,6,4,7,3,4, ...

  e.g., IP packets, search engine queries, data read from external memory, device, sensor readings...

- *Data-Stream Model:*

  No control over the ordering of elements
  Limited working memory S
  Limited time to process each element

[Morris '78] [Munro, Paterson '78] [Flajolet, Martin '85] [Alon, Matias, Szegedy '96]
[Henzinger, Raghavan, Rajagopalan '98] [Feigenbaum, Kannan, Strauss, Viswanathan '99]

Data Stream Model

- *Stream:* *m* elements from a universe of size *n*:

    3,5,3,7,5,4,8,5,3,7,5,4,8,6,3,2,6,4,7,3,4, ...

  e.g., IP packets, search engine queries, data read from external memory, device, sensor readings...

- *Data-Stream Model:*

    No control over the ordering of elements
    Limited working memory *S*
    Limited time to process each element

  [Morris '78] [Munro, Paterson '78] [Flajolet, Martin '85] [Alon, Matias, Szegedy '96] [Henzinger, Raghavan, Rajagopalan '98] [Feigenbaum, Kannan, Strauss, Viswanathan '99]

- *Previous work:* quantiles, frequency moments, histograms, clustering, entropy, graph problems...

Stream Order?

Stream Order?

- Almost all prior research considers *adversarial-order model (AOM)*.

Stream Order?

- Almost all prior research considers *adversarial-order model (AOM)*.
- What about the *random-order model (ROM)*?
  Form of average case analysis
  Stream of independent samples
  Uncorrelated fields in a database...

Stream Order?

- Almost all prior research considers *adversarial-order model (AOM)*.
- What about the *random-order model (ROM)*?
  - Form of average case analysis
  - Stream of independent samples
  - Uncorrelated fields in a database...
- *Previous Work:*
  - Frequent elements [Demaine, Lopez-Ortiz, Munro '02]
  - Entropy & Distances [Guha, McGregor, Venkatasubramanian '06]
  - Histograms [Guha, McGregor '07]
  - Quantiles... [Munro, Paterson '78], [Guha, McGregor '06]

Quantile Estimation

Quantile Estimation

- Given a set of m elements, a *t-approx median* is any element of rank = $m/2 \pm t$

Quantile Estimation

- Given a set of m elements, a *t-approx median* is any element of rank = $m/2 \pm t$
- *Previous Work:*
  AOM: εm-approx in $O(\varepsilon^{-1} \lg \varepsilon m)$ space
  [Greenwald, Khanna '01], [Shrivastava, Buragohain, Agrawal, Suri '04]
  [Cormode, Korn, Muthukrishnan, Srivastava '06]

Quantile Estimation

- Given a set of m elements, a *t-approx median* is any element of rank = $m/2 \pm t$

- *Previous Work:*
  AOM: εm-approx in $O(\varepsilon^{-1} \lg \varepsilon m)$ space
  [Greenwald, Khanna '01], [Shrivastava, Buragohain, Agrawal, Suri '04]
  [Cormode, Korn, Muthukrishnan, Srivastava '06]

  ROM: 1-pass exact selection in $O(m^{1/2})$ space
  [Munro, Paterson '78]

  1-pass $m^{1/2+\varepsilon}$-approx in $O(2^{1/\varepsilon} \text{ polylog } m)$ space
  $O(\lg \lg m)$-pass selection in $O(\text{polylog } m)$ space
  [Guha, McGregor '06]

Quantile Estimation

- Given a set of m elements, a *t-approx median* is any element of rank = $m/2 \pm t$

- *Previous Work:*
  AOM: εm-approx in $O(\varepsilon^{-1} \lg \varepsilon m)$ space
  [Greenwald, Khanna '01], [Shrivastava, Buragohain, Agrawal, Suri '04]
  [Cormode, Korn, Muthukrishnan, Srivastava '06]

  ROM: 1-pass exact selection in $O(m^{1/2})$ space
  [Munro, Paterson '78]

  1-pass $m^{1/2+\varepsilon}$-approx in $O(2^{1/\varepsilon} \text{ polylog } m)$ space
  $O(\lg \lg m)$-pass selection in $O(\text{polylog } m)$ space
  [Guha, McGregor '06]

- *Main Questions:*
  Are these ROM results possible in the AOM model?
  Can these ROM results be improved?

Results

- *Thm:* For a stream in *random order*:
  a) 1-pass, O(polylg m)-space, $\tilde{O}(m^{1/2})$-approx
  b) O(lg lg m)-pass, O(polylg m)-space exact selection

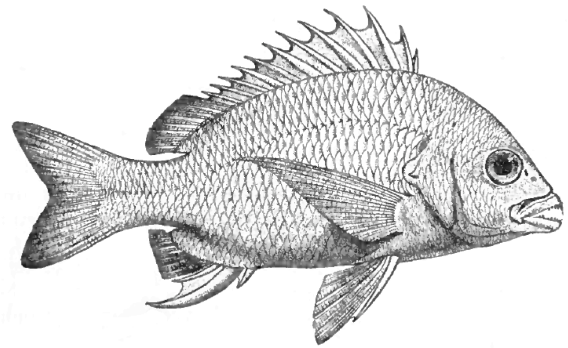
- *Thm:* For a stream in *adversarial order*:
  a) 1-pass, $\tilde{O}(m^{1/2})$-approx requires $\Omega(m^{1/2})$ space
  b) O(polylg m)-space exact requires $\Omega(\lg m)$ passes

- *Bonus Thm:* For a stream in *random order*, a single pass, t-approx requires $\Omega(m^{1/2} t^{-3/2})$ space.

1: Algorithm *(Random)*

2: Lower-Bound *(Random)*

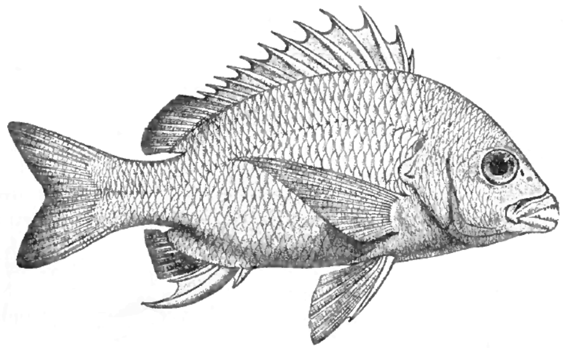
3: Lower-Bound *(Advesarial)*



1: Algorithm *(Random)*

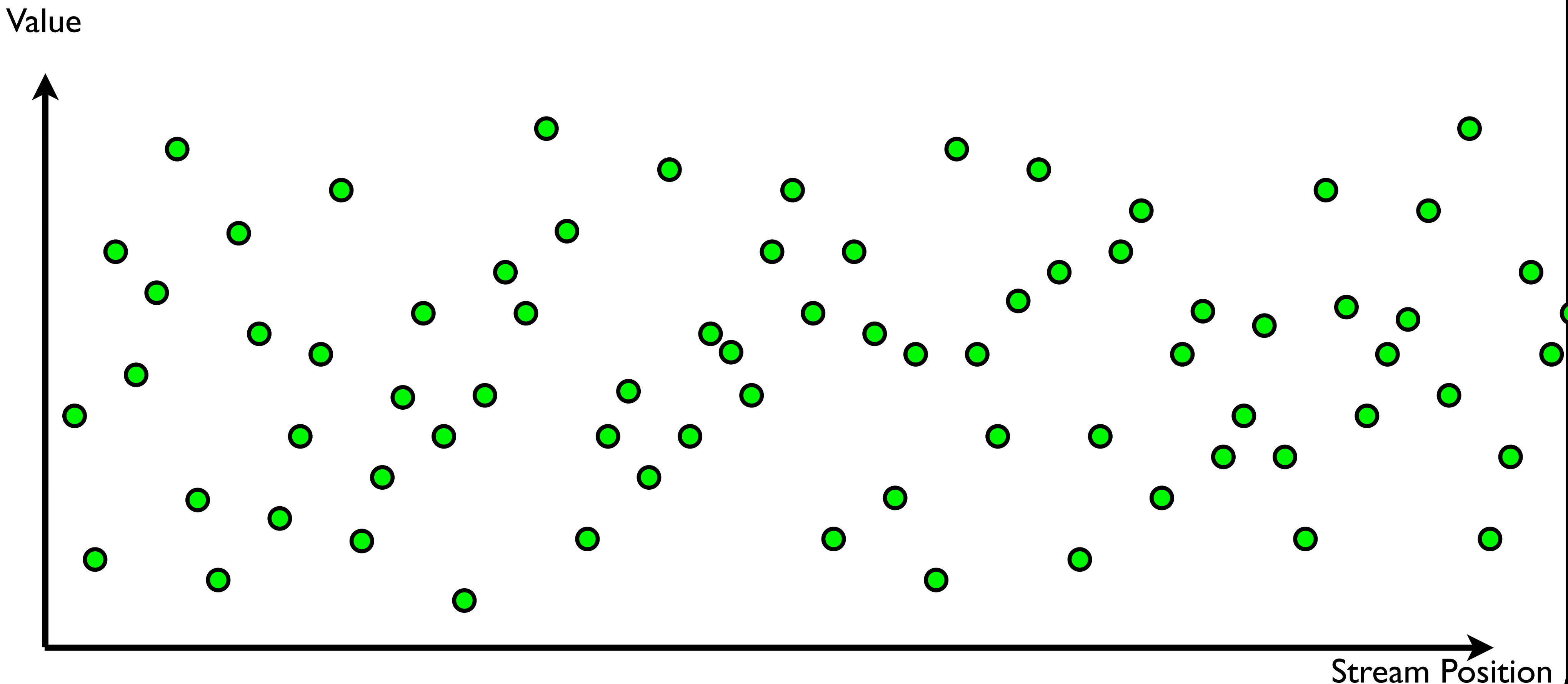
2: Lower-Bound *(Random)*

3: Lower-Bound *(Advesarial)*



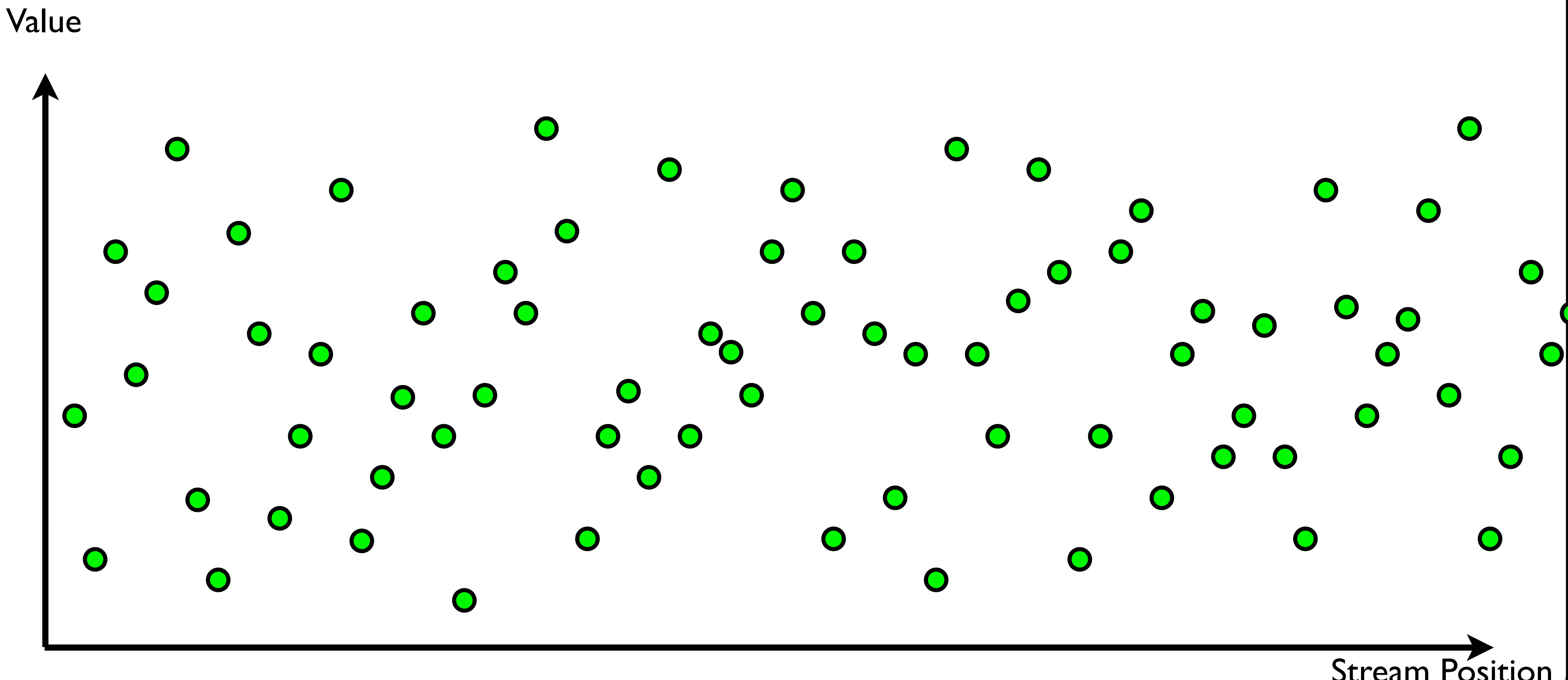
Algorithm

Algorithm



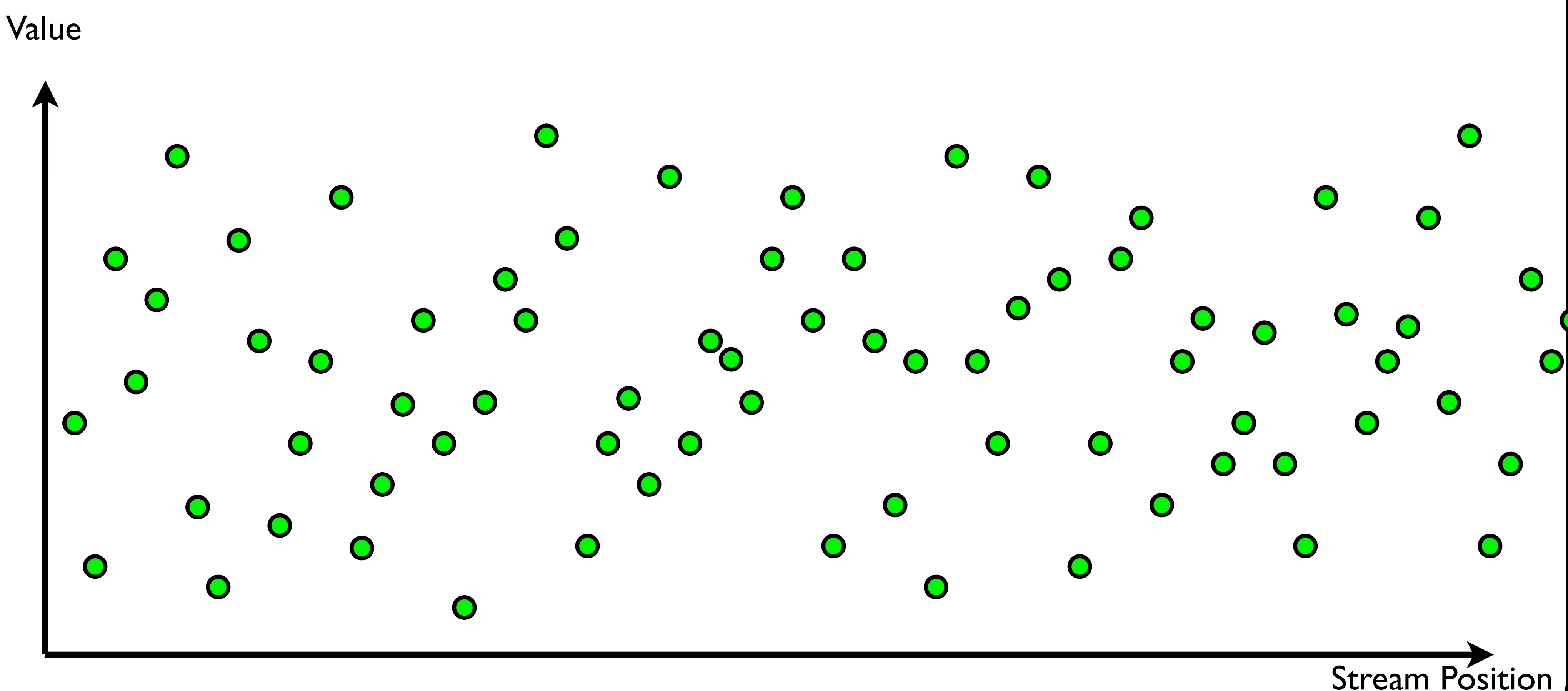
Algorithm

1) Maintain bounds [*a*,*b*] for median and *c* in [*a*,*b*]



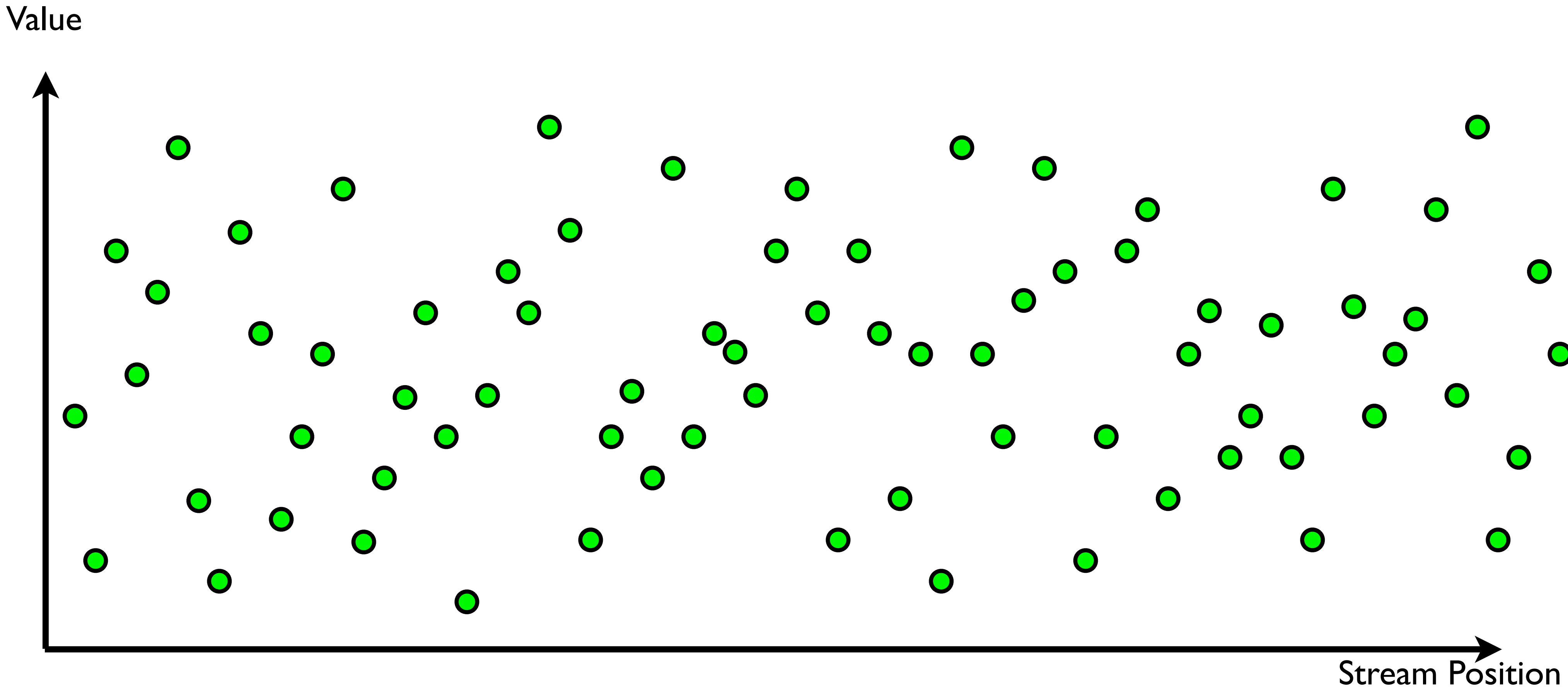
Algorithm

1) Maintain bounds [a,b] for median and c in [a,b]
2) Split stream in segments: $S_1, E_1, S_2, E_2, \ldots, S_p, E_p$



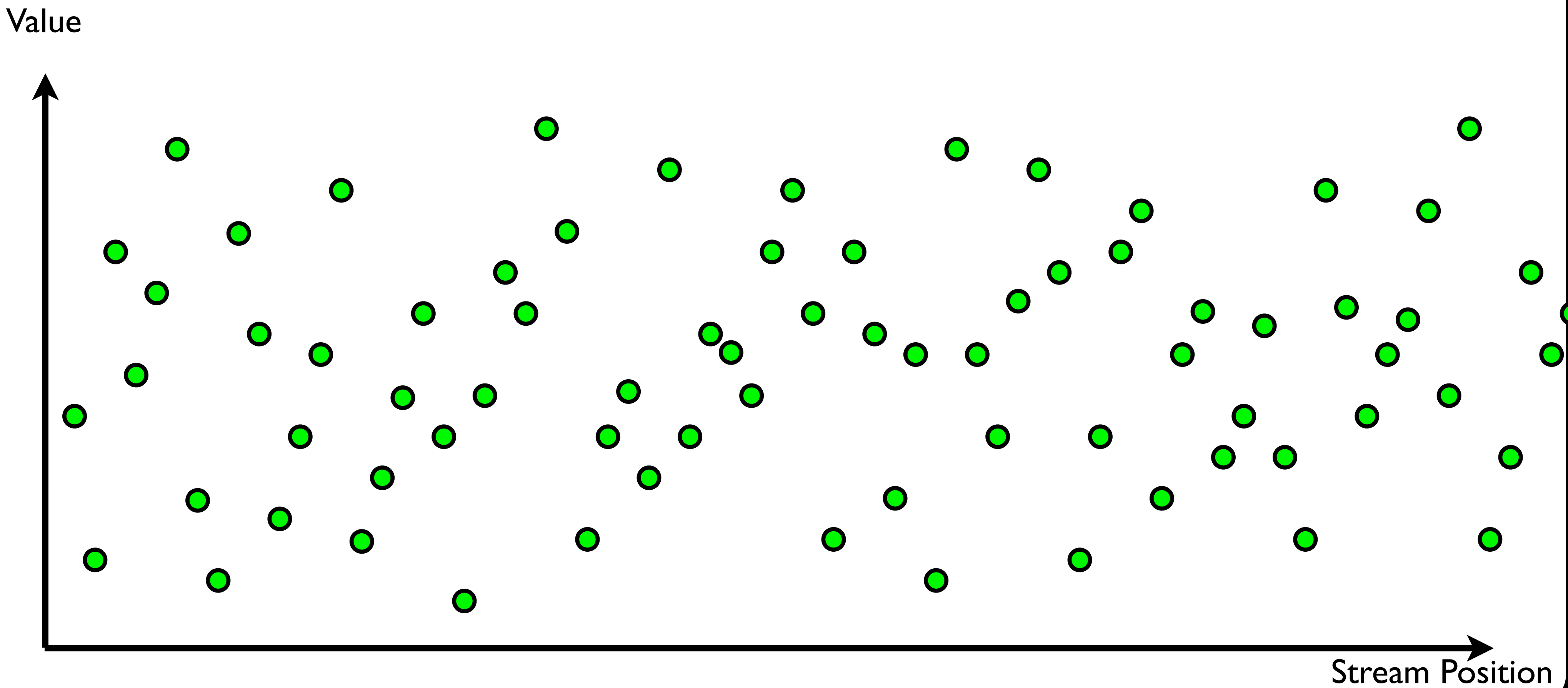
Algorithm

1) Maintain bounds [a,b] for median and c in [a,b]
2) Split stream in segments: $S_1, E_1, S_2, E_2, \ldots, S_p, E_p$
3) For $i \in [p]$: Sample $c \in S_i \cap [a,b]$



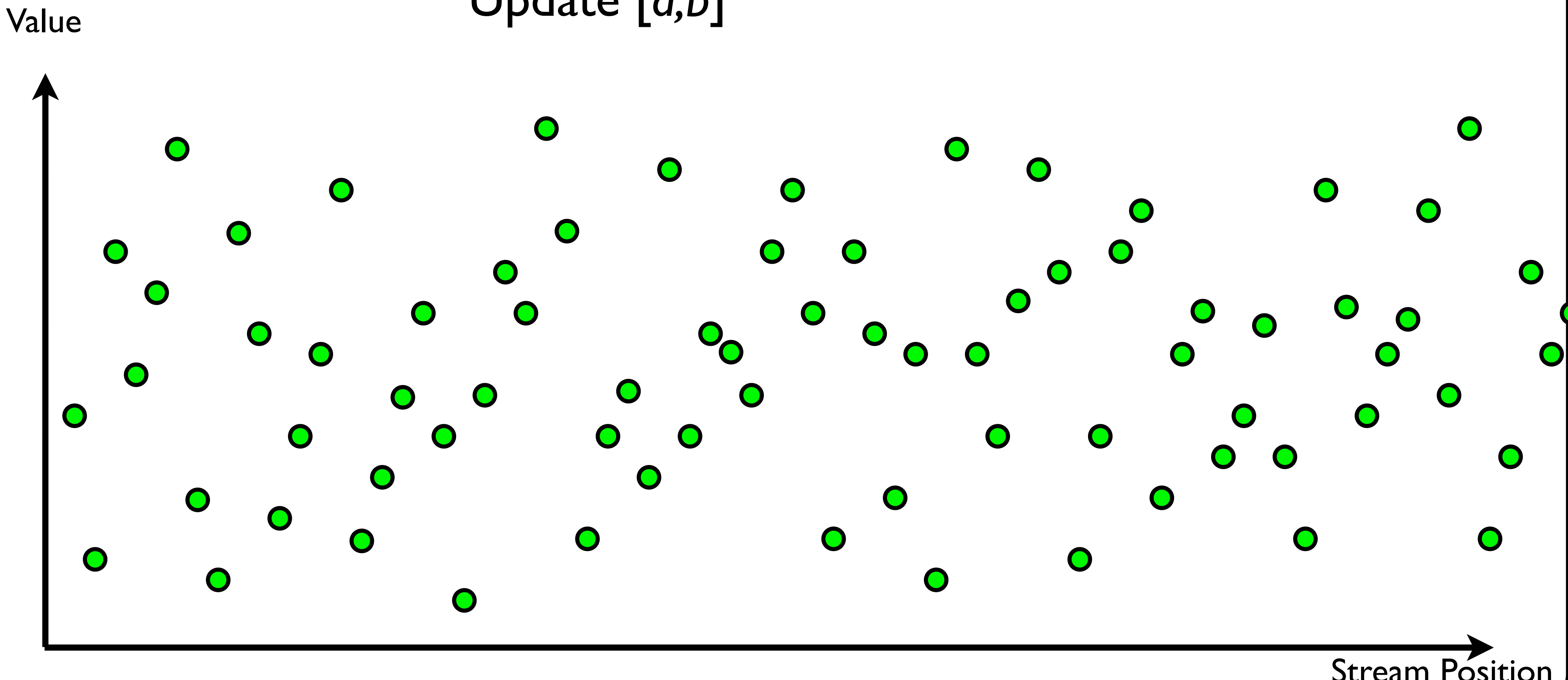
Algorithm

1) Maintain bounds [a,b] for median and c in [a,b]
2) Split stream in segments: $S_1, E_1, S_2, E_2, \ldots, S_p, E_p$
3) For $i \in [p]$: Sample $c \in S_i \cap [a,b]$
   Estimate $rank(c)$ from E_i



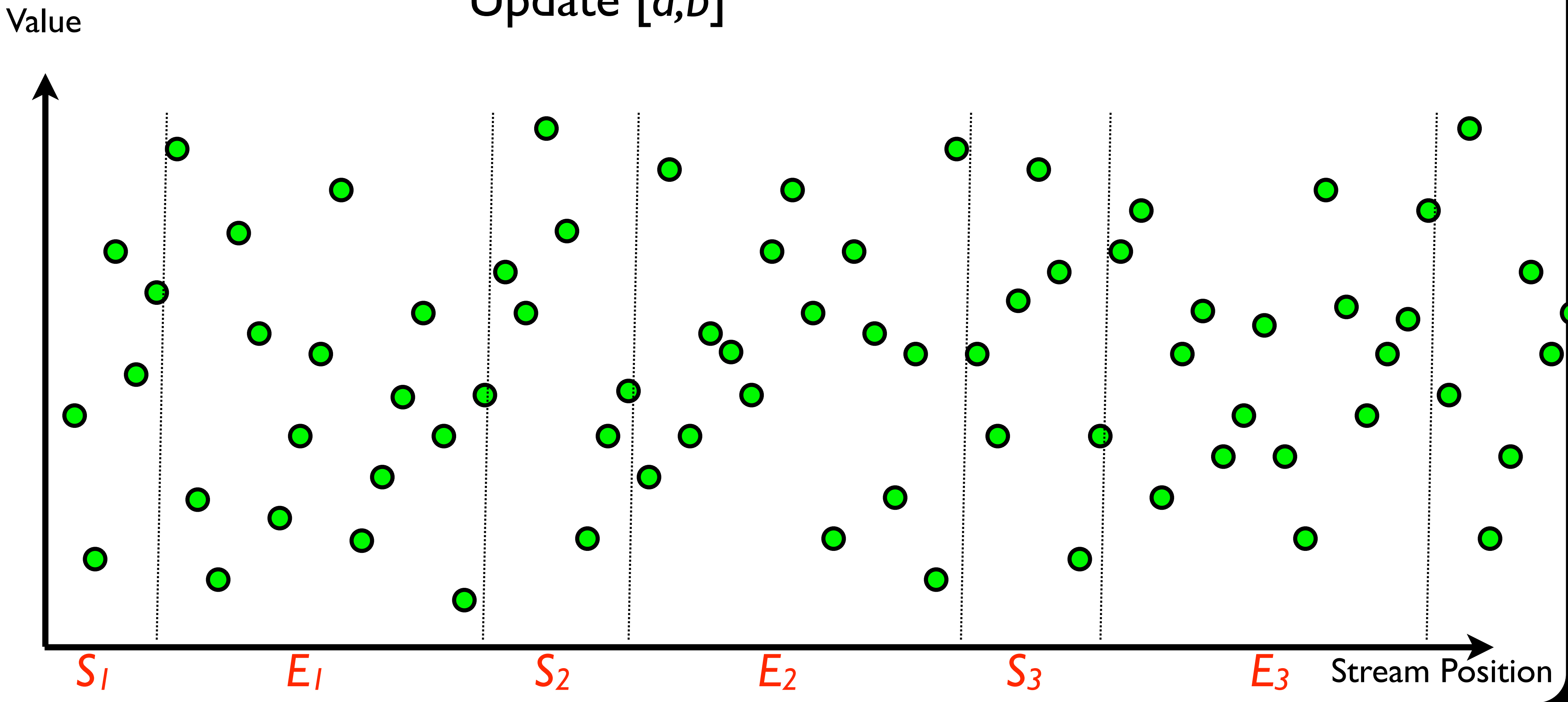
Algorithm

1) Maintain bounds [a,b] for median and c in [a,b]
2) Split stream in segments: $S_1, E_1, S_2, E_2, \ldots, S_p, E_p$
3) For $i \in [p]$: Sample $c \in S_i \cap [a,b]$
   Estimate $rank(c)$ from E_i
   Update [a,b]



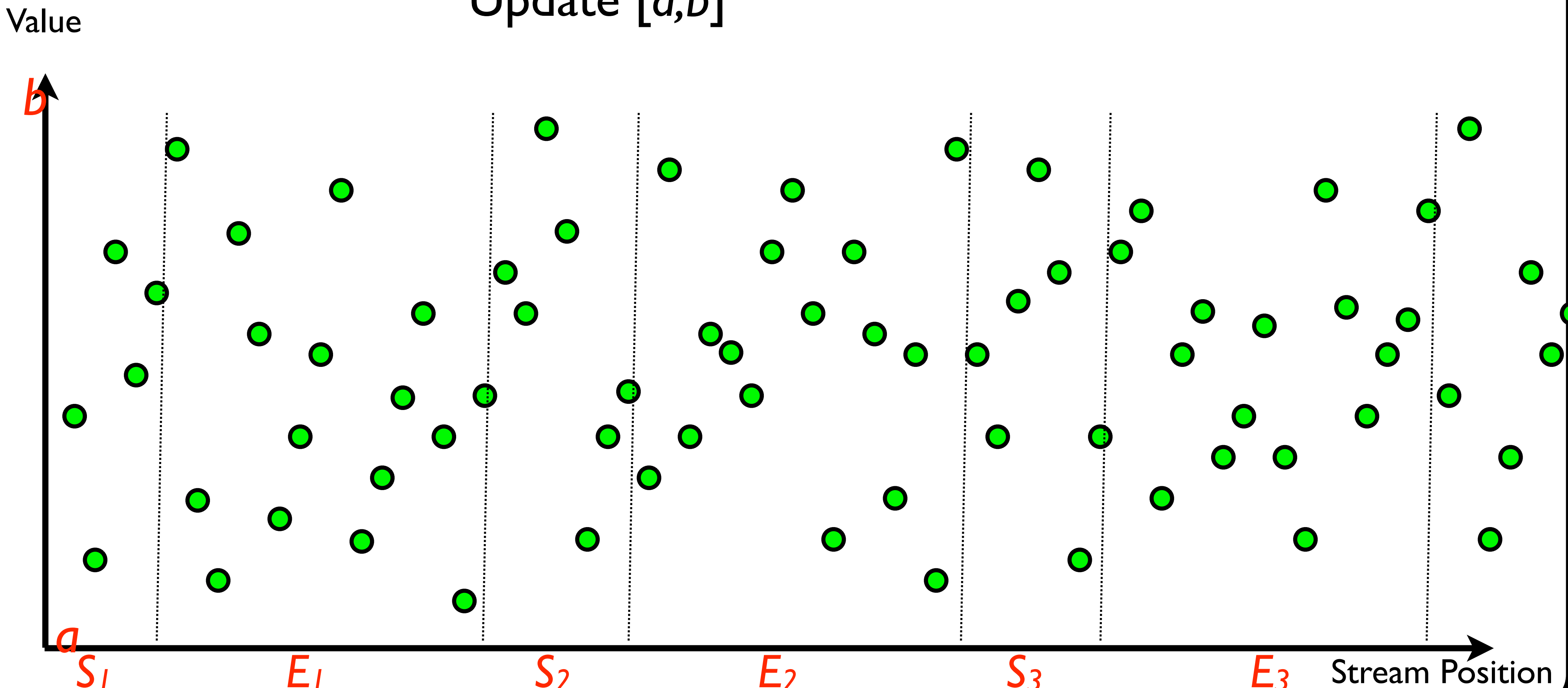
Algorithm

1) Maintain bounds $[a,b]$ for median and c in $[a,b]$
2) Split stream in segments: $S_1, E_1, S_2, E_2, \ldots, S_p, E_p$
3) For $i \in [p]$: Sample $c \in S_i \cap [a,b]$
   Estimate $rank(c)$ from E_i
   Update $[a,b]$



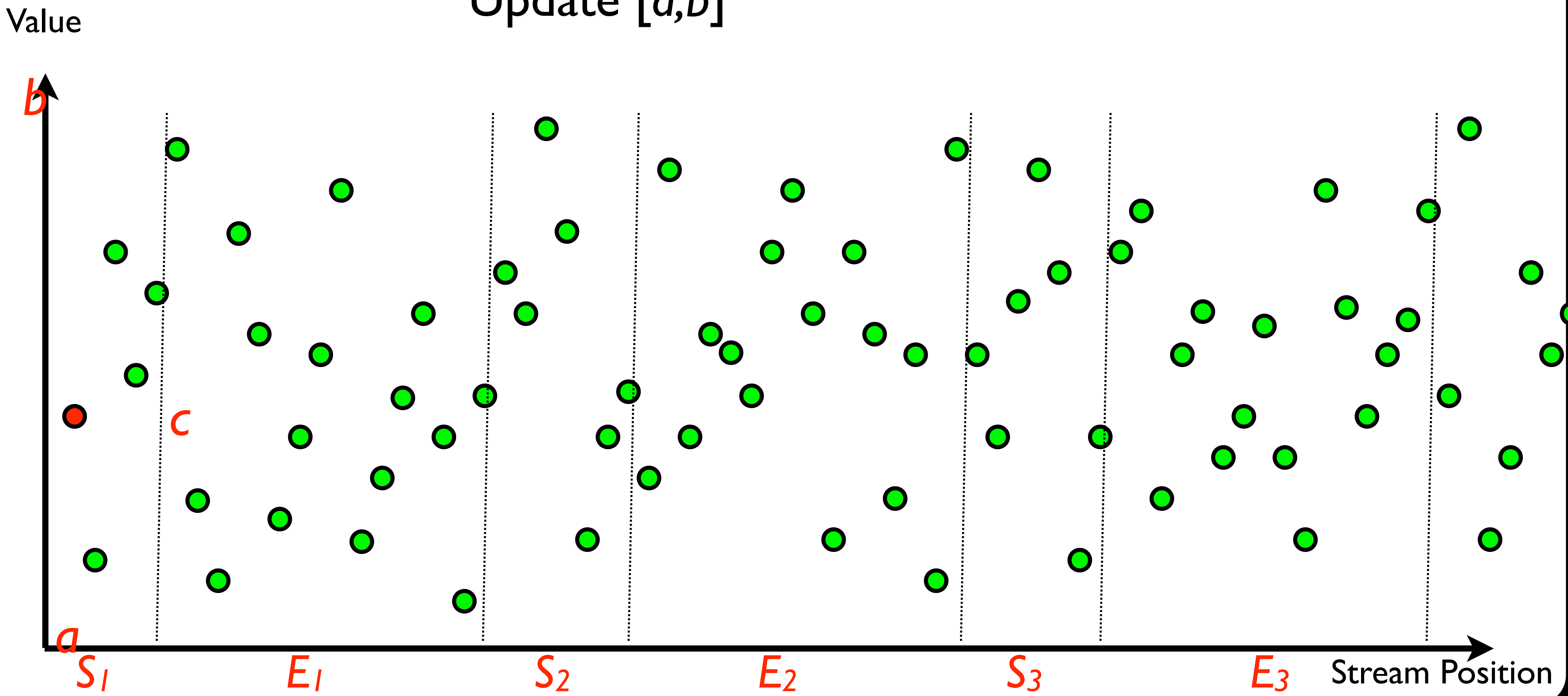
Algorithm

1) Maintain bounds [a,b] for median and c in [a,b]
2) Split stream in segments: $S_1, E_1, S_2, E_2, \ldots, S_p, E_p$
3) For $i \in [p]$: Sample $c \in S_i \cap [a,b]$
   Estimate *rank(c)* from E_i
   Update [a,b]



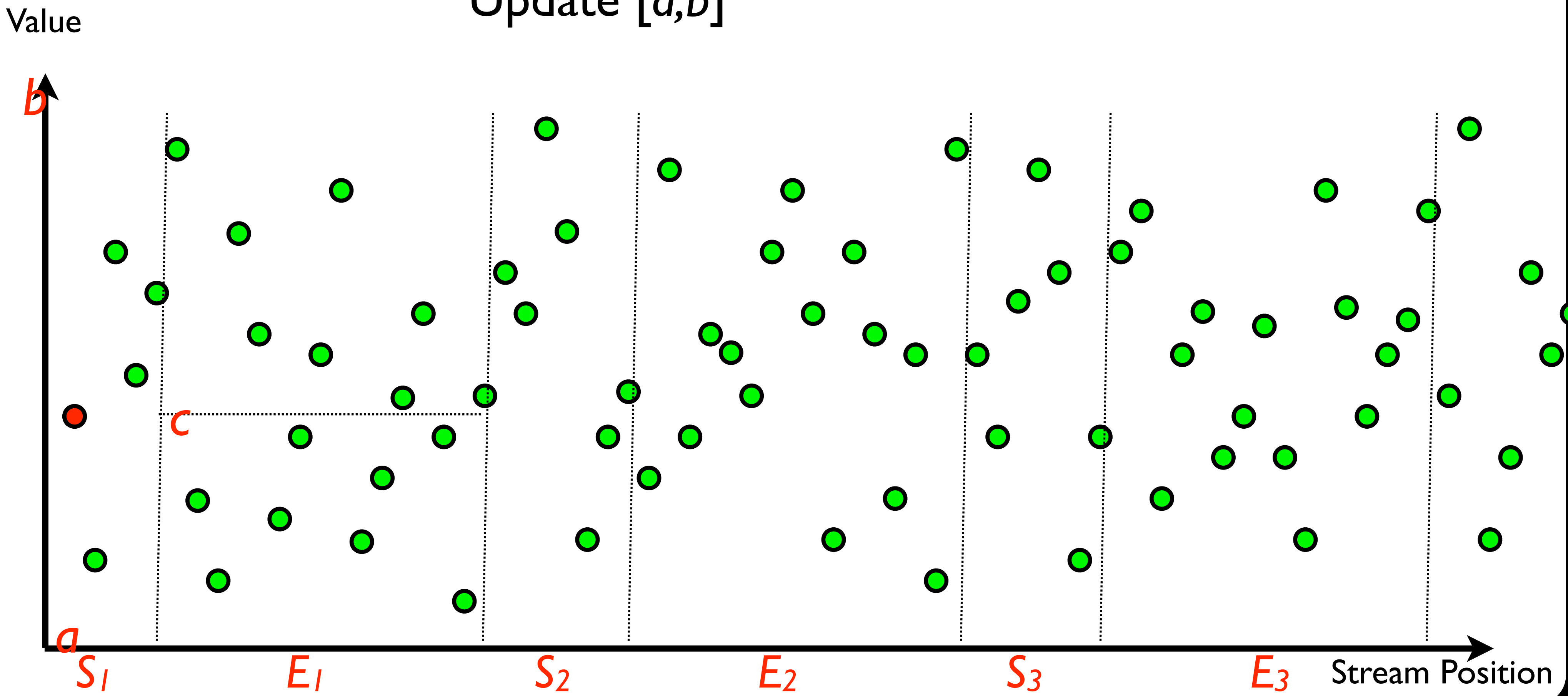
Algorithm

1) Maintain bounds $[a,b]$ for median and c in $[a,b]$
2) Split stream in segments: $S_1, E_1, S_2, E_2, \ldots, S_p, E_p$
3) For $i \in [p]$: Sample $c \in S_i \cap [a,b]$
   Estimate $rank(c)$ from E_i
   Update $[a,b]$



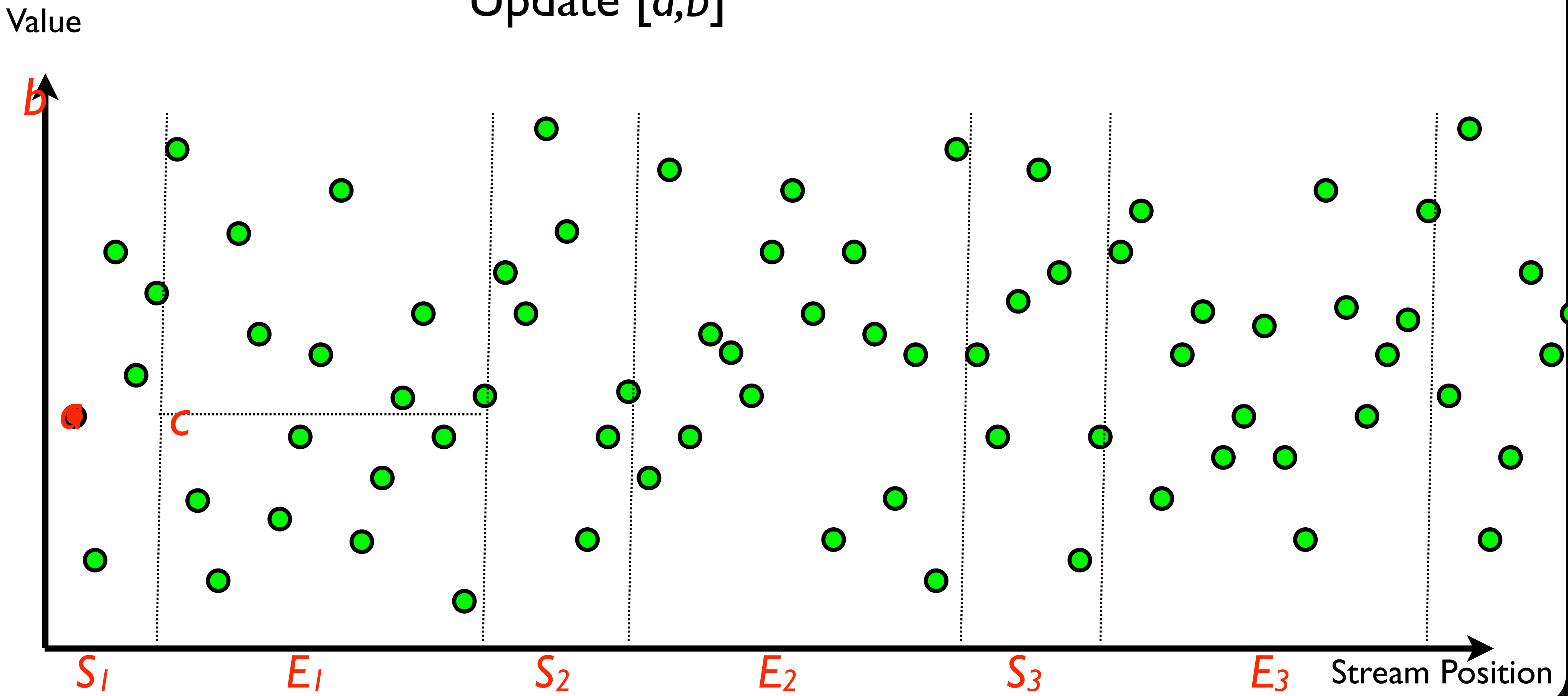
Algorithm

1) Maintain bounds $[a,b]$ for median and c in $[a,b]$
2) Split stream in segments: $S_1, E_1, S_2, E_2, \ldots, S_p, E_p$
3) For $i \in [p]$: Sample $c \in S_i \cap [a,b]$
   Estimate $rank(c)$ from E_i
   Update $[a,b]$



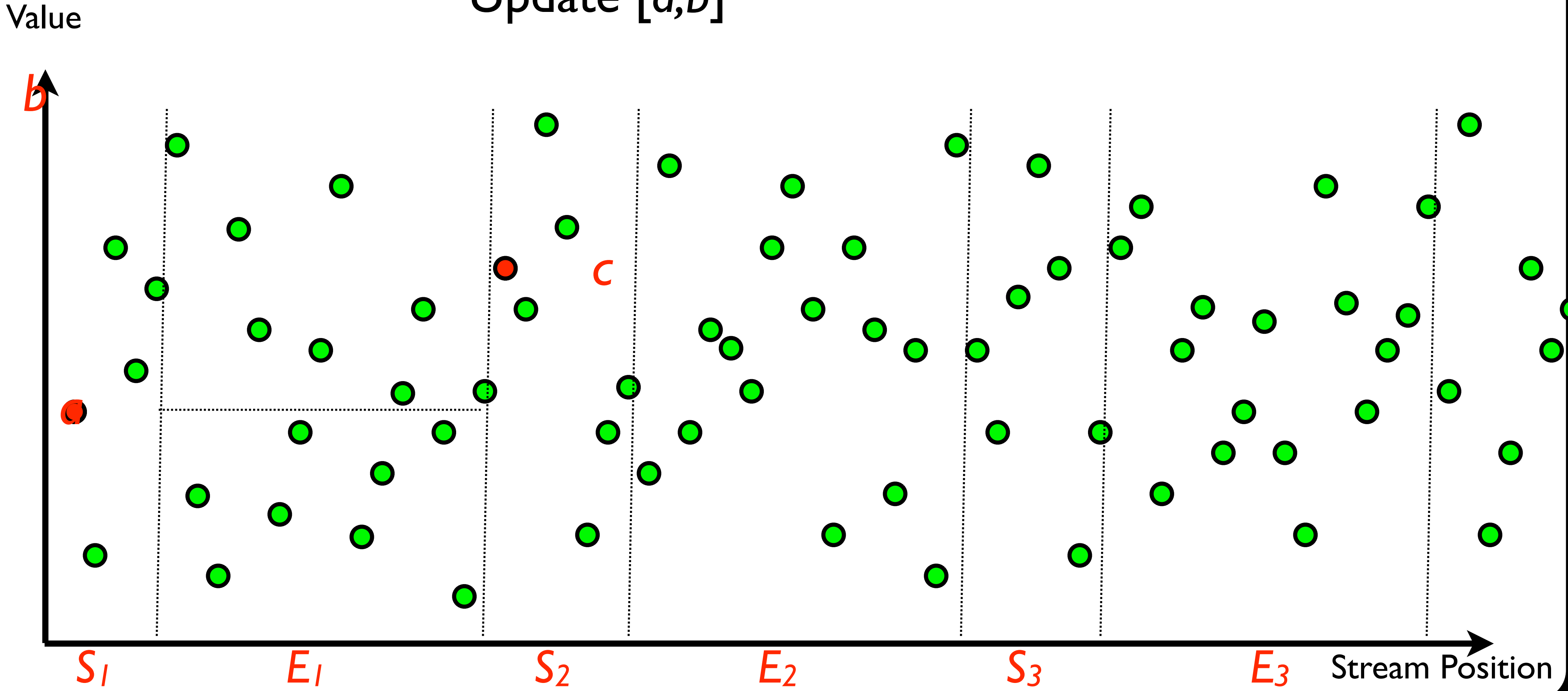
Algorithm

1) Maintain bounds [*a*,*b*] for median and *c* in [*a*,*b*]
2) Split stream in segments: $S_1, E_1, S_2, E_2, \ldots, S_p, E_p$
3) For $i \in [p]$: Sample $c \in S_i \cap [a,b]$
   Estimate *rank(c)* from E_i
   Update [*a*,*b*]



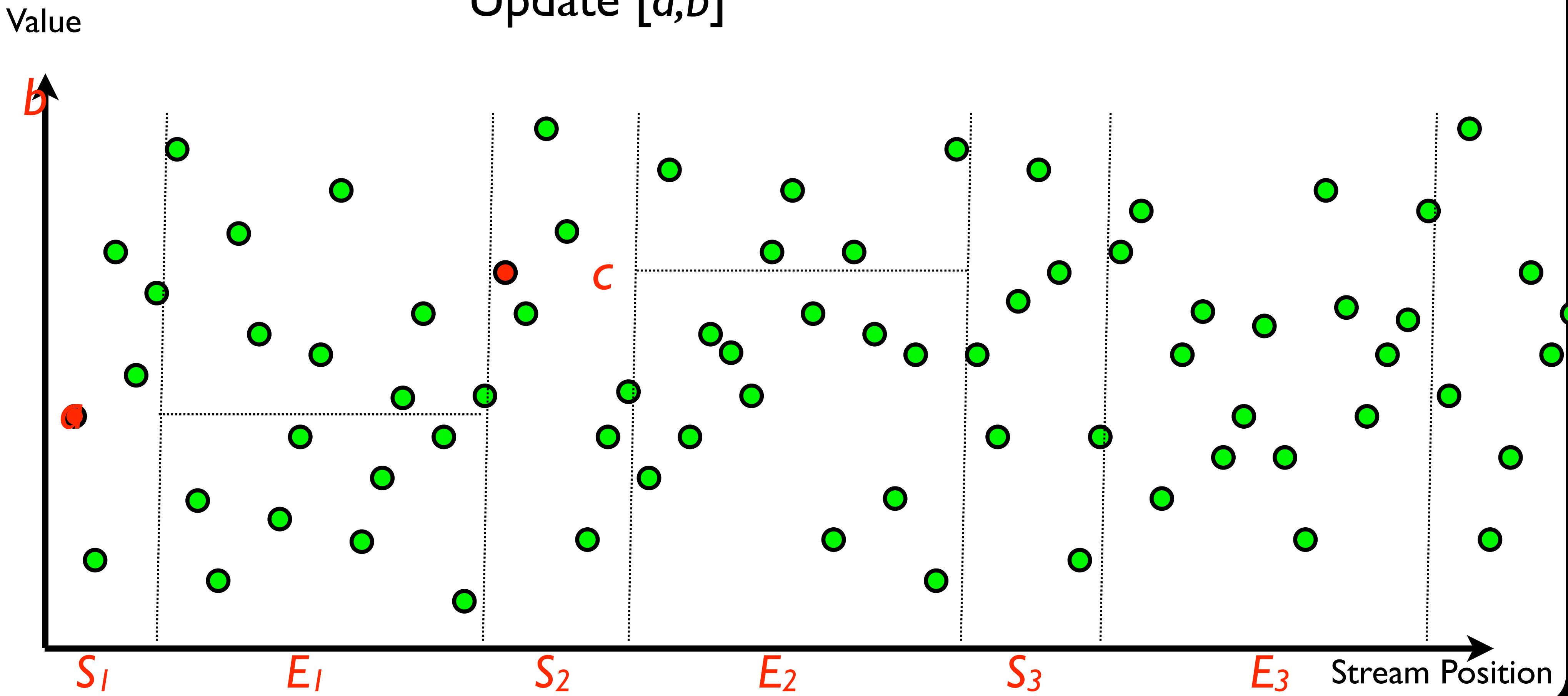
Algorithm

1) Maintain bounds $[a,b]$ for median and c in $[a,b]$
2) Split stream in segments: $S_1, E_1, S_2, E_2, \dots, S_p, E_p$
3) For $i \in [p]$: Sample $c \in S_i \cap [a,b]$
   Estimate $rank(c)$ from E_i
   Update $[a,b]$



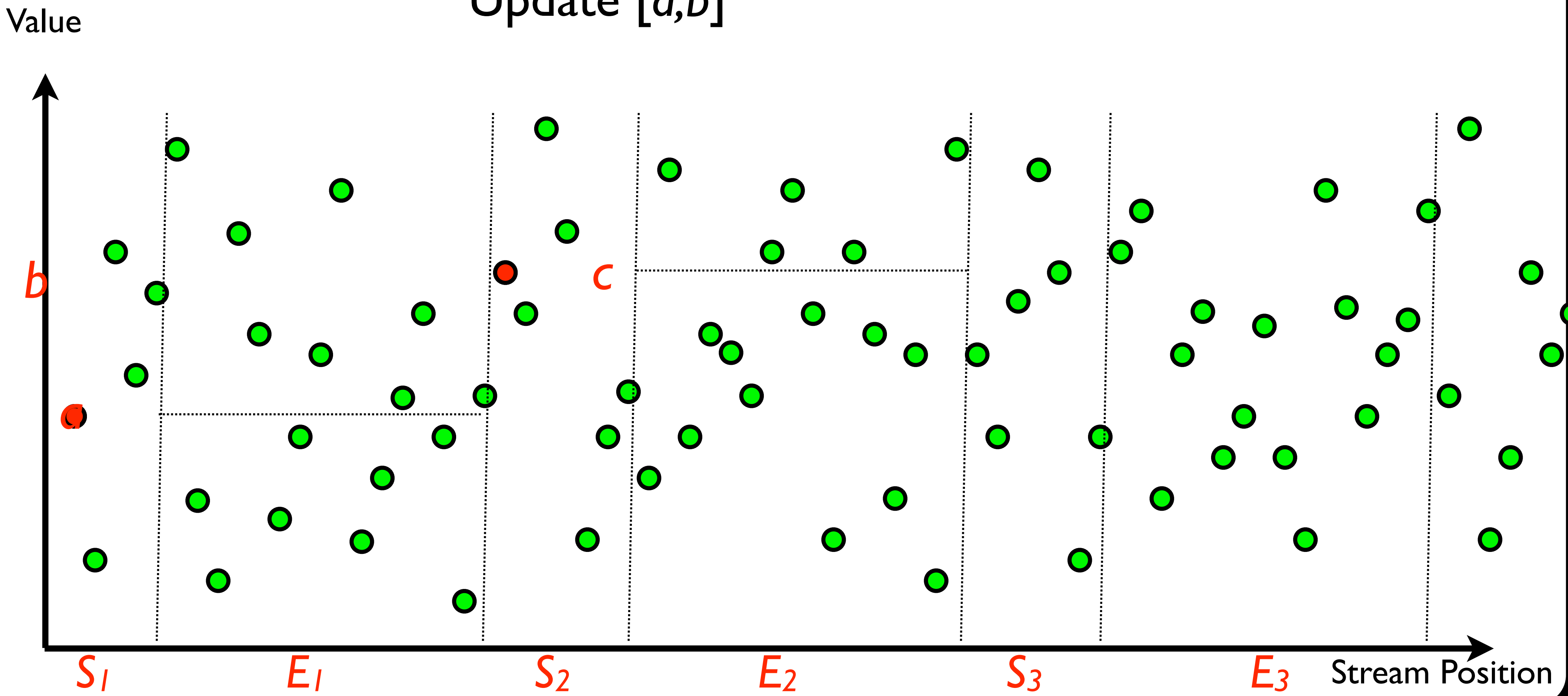
Algorithm

1) Maintain bounds [a,b] for median and c in [a,b]
2) Split stream in segments: $S_1, E_1, S_2, E_2, \ldots, S_p, E_p$
3) For $i \in [p]$: Sample $c \in S_i \cap [a,b]$
   Estimate *rank(c)* from E_i
   Update [a,b]



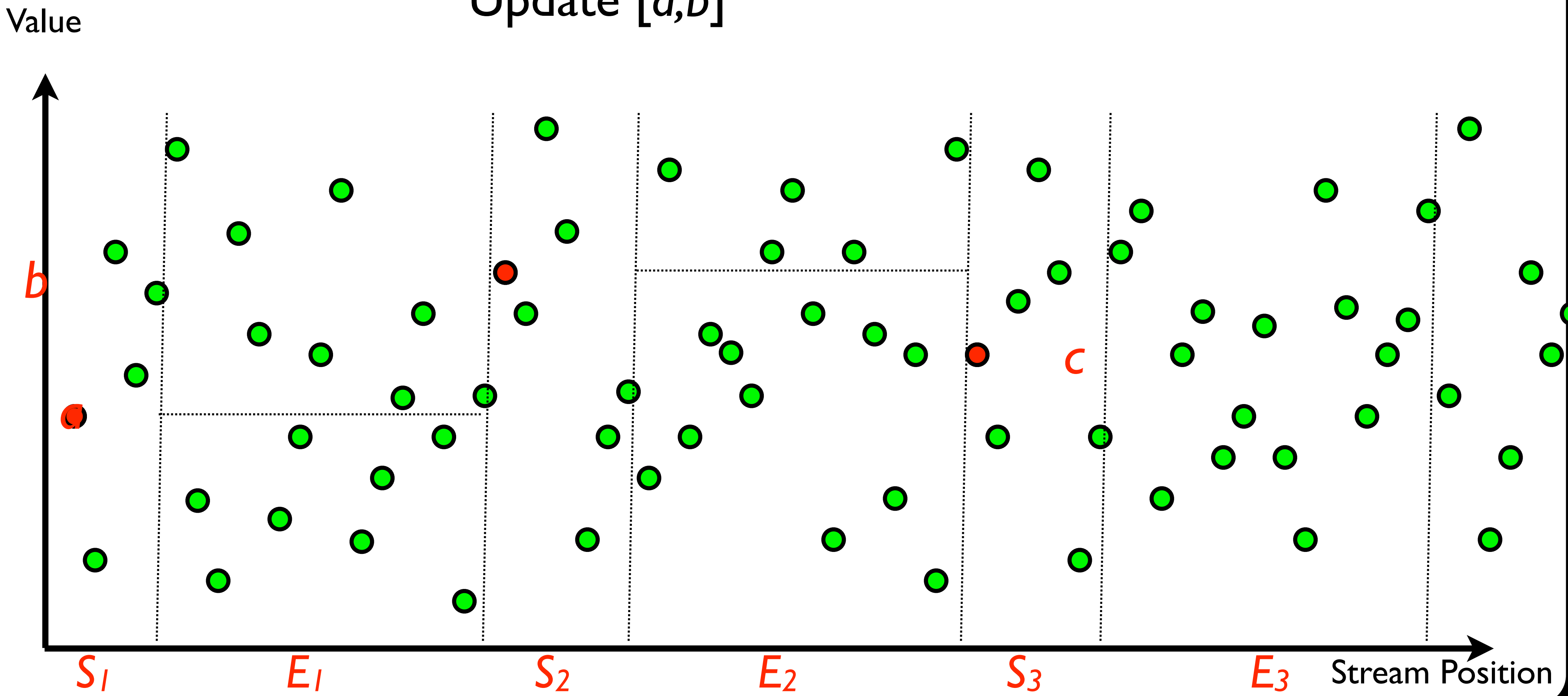
Algorithm

1) Maintain bounds [a,b] for median and c in [a,b]
2) Split stream in segments: $S_1, E_1, S_2, E_2, \ldots, S_p, E_p$
3) For $i \in [p]$: Sample $c \in S_i \cap [a,b]$
   Estimate *rank(c)* from E_i
   Update [a,b]



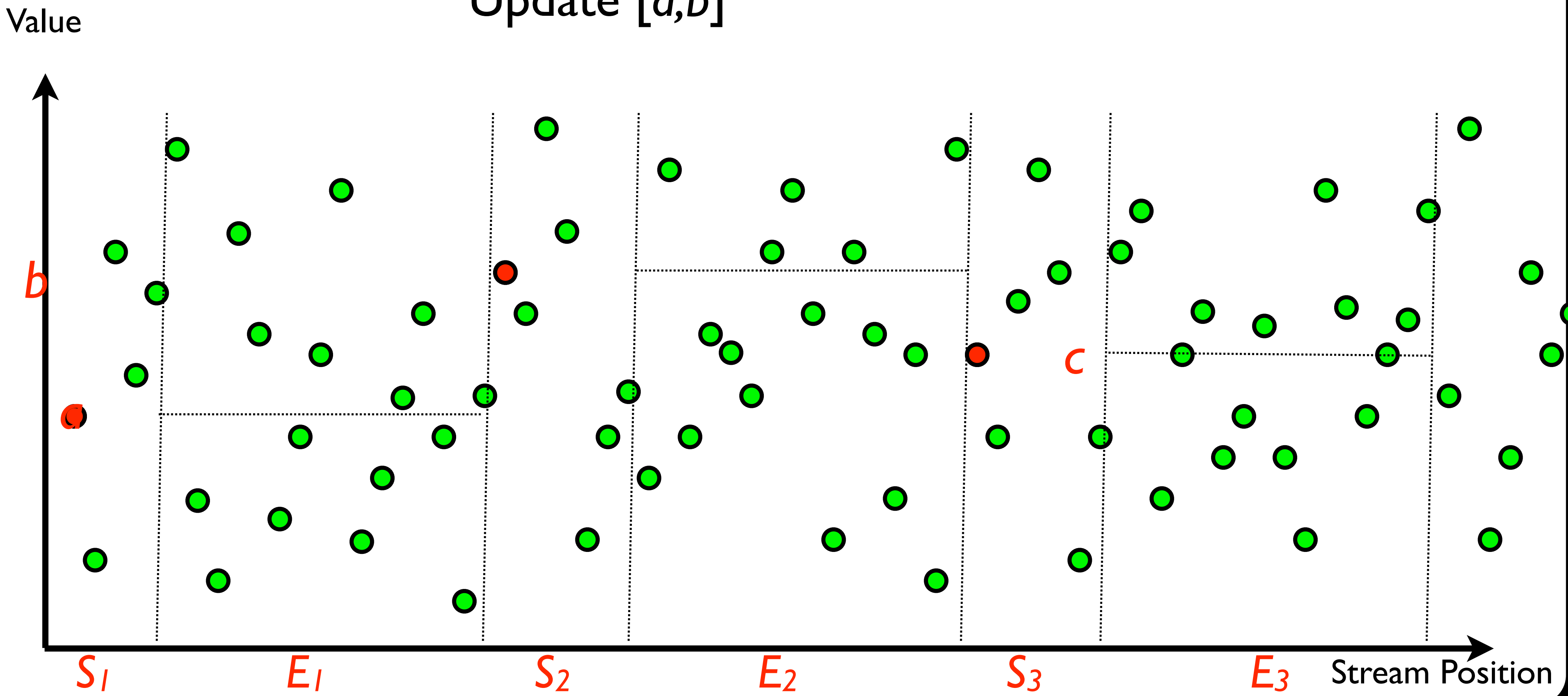
Algorithm

1) Maintain bounds $[a,b]$ for median and c in $[a,b]$
2) Split stream in segments: $S_1, E_1, S_2, E_2, \ldots, S_p, E_p$
3) For $i \in [p]$: Sample $c \in S_i \cap [a,b]$
   Estimate $rank(c)$ from E_i
   Update $[a,b]$



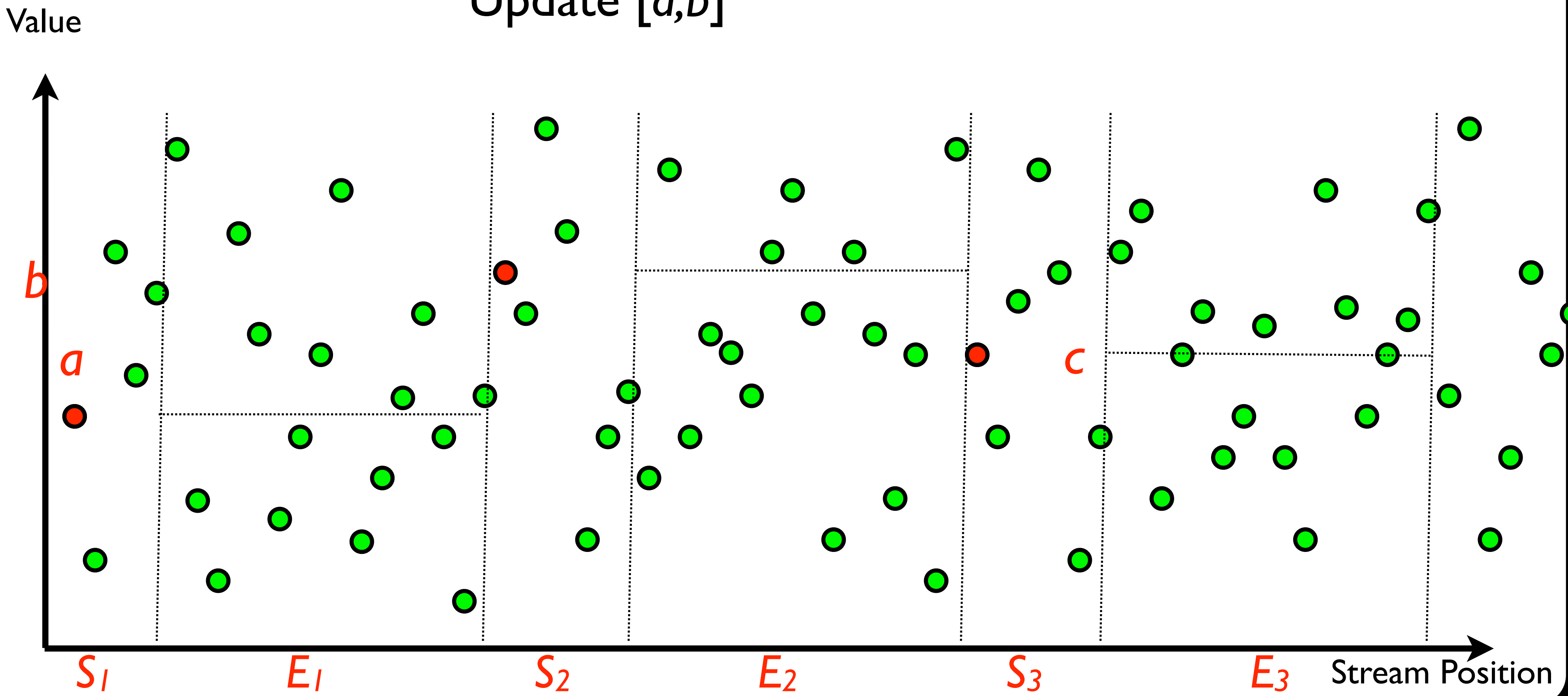
Algorithm

1) Maintain bounds $[a,b]$ for median and c in $[a,b]$
2) Split stream in segments: $S_1, E_1, S_2, E_2, \ldots, S_p, E_p$
3) For $i \in [p]$: Sample $c \in S_i \cap [a,b]$
   Estimate $rank(c)$ from E_i
   Update $[a,b]$



Algorithm

1) Maintain bounds [a,b] for median and c in [a,b]
2) Split stream in segments: $S_1, E_1, S_2, E_2, \ldots, S_p, E_p$
3) For $i \in [p]$: Sample $c \in S_i \cap [a,b]$
   Estimate *rank(c)* from E_i
   Update [a,b]



Analysis

Analysis

- Let t = $O(m^{1/2} \lg^2 m)$

Analysis

- Let $t = O(m^{1/2} \lg^2 m)$
- *Lemma:* For $|E_i| = \Omega(m/\lg m)$, error of estimate of rank(c) is $\pm t$ w.h.p.

Analysis

- Let t = $O(m^{1/2} \lg^2 m)$
- *Lemma:* For $|E_i| = \Omega(m/\lg m)$, error of estimate of rank(c) is $\pm t$ w.h.p.
- *Lemma:* For $|S_i|=\Omega(t)$, if rank(b)-rank(a)=Ω(t), then there exists c in $S_i \cap [a,b]$ w.h.p.

Analysis

- Let t = $O(m^{1/2} \lg^2 m)$
- *Lemma:* For $|E_i| = \Omega(m/\lg m)$, error of estimate of rank(c) is $\pm t$ w.h.p.
- *Lemma:* For $|S_i|=\Omega(t)$, if rank(b)-rank(a)=Ω(t), then there exists c in $S_i \cap [a,b]$ w.h.p.
- *Lemma:* Expect rank(b)-rank(a) to half per-phase, hence $p = O(\lg m)$ w.h.p.

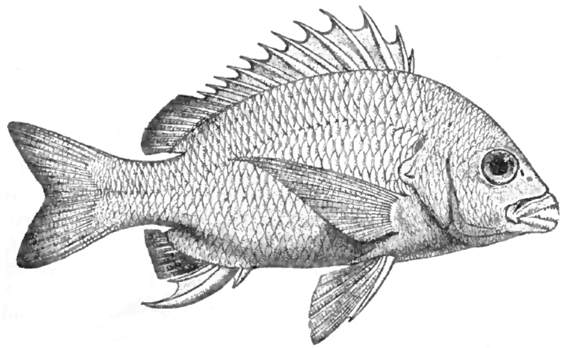
Analysis

- Let t = $O(m^{1/2} \lg^2 m)$
- *Lemma:* For $|E_i| = \Omega(m/\lg m)$, error of estimate of rank(c) is $\pm t$ w.h.p.
- *Lemma:* For $|S_i|=\Omega(t)$, if rank(b)-rank(a)=Ω(t), then there exists c in $S_i \cap [a,b]$ w.h.p.
- *Lemma:* Expect rank(b)-rank(a) to half per-phase, hence $p = O(\lg m)$ w.h.p.
- *Thm:* If stream order is random, we return an element with rank $m/2 \pm t$ using O(log m) space.

1: Algorithm *(Random)*

2: Lower-Bound *(Random)*

3: Lower-Bound *(Advesarial)*



Alice

length *m*

binary string *x*

Bob

index *i* in

range [*m*]

INDEX: "What's the value of x_i?"

Requires $\Omega(n)$ bits transmitted.

Alice

length *m*
binary string *x*

Bob

index *i* in
range [*m*]

INDEX: “What’s the value of x_i?”

Requires $\Omega(n)$ bits transmitted.

Alice
length *m*
binary string *x*

Bob
index *i* in
range [*m*]

- Assume a single-pass (AOM) algorithm for median with prob. at least 3/4 using S space.

$2+x_1$ ... $2i+x_i$ ... $2m+x_m$

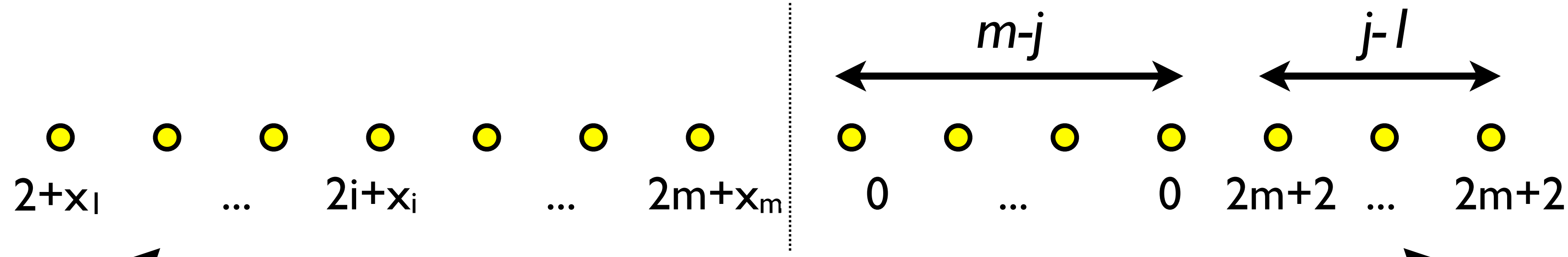
INDEX: "What's the value of x_i?"

Requires $\Omega(n)$ bits transmitted.

Alice
length *m*
binary string *x*

Bob
index *i* in
range [*m*]

- Assume a single-pass (AOM) algorithm for median with prob. at least 3/4 using S space.



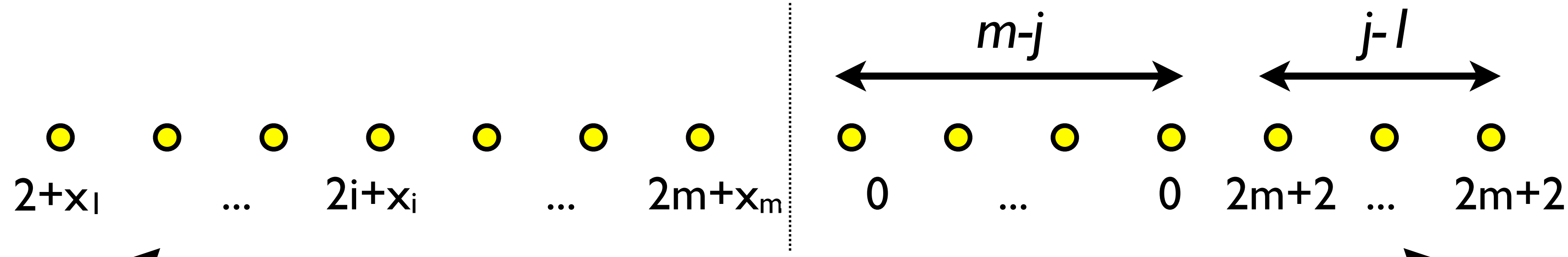
INDEX: “What’s the value of x_i?”

Requires $\Omega(n)$ bits transmitted.

Alice
length *m*
binary string *x*

Bob
index *i* in
range [*m*]

- Assume a single-pass (AOM) algorithm for median with prob. at least 3/4 using S space.



INDEX: "What's the value of x_i?"

Requires $\Omega(n)$ bits transmitted.

Alice
length *m*
binary string *x*

MEMORY STATE OF ALGORITHM

Bob
index *i* in
range [*m*]

- Assume a single-pass (AOM) algorithm for median with prob. at least 3/4 using S space.

m-j j-l

$2+x_1$ ... $2i+x_i$ ... $2m+x_m$ | 0 ... 0 $2m+2$ ... $2m+2$

INDEX: "What's the value of x_i?"

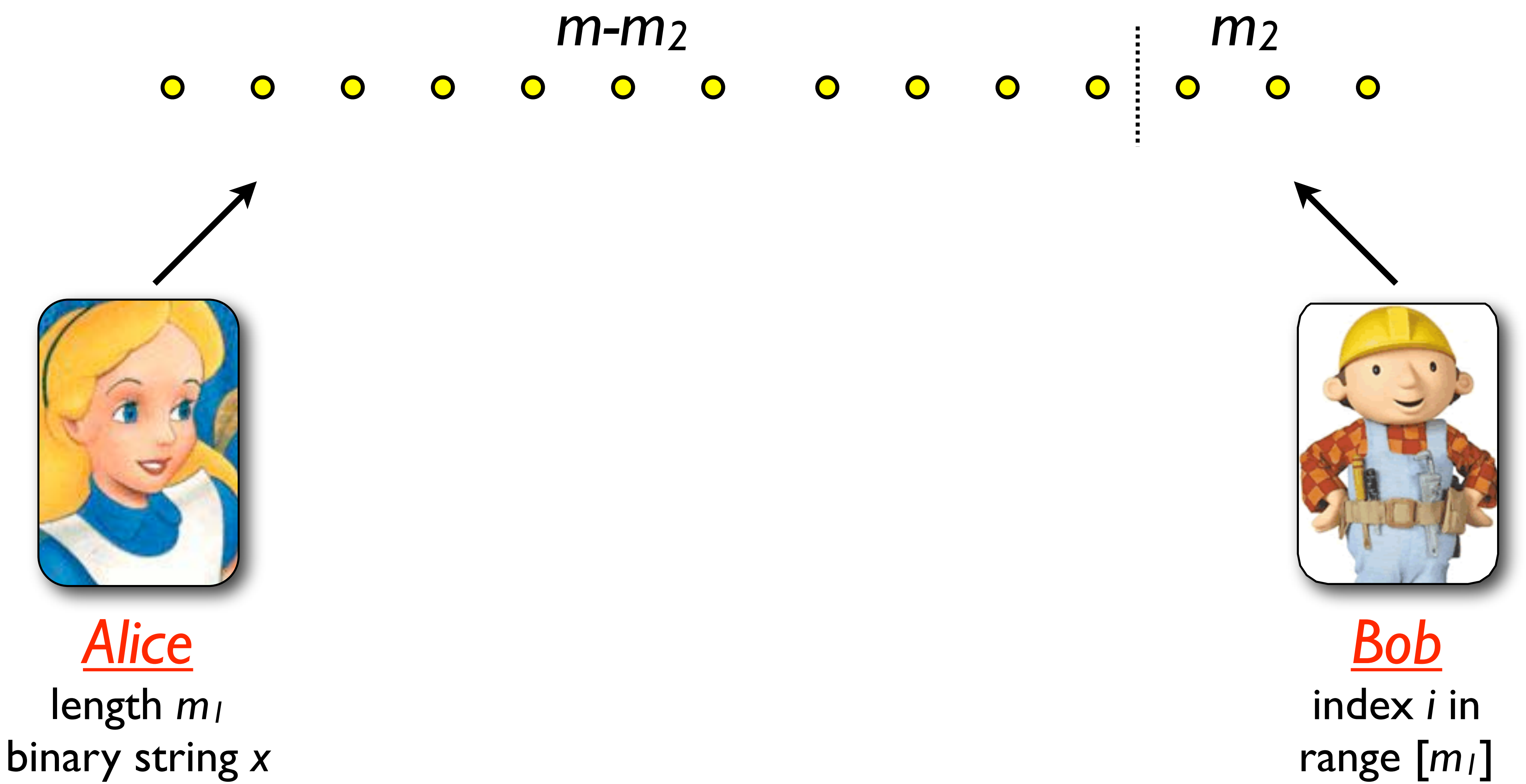
Requires $\Omega(n)$ bits transmitted.

Alice
length m
binary string x

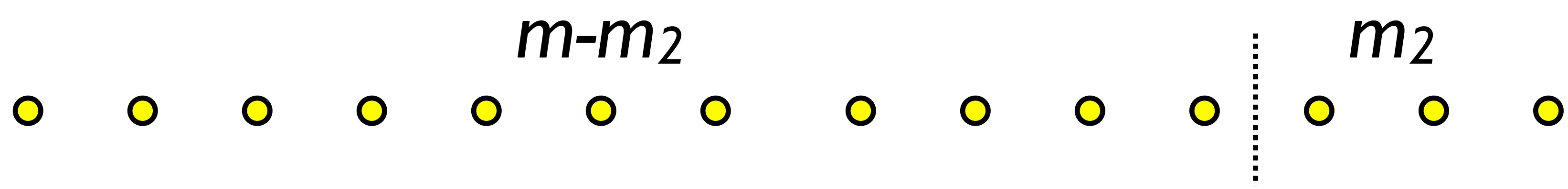
MEMORY STATE OF ALGORITHM

Bob
index i in range $[m]$

- Assume a single-pass (AOM) algorithm for median with prob. at least 3/4 using S space.
- *Thm:* $S=\Omega(m)$ [Henzinger, Raghavan, Rajagopalan '99]



m-m2
m2
Alice
length m1
binary string x
Bob
index i in
range [m1]



Alice: picks b randomly from $[m_1]$ and inserts a random permutation of,

$$\{\underbrace{0, \ldots, 0}_{(m-m_1-m_2-b)/2}, 2+x_1, \ldots, 2m_1+x_{m_1}, \underbrace{2m+2, \ldots, 2m+2}_{(m-m_1-m_2+b)/2}\}$$

Alice

length m_1

binary string x

Bob

index i in

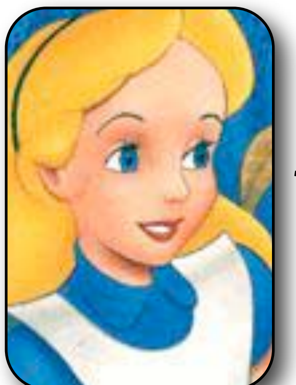
range $[m_1]$

m-m_2 m_2

Alice: picks b randomly from $[m_1]$ and inserts a random permutation of,

$$\{\underbrace{0, \ldots, 0}_{(m-m_1-m_2-b)/2}, 2+x_1, \ldots, 2m_1+x_{m_1}, \underbrace{2m+2, \ldots, 2m+2}_{(m-m_1-m_2+b)/2}\}$$

MEMORY STATE OF ALGORITHM and "b"



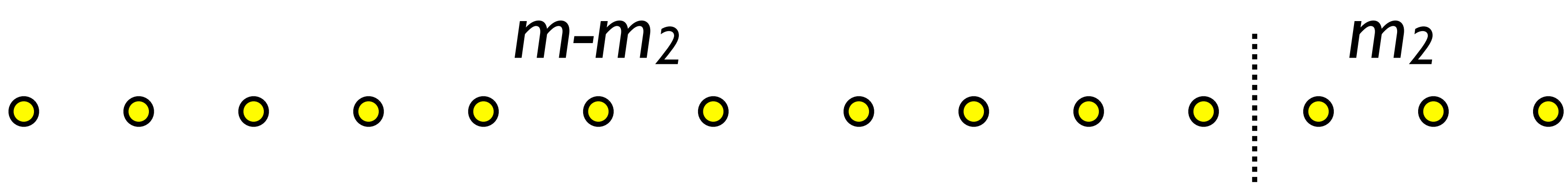
Alice

length m_1

binary string x

Bob

index i in range $[m_1]$



Alice: picks b randomly from $[m_1]$ and inserts a random permutation of,

$$\{\underbrace{0, \quad \dots \quad, 0}_{(m-m_1-m_2-b)/2}, 2+x_1, \dots, 2m_1+x_{m_1}, \underbrace{2m+2, \dots, 2m+2}_{(m-m_1-m_2+b)/2}\}$$

MEMORY STATE OF ALGORITHM and "b"

Bob: inserts a random permutation of,

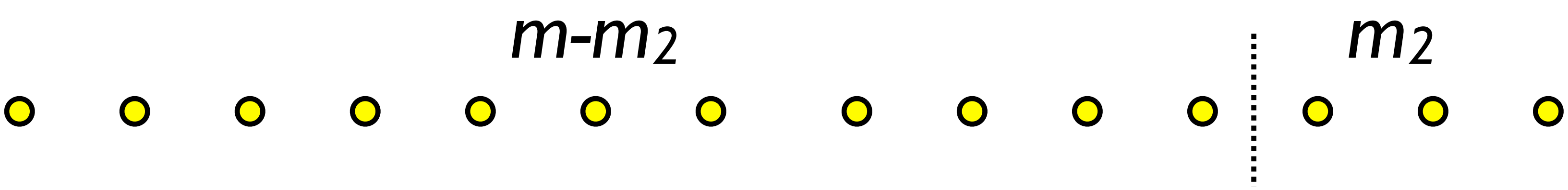
$$\{\underbrace{0, \quad \dots \quad, 0}_{(m_2+b-j-1)/2}, \underbrace{2m+2, \dots, 2m+2}_{(m_2-b+j)/2}\}$$

Alice

length m_1
binary string x

Bob

index i in range $[m_1]$



Alice: picks b randomly from $[m_1]$ and inserts a random permutation of,

$$\{\underbrace{0, \quad \ldots \quad, 0}_{(m-m_1-m_2-b)/2}, 2+x_1, \ldots, 2m_1+x_{m_1}, \underbrace{2m+2, \ldots, 2m+2}_{(m-m_1-m_2+b)/2}\}$$

Alice

length m_1

binary string x

MEMORY STATE OF ALGORITHM and "b"

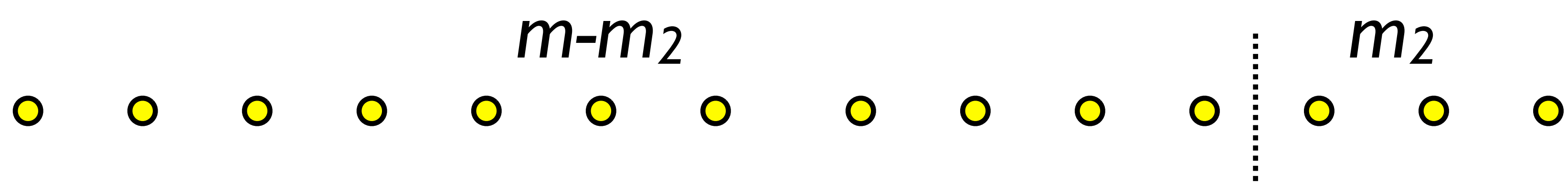
Bob: inserts a random permutation of,

$$\{\underbrace{0, \quad \ldots \quad, 0}_{(m_2+b-j-1)/2}, \underbrace{2m+2, \ldots, 2m+2}_{(m_2-b+j)/2}\}$$

Bob

index i in range $[m_1]$

- If $m_1=o(\sqrt{m_2})$ and $m_1 m_2=o(m)$ then ordering is "close" to random in L_1 sense.



Alice: picks b randomly from $[m_1]$ and inserts a random permutation of,

$$\{\underbrace{0, \ldots, 0}_{(m-m_1-m_2-b)/2}, 2+x_1, \ldots, 2m_1+x_{m_1}, \underbrace{2m+2, \ldots, 2m+2}_{(m-m_1-m_2+b)/2}\}$$

MEMORY STATE OF ALGORITHM and "b"

Bob: inserts a random permutation of,

$$\{\underbrace{0, \ldots, 0}_{(m_2+b-j-1)/2}, \underbrace{2m+2, \ldots, 2m+2}_{(m_2-b+j)/2}\}$$

Alice

length m_1 binary string x

Bob

index i in range $[m_1]$

- If $m_1=o(\sqrt{m_2})$ and $m_1 m_2=o(m)$ then ordering is "close" to random in L_1 sense.
- An algorithm succeeding w/p 3/4 for random-ordering, succeeds w/p 2/3.

m-m_2 | m_2

Alice: picks b randomly from $[m_1]$ and inserts a random permutation of,

$$\{\underbrace{0,\ \ldots\ ,0}_{(m-m_1-m_2-b)/2},2+x_1,\ldots,2m_1+x_{m_1},\underbrace{2m+2,\ldots,2m+2}_{(m-m_1-m_2+b)/2}\}$$

Alice

length m_1

binary string x

MEMORY STATE OF ALGORITHM and "*b*"

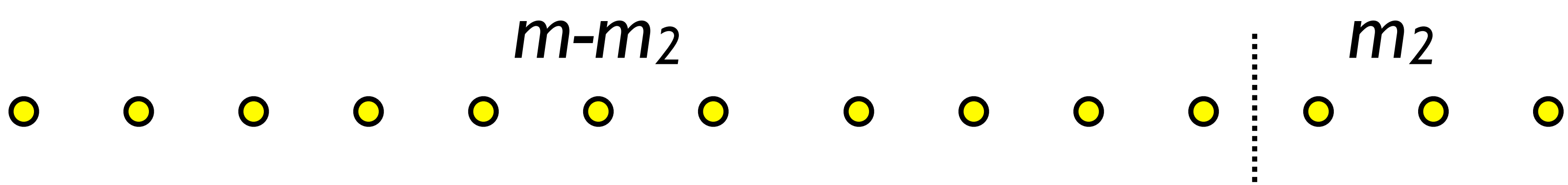
Bob: inserts a random permutation of,

$$\{\underbrace{0,\ \ldots\ ,0}_{(m_2+b-j-1)/2},\underbrace{2m+2,\ldots,2m+2}_{(m_2-b+j)/2}\}$$

Bob

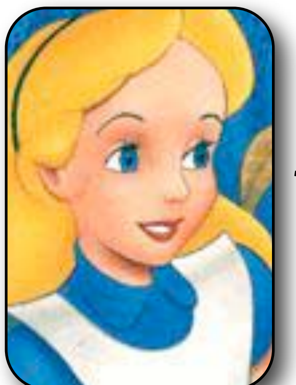
index i in range $[m_1]$

- If $m_1=o(\sqrt{m_2})$ and $m_1 m_2=o(m)$ then ordering is "close" to random in L_1 sense.
- An algorithm succeeding w/p 3/4 for random-ordering, succeeds w/p 2/3.
- *Thm:* $S=\Omega(m^{1/3})$



Alice: picks b randomly from $[m_1]$ and inserts a random permutation of,

$$\{\underbrace{0, \quad \ldots \quad, 0}_{(m-m_1-m_2-b)/2}, 2+x_1, \ldots, 2m_1+x_{m_1}, \underbrace{2m+2, \ldots, 2m+2}_{(m-m_1-m_2+b)/2}\}$$



Alice

length m_1

binary string x

MEMORY STATE OF ALGORITHM and "b"

Bob: inserts a random permutation of,

$$\{\underbrace{0, \quad \ldots \quad, 0}_{(m_2+b-j-1)/2}, \underbrace{2m+2, \ldots, 2m+2}_{(m_2-b+j)/2}\}$$

Bob

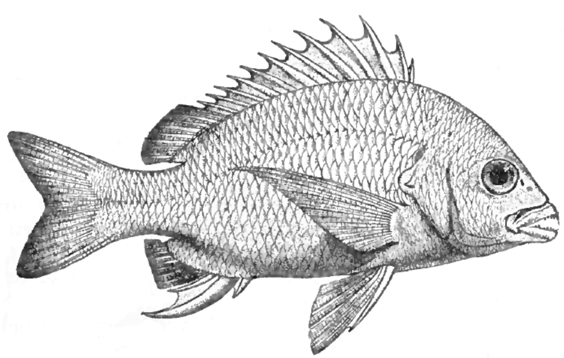
index i in range $[m_1]$

- If $m_1=o(\sqrt{m_2})$ and $m_1 m_2=o(m)$ then ordering is "close" to random in L_1 sense.
- An algorithm succeeding w/p 3/4 for random-ordering, succeeds w/p 2/3.
- *Thm:* $S=\Omega(m^{1/2})$

1: Algorithm *(Random)*

2: Lower-Bound *(Random)*

3: Lower-Bound *(Advesarial)*



Pointer Chasing

Alice
function
f_A: [t]→[t]

Bob
function
f_B: [t]→[t]

Pointer Chasing

Alice

function

$f_A: [t] \to [t]$

Compute

$$f_A(f_B(\ \dots\ (f_A(f_B(f_A(1))))\ \dots\ ))$$

$\longleftarrow$ k $\longrightarrow$

Bob

function

$f_B: [t] \to [t]$

Pointer Chasing

Alice

function

$f_A: [t] \to [t]$

Compute

$$f_A(f_B(\ldots (f_A(f_B(f_A(1)))) \ldots))$$

$\longleftrightarrow$ k

If Bob speaks first:

k messages: $O(k \log t)$ bits.

k-1 messages: $\Omega(t)$ bits.

[Nisan, Wigderson '93]

Bob

function

$f_B: [t] \to [t]$

Multi-Party Pointer Chasing

Alice
function
$f_A: [t] \to [t]$

Bob
function
$f_B: [t] \to [t]$

Claude
function
$f_C: [t] \to [t]$

...

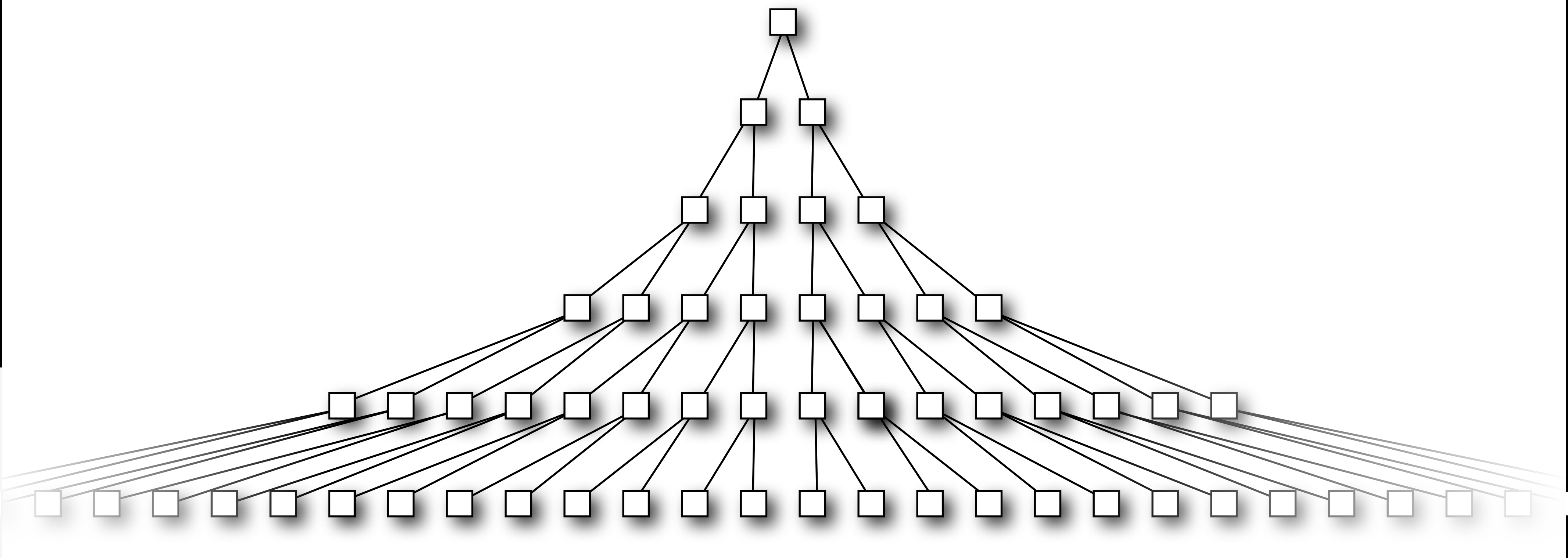
Zebedee
function
$f_Z: [t] \to [t]$

Compute $f_Z(\ \dots f_C(f_B(f_A(1))) \dots)$

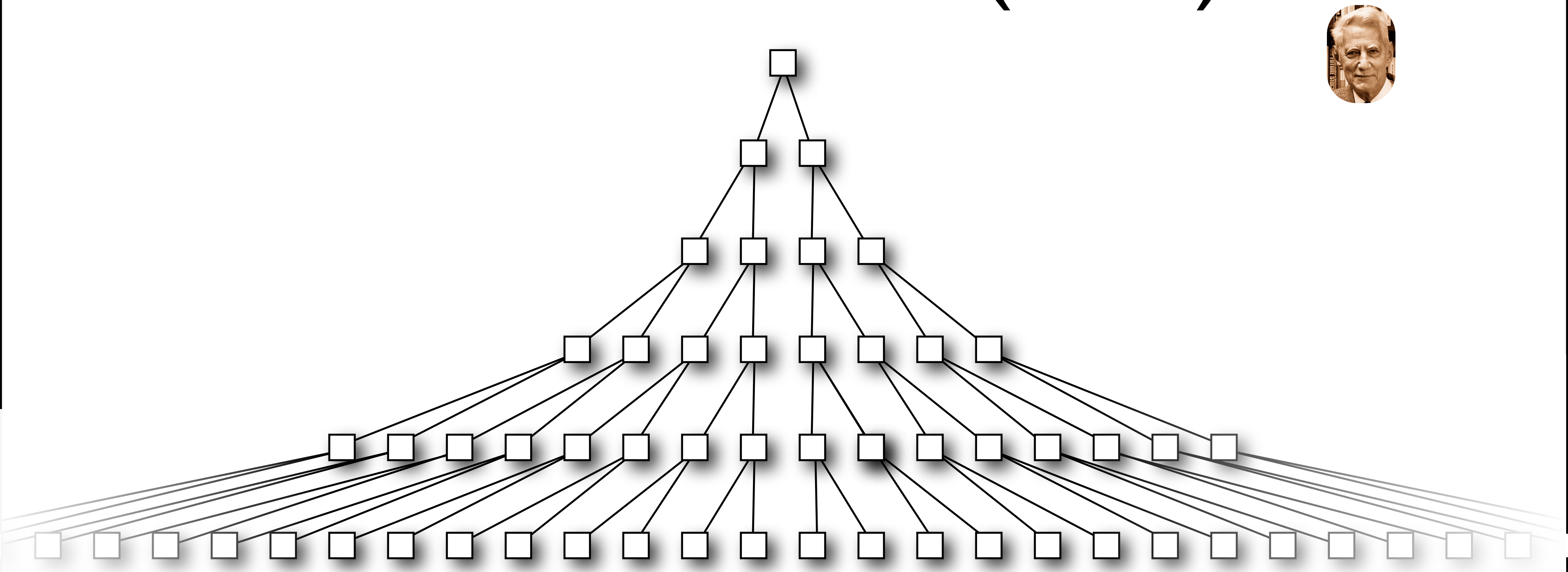
Order of speaking "Z, ..., C, B, A"

- k rounds: $O(k \log t)$ bits.
- k-1 rounds: $\Omega(t)$ bits.

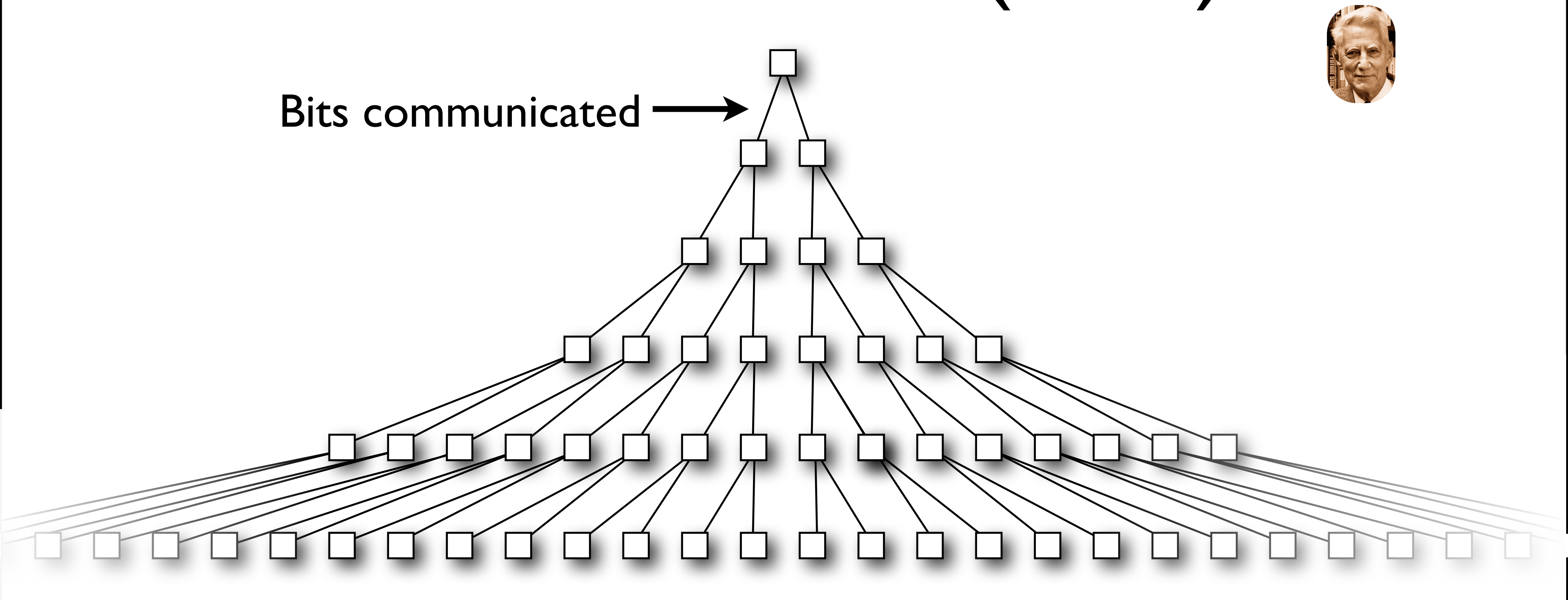
Proof Sketch (k=3)



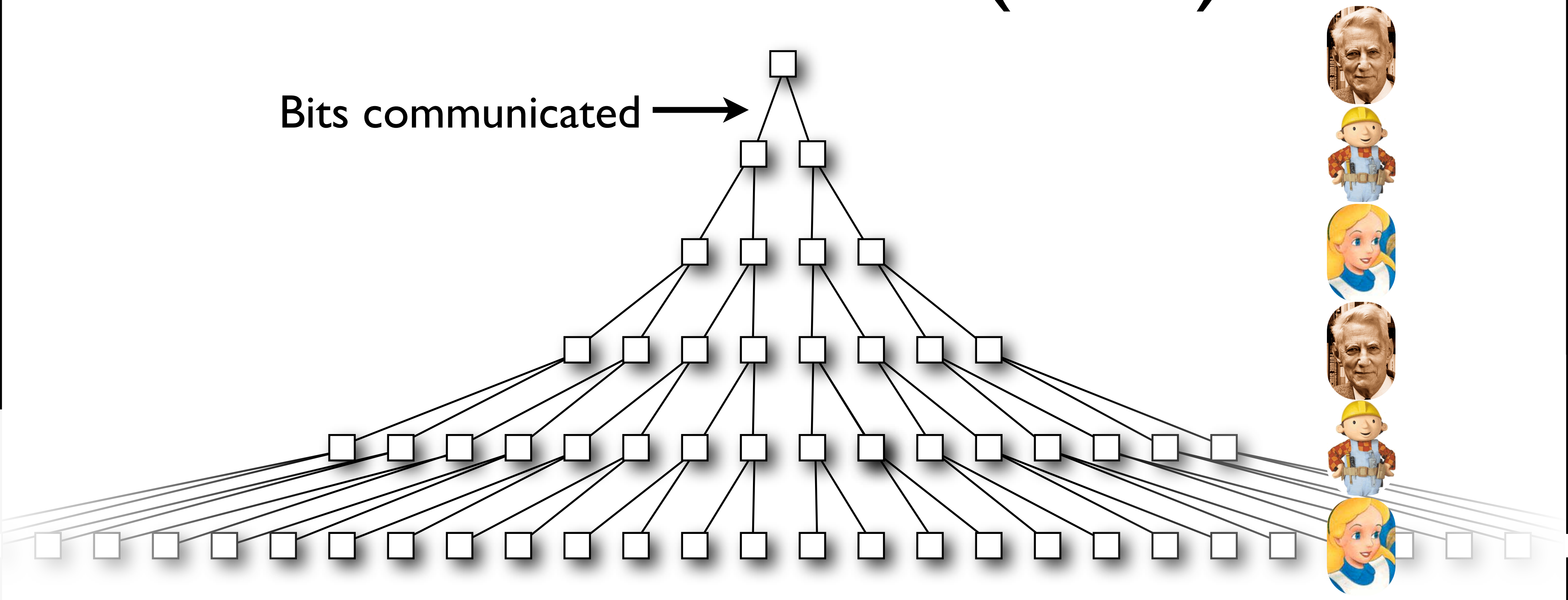
Proof Sketch (*k=3*)



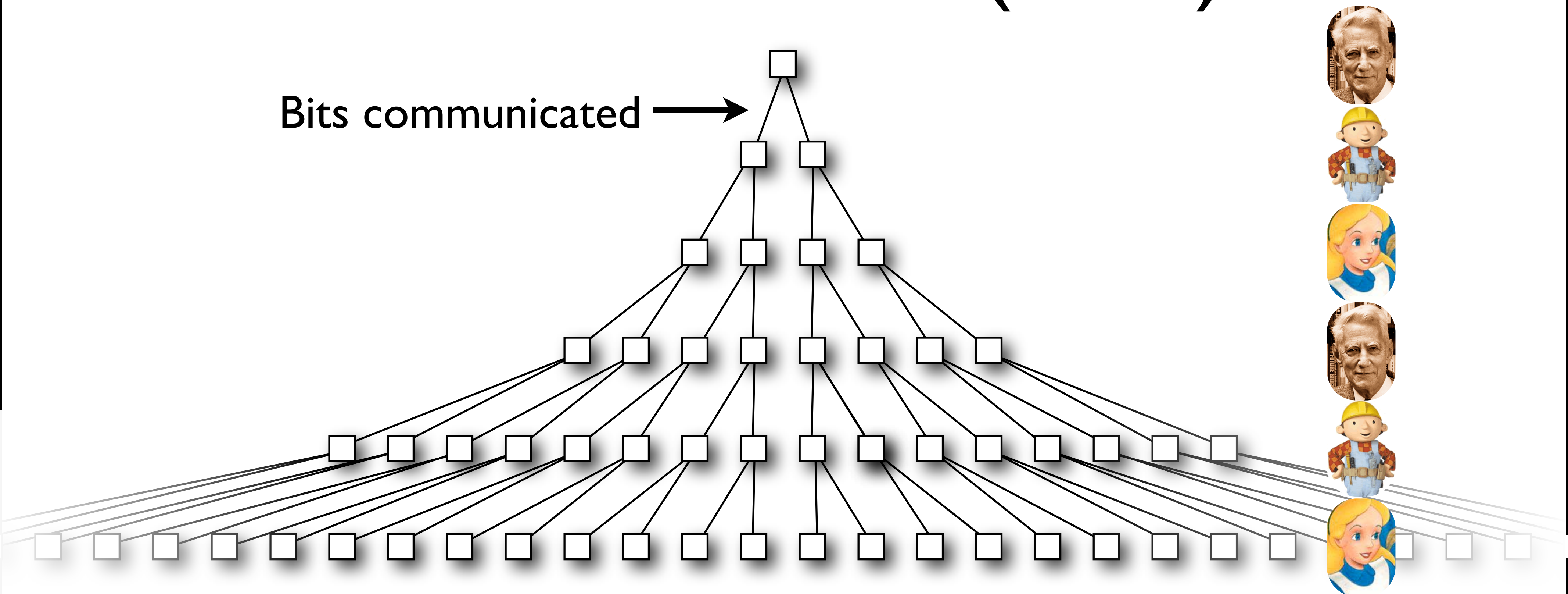
Proof Sketch (*k=3*)



Proof Sketch (k=3)

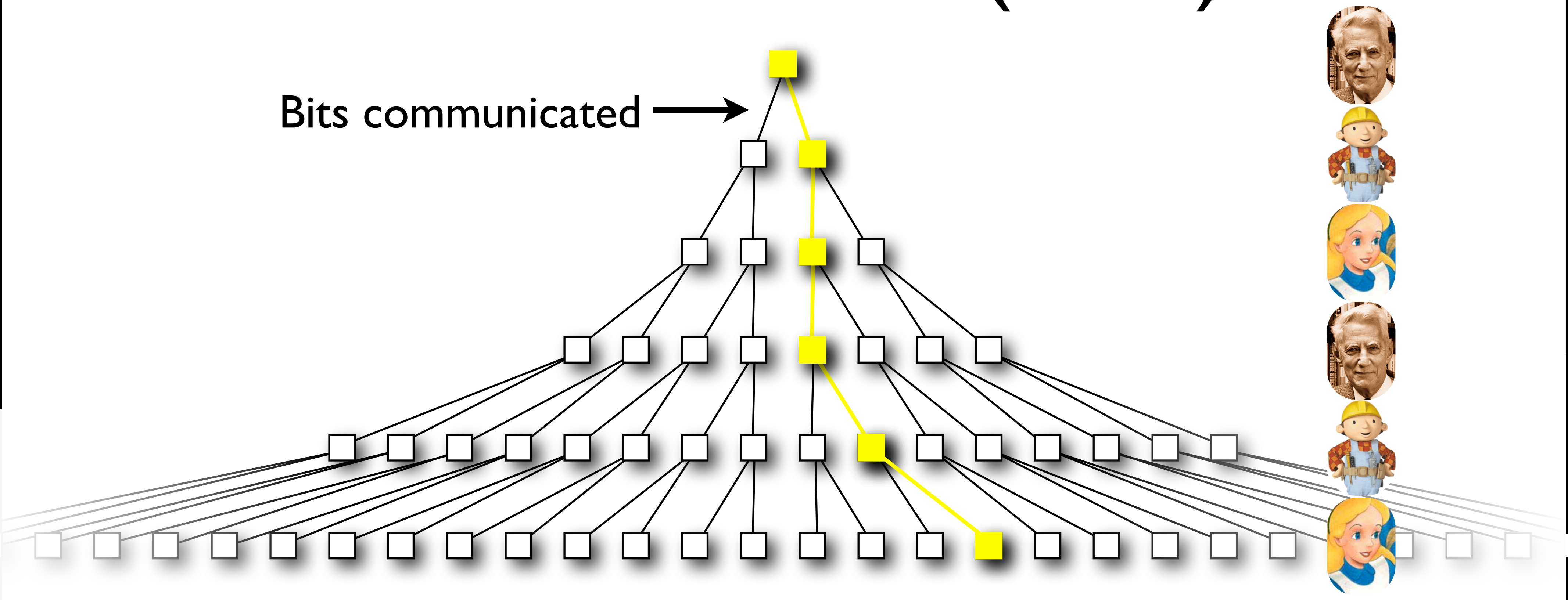


Proof Sketch (*k*=3)



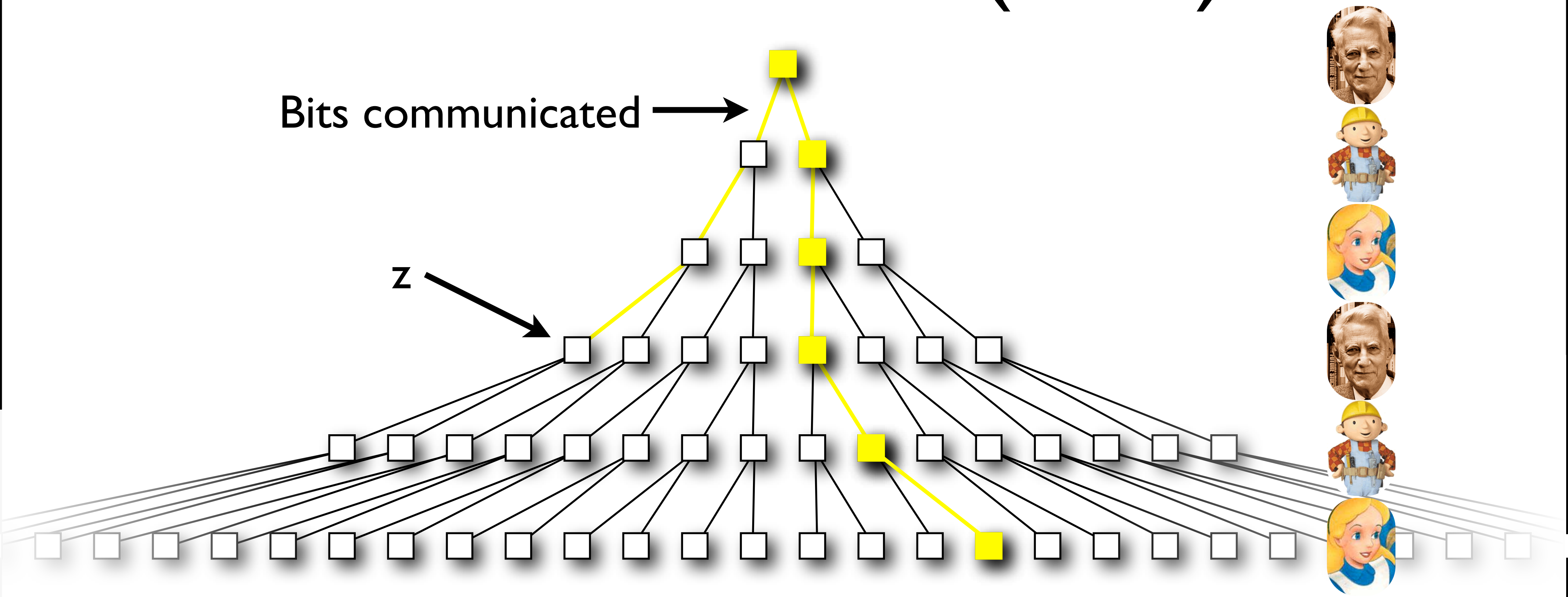
- Consider *deterministic* protocols and *random* f_A, f_B, f_C

Proof Sketch (k=3)



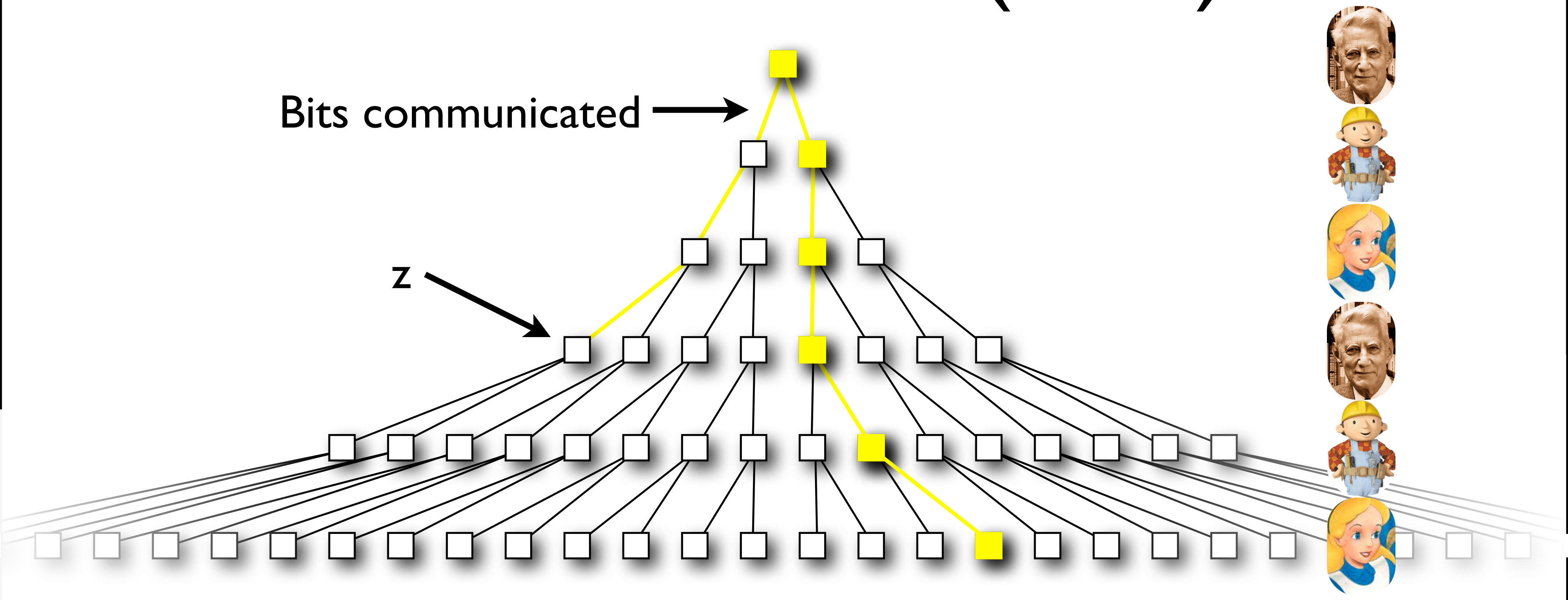
- Consider *deterministic* protocols and *random* f_A, f_B, f_C

Proof Sketch (*k*=3)



- Consider *deterministic* protocols and *random* f_A, f_B, f_C
- Each node *z* defines random variables $z(f_A)$, $z(f_B)$, $z(f_C)$

Proof Sketch (*k*=3)

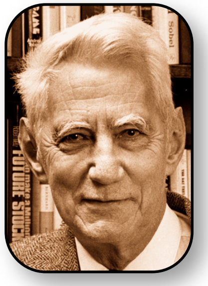


- Consider *deterministic* protocols and *random* f_A, f_B, f_C
- Each node *z* defines random variables $z(f_A)$, $z(f_B)$, $z(f_C)$
- Induction: For each *z*, entropy of variables is high.

Reduction to Selection

f_A: $[t] \to [t]$

f_B: $[t] \to [t]$

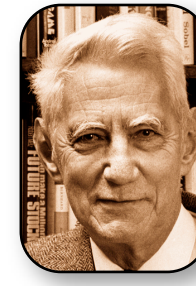


f_C: $[t] \to [t]$

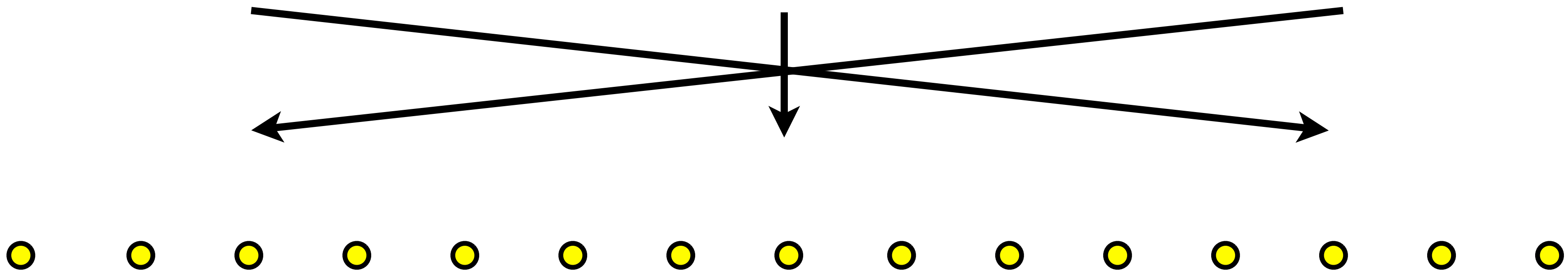
Reduction to Selection

f_A: $[t] \rightarrow [t]$

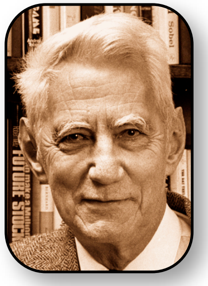
f_B: $[t] \rightarrow [t]$



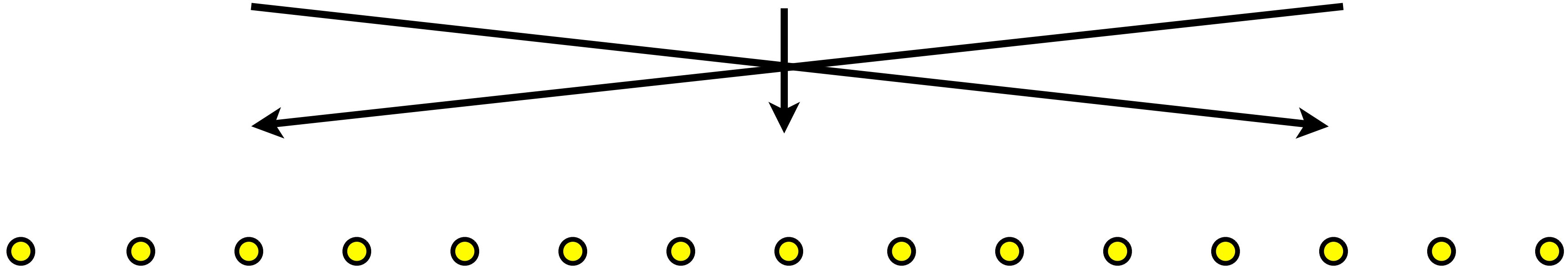
f_C: $[t] \rightarrow [t]$



Reduction to Selection

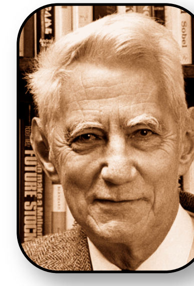


f_A: $[t] \to [t]$ f_B: $[t] \to [t]$ f_C: $[t] \to [t]$



$$\text{Median} = f_A(1)\ f_B(f_A(1))\ f_C(f_B(f_A(1)))$$

Reduction to Selection



$0\ 0\ 0 \times (3 - f_A(1)) \times 5$

$1\ 0\ 0 \times (3 - f_B(1))$

$1\ 1\ f_C(1)$

$1\ 2\ f_C(2)$

$1\ 3\ f_C(3)$

$1\ 4\ 0 \times (f_B(1) - 1)$

$2\ 0\ 0 \times (3 - f_B(2))$

$2\ 1\ f_C(1)$

$2\ 2\ f_C(2)$

$2\ 3\ f_C(3)$

$2\ 4\ 0 \times (f_B(2) - 1)$

$3\ 0\ 0 \times (3 - f_B(3))$

$3\ 1\ f_C(1)$

$3\ 2\ f_C(2)$

$3\ 3\ f_C(3)$

$3\ 4\ 0 \times (f_B(3) - 1)$

$4\ 0\ 0 \times (f_A(1) - 1) \times 5$

VALUE

The Result

The Result

- Setting *t* such that length of stream is *m* and generalizing to approximation gives...

The Result

- Setting *t* such that length of stream is *m* and generalizing to approximation gives...
- *Thm:* Finding an m^{δ}-approximate median in p passes requires $\Omega(m^{(1-\delta)/p}\, p^{-6})$ space.

Summary

Summary

- *Thm:* For a stream in *random order*:

a) 1-pass, O(polylog m)-space, $\tilde{O}(m^{1/2})$-approx

b) O(lg lg m)-pass, O(polylog m)-space exact selection

Summary

- *Thm:* For a stream in *random order*:

  a) 1-pass, O(polylog m)-space, $\tilde{O}(m^{1/2})$-approx

  b) O(lg lg m)-pass, O(polylog m)-space exact selection

- *Thm:* For a stream in *adversarial order*:

  a) 1-pass, $\tilde{O}(m^{1/2})$-approx requires $\Omega(m^{1/2})$ space

  b) O(polylog m)-space exact requires $\Omega(\lg m)$ passes

Summary

- *Thm:* For a stream in *random order*:

  a) 1-pass, O(polylog m)-space, $\tilde{O}(m^{1/2})$-approx

  b) O(lg lg m)-pass, O(polylog m)-space exact selection

- *Thm:* For a stream in *adversarial order*:

  a) 1-pass, $\tilde{O}(m^{1/2})$-approx requires $\Omega(m^{1/2})$ space

  b) O(polylog m)-space exact requires $\Omega(\lg m)$ passes

- *Bonus Thm:* For a stream in *random order*, a single pass, t-approx. requires $\Omega(m^{1/2} t^{-3/2})$ space.

Summary

- *Thm:* For a stream in *random order*:
  a) 1-pass, O(polylog m)-space, $\tilde{O}(m^{1/2})$-approx
  b) O(lg lg m)-pass, O(polylog m)-space exact selection

- *Thm:* For a stream in *adversarial order*:
  a) 1-pass, $\tilde{O}(m^{1/2})$-approx requires $\Omega(m^{1/2})$ space
  b) O(polylog m)-space exact requires $\Omega(\lg m)$ passes

- *Bonus Thm:* For a stream in *random order*, a single pass, t-approx. requires $\Omega(m^{1/2} t^{-3/2})$ space.
- *Future Work:* Multi-pass ROM, trade-offs, *the brave new world of random-order streaming...*