Title: Quantum Binary Search via Adaptive Learning

Date: Jan 09, 2008 04:00 PM

URL: http://pirsa.org/08010006

Abstract: We use a Bayesian approach to optimally solve problems in
noisy binary search. We deal with two variants:
1. Each comparison can be erroneous with some probability 1 - p.
2. At each stage k comparisons can be performed in parallel and
a noisy answer is returned.
We present a (classic) algorithm which optimally solves both variants together, up to an additive term of O(log log (n)), and prove matching information theoretic lower bounds. We use the algorithm with the results of Farhi et al. (FGGS99)presenting a quantum search algorithm in an ordered list of expected complexity less than log(n)/3, and some improved quantum lower bounds on noisy search, and search with an error probability.
Joint work with Michael Ben-Or.

The Bayesian Learner is Optimal for Noisy Binary Search

Michael Ben-Or
Avinatan Hassidim
Hebrew University, Jerusalem Israel

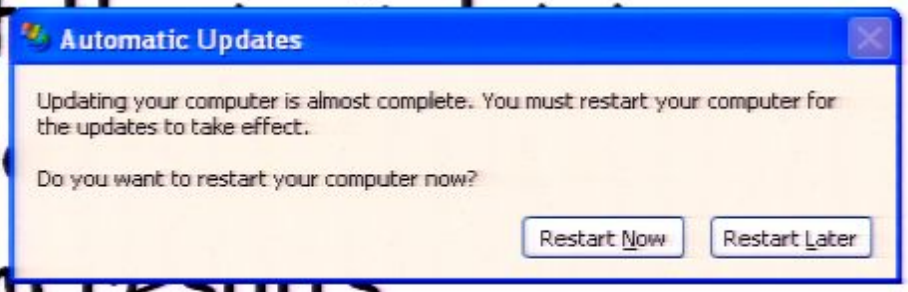
strstr(const char *needle, const char *haystack)

Talk Structure

- A survey of noisy binary search
- Classic algorithm
- Proof of the main lemma
- Lower bounds
- Quantum results

Talk Structure

- A survey of noisy binary search
- Classic algorithm
- Proof of
- Lower b
- Quantum results



Talk Structure

- A survey of noisy binary search
- Classic algorithm
- Proof of the main lemma
- Lower bounds
- Quantum results

Talk Structure

- A survey of noisy binary search
- Classic algorithm
- Proof of the main lemma
- Lower bounds
- Quantum results

Liar Model

- Searching n sorted elements $x_1 \ldots x_n$, using comparisons, some of which are erroneous
  - In one model, the false comparisons can occur at any place
  - In another model, up to a constant fraction of the comparisons made so far can erroneous
- Both models were studied extensively
  - Kleitman, Meyer, Rivest, Spencer, Winklmann [KMRSW80]
  - Borgstrom, Kosaraju [BK93] $O(\log(n)/p^2)$ for $p < 1/2$
  - Pedrotti [Ped99] $(\log(n)/(1+3p)(1-3p^2))\ (8\ln 2/3)$ for $p < 1/3$
  - Pelc [Pel02]

Liar Model

- Searching n sorted elements $x_1 \ldots x_n$, using comparisons, some of which are erroneous
  - In one model, the false comparisons can occur at any place
  - In another model, up to a constant fraction of the comparisons made so far can erroneous
- Both models were studied extensively
  - Kleitman, Meyer, Rivest, Spencer, Winklmann [KMRSW80]
  - Borgstrom, Kosaraju [BK93] $O(\log(n)/p^2)$ for $p < 1/2$
  - Pedrotti [Ped99] $(\log(n)/(1+3p)(1-3p^2))\ (8\ln 2/3)$ for $p < 1/3$
  - Pelc [Pel02]

Liar Model

- Searching n sorted elements $x_1 \ldots x_n$, using comparisons, some of which are erroneous
  - In one model, the false comparisons can occur at any place
  - In another model, up to a constant fraction of the comparisons made so far can erroneous
- Both models were studied extensively
  - Kleitman, Meyer, Rivest, Spencer, Winklmann [KMRSW80]
  - Borgstrom, Kosaraju [BK93] $O(\log(n)/p^2)$ for $p < 1/2$
  - Pedrotti [Ped99] $(\log(n)/(1+3p)(1-3p^2))\ (8\ln 2/3)$ for $p < 1/3$
  - Pelc [Pel02]

Liar Model

- Searching n sorted elements $x_1 \ldots x_n$, using comparisons, some of which are erroneous
  - In one model, the false comparisons can occur at any place
  - In another model, up to a constant fraction of the comparisons made so far can erroneous
- Both models were studied extensively
  - Kleitman, Meyer, Rivest, Spencer, Winklmann [KMRSW80]
  - Borgstrom, Kosaraju [BK93] $O(\log(n)/p^2)$ for $p < 1/2$
  - Pedrotti [Ped99] $(\log(n)/(1+3p)(1-3p^2))\ (8\ln 2/3)$ for $p < 1/3$
  - Pelc [Pel02]

Noisy Model

- Every answer is correct with probability p
  - Intuitive result – log(n)/(1-H(p))

- Feige, Raghavan, Peleg, Upfal [FRPU94] – traveling the search tree

$$O\left(\frac{\log(n)}{(p-0.5)^2}\right)$$

$$I(p)=1-H(p)=1+p\cdot\log(p)+(1-p)\cdot\log(1-p)\approx 0.7(p-0.5)^2$$

- Improvement to the constant factor made by Aslam [Asl95]

Noisy Model

- Every answer is correct with probability p
  - Intuitive result – log(n)/(1-H(p))

- Feige, Raghavan, Peleg, Upfal [FRPU94] – traveling the search tree

$$O\left(\frac{\log(n)}{(p-0.5)^2}\right)$$

$$I(p)=1-H(p)=1+p\cdot\log(p)+(1-p)\cdot\log(1-p)\approx 0.7(p-0.5)^2$$

- Improvement to the constant factor made by Aslam [Asl95]

Noisy Model

- Every answer is correct with probability p
  - Intuitive result – log(n)/(1-H(p))

- Feige, Raghavan, Peleg, Upfal [FRPU94] – traveling the search tree

$$O\left(\frac{\log(n)}{(p-0.5)^2}\right)$$

$$I(p)=1-H(p)=1+p\cdot\log(p)+(1-p)\cdot\log(1-p)\approx 0.7(p-0.5)^2$$

- Improvement to the constant factor made by Aslam [Asl95]

Composite Comparisons

The special element s is compared to a few elements at once

- Input: k indexes of elements $i_1 \ldots i_k$

A single (noisy) segment is returned (i_j, i_{j+1})

$x_{ij} \leq s < x_{ij+1}$

Existing algorithms have suboptimal dependency with respect to k

Composite Comparisons

The special element s is compared to a few elements at once

- Input: k indexes of elements $i_1 \ldots i_k$

A single (noisy) segment is returned (i_j, i_{j+1})

$x_{ij} \leq s < x_{ij+1}$

Existing algorithms have suboptimal dependency with respect to k

Composite Comparisons

The special element s is compared to a few elements at once

- Input: k indexes of elements $i_1 \ldots i_k$

A single (noisy) segment is returned (i_j, i_{j+1})

$x_{ij} \leq s < x_{ij+1}$

Existing algorithms have suboptimal dependency with respect to k

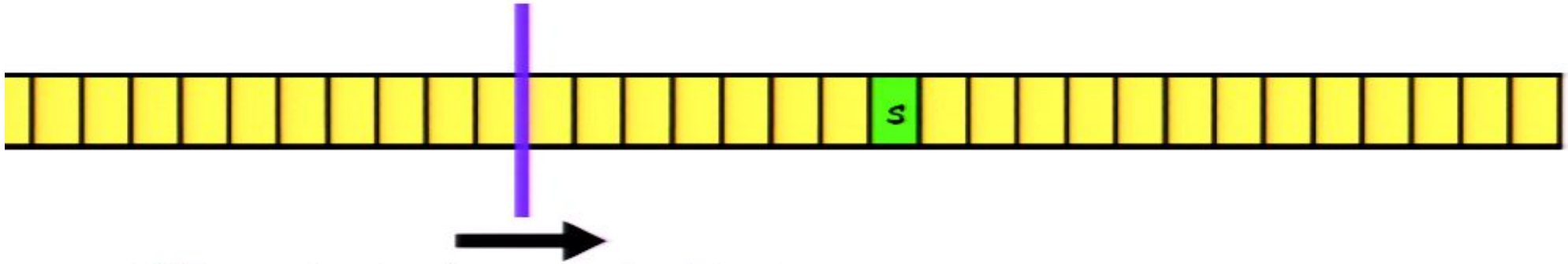
Composite Comparisons

The special element s is compared to a few elements at once

- Input: k indexes of elements $i_1 \ldots i_k$

A single (noisy) segment is returned (i_j, i_{j+1})

$x_{ij} \leq s < x_{ij+1}$



n=31, comparing to x_{11}, output is $s \geq x_{11}$

(correct)

Existing algorithms have suboptimal dependency with respect to k

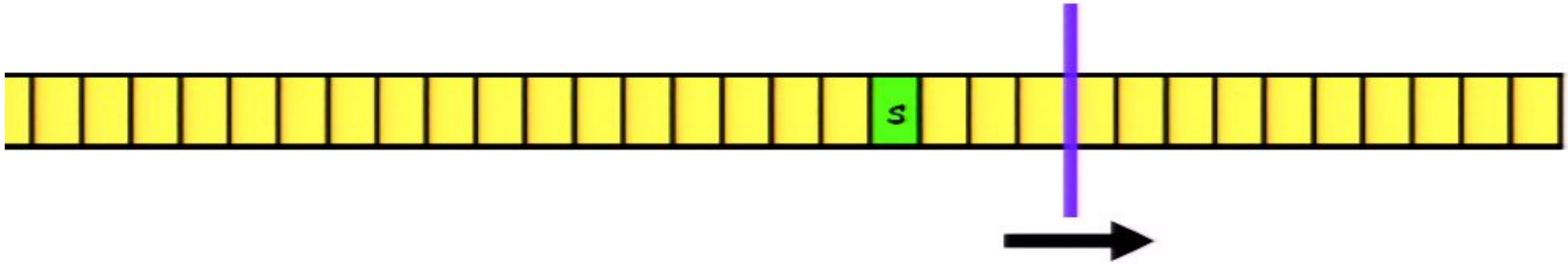
Composite Comparisons

The special element s is compared to a few elements at once

- Input: k indexes of elements $i_1 \ldots i_k$

A single (noisy) segment is returned (i_j, i_{j+1})

$x_{ij} \leq s < x_{ij+1}$



n=31, comparing to x_{23}, output is $s \geq x_{23}$ (mistake!)

Existing algorithms have suboptimal dependency with respect to k

Composite Comparisons

The special element s is compared to a few elements at once

- Input: k indexes of elements $i_1 \ldots i_k$

A single (noisy) segment is returned (i_j, i_{j+1})

$x_{ij} \leq s < x_{ij+1}$

Existing algorithms have suboptimal dependency with respect to k

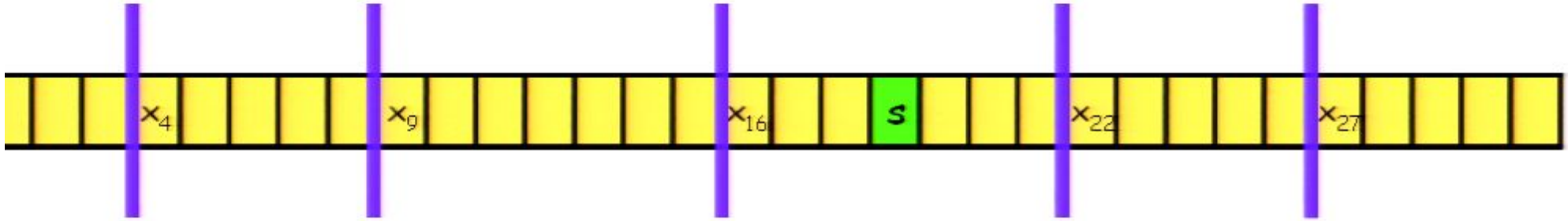
Composite Comparisons

The special element s is compared to a few elements at once

- Input: k indexes of elements $i_1 \ldots i_k$

A single (noisy) segment is returned (i_j, i_{j+1})

$x_{ij} \leq s < x_{ij+1}$



n=31, k=5, input 4,9,16,22,27, output is $x_{16} \leq s < x_{22}$

(correct)

Existing algorithms have suboptimal dependency with respect to k

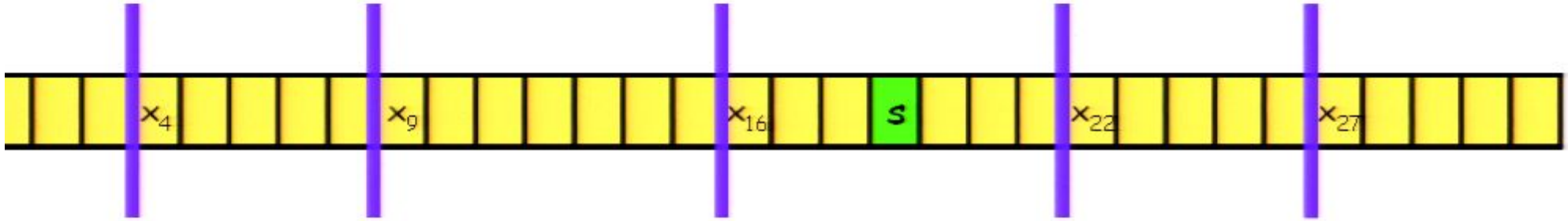
Composite Comparisons

The special element s is compared to a few elements at once

- Input: k indexes of elements $i_1 \ldots i_k$

A single (noisy) segment is returned (i_j, i_{j+1})

$x_{ij} \leq s < x_{ij+1}$



n=31, k=5, input 4,9,16,22,27, output is $x_{16} \leq s < x_{22}$

(correct)

Existing algorithms have suboptimal dependency with respect to k

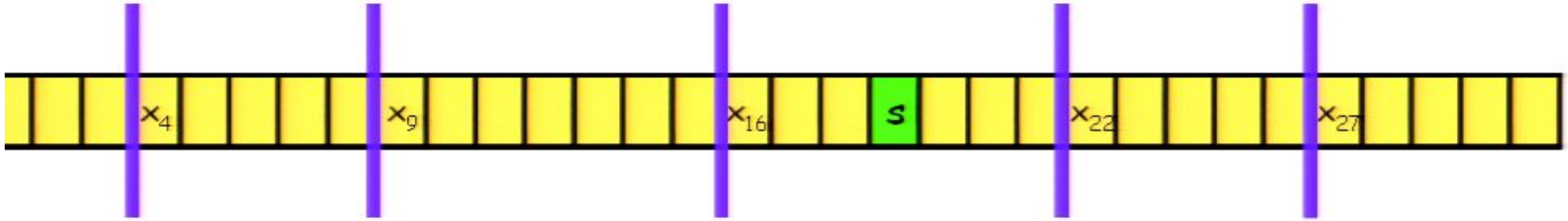
Composite Comparisons

The special element s is compared to a few elements at once

- Input: k indexes of elements $i_1 \ldots i_k$

A single (noisy) segment is returned (i_j, i_{j+1})

$x_{ij} \leq s < x_{ij+1}$



n=31, k=5, input 4,9,16,22,27, output is $x_{16} \leq s < x_{22}$

(correct)

Existing algorithms have suboptimal dependency with respect to k

Our Results

- A tight (up to polyloglog(n) factors) algorithm for the noisy model
- A generalization for the composite query model
- A quantum search algorithm with expected query complexity of 0.31log(n)
- Better quantum lower bounds for noisy search and search with error probability

Our Results - Search

- With success probability $1-\delta$ the right element can be found with an expected number of queries $\frac{\log(n)}{1-H(p)}+O\left(\log\log(n),\log(1/\delta),H(p)\right)$

$$\frac{\log(n)}{1-H(p)}+O\left(\log\log(n)\log(1/\delta)/(1-H(p))\right),\ \delta\geq\log\log^{-3}n$$

- Communication complexity lower bound is

$$\frac{\log(n)}{1-H(p)}-\frac{\log(1/\delta)}{1-H(p)}$$

Our Results - Search

- With success probability $1-\delta$ the right element can be found with an expected number of queries $\frac{\log(n)}{1-H(p)}+O\left(\log\log(n),\log(1/\delta),H(p)\right)$

$$\frac{\log(n)}{1-H(p)}+O\left(\log\log(n)\log(1/\delta)/(1-H(p))\right),\ \delta\geq\log\log^{-3}n$$

- Communication complexity lower bound is

$$\frac{\log(n)}{1-H(p)}-\frac{\log(1/\delta)}{1-H(p)}$$

Our Results - Search

- With success probability $1-\delta$ the right element can be found with an expected number of queries $\frac{\log(n)}{1-H(p)}+O\left(\log\log(n),\log(1/\delta),H(p)\right)$

$$\frac{\log(n)}{1-H(p)}+O\left(\frac{\log\log^4(n)}{1-H(p)}\right)+O\left(\frac{\log(1/\delta)}{1-H(p)}\right),\ \delta<\log\log^{-3}n$$

$$\frac{\log(n)}{1-H(p)}+O\left(\log\log(n)\log(1/\delta)/(1-H(p))\right),\ \delta\geq\log\log^{-3}n$$

- Communication complexity lower bound is

$$\frac{\log(n)}{1-H(p)}-\frac{\log(1/\delta)}{1-H(p)}$$

Composite Comparisons

- Assuming k comparisons at once and a probability for the right answer p

$$\frac{\log(n)}{\log(k)-H(Q)}+O(\log\log(n),H(Q),\log(1/\delta))$$

- I(Q) = log(k) - H(Q) = log(k) - H(p) - (1-p)log(k-1)

$$\frac{\log(n)}{I(Q)}+O\left(\frac{\log\log^4(n)}{I(Q)}\right)+O\left(\frac{\log(1/\delta)}{I(Q)}\right)$$

- Communication complexity lower bound is

$$\frac{\log(n)}{I(Q)}-\frac{\log(1/\delta)}{I(Q)}$$

Our Results - Search

- With success probability $1-\delta$ the right element can be found with an expected number of queries $\frac{\log(n)}{1-H(p)}+O\left(\log\log(n),\log(1/\delta),H(p)\right)$

$$\frac{\log(n)}{1-H(p)}+O\left(\frac{\log\log^4(n)}{1-H(p)}\right)+O\left(\frac{\log(1/\delta)}{1-H(p)}\right),\ \delta<\log\log^{-3}n$$

$$\frac{\log(n)}{1-H(p)}+O\left(\log\log(n)\log(1/\delta)/(1-H(p))\right),\ \delta\geq\log\log^{-3}n$$

- Communication complexity lower bound is

$$\frac{\log(n)}{1-H(p)}-\frac{\log(1/\delta)}{1-H(p)}$$

Composite Comparisons

- Assuming k comparisons at once and a probability for the right answer p

$$\frac{\log(n)}{\log(k)-H(Q)}+O(\log\log(n),H(Q),\log(1/\delta))$$

- I(Q) = log(k) - H(Q) = log(k) - H(p) - (1-p)log(k-1)

$$\frac{\log(n)}{I(Q)}+O\left(\frac{\log\log^4(n)}{I(Q)}\right)+O\left(\frac{\log(1/\delta)}{I(Q)}\right)$$

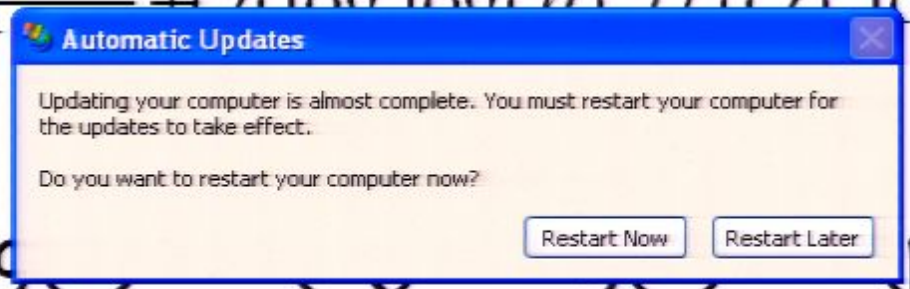
- Communication complexity lower bound is

$$\frac{\log(n)}{I(Q)}-\frac{\log(1/\delta)}{I(Q)}$$

Composite Comparisons

- Assuming k comparisons at once and a probability for the right answer p

$$\frac{\log(n)}{\log(k)-H} + O(\log\log(n)\, H(Q)\, \log(1/\delta))$$



- I(Q) = log(k) - H(p) - (1-p)log(k-1)

$$\frac{\log(n)}{I(Q)} + O\left(\frac{\log\log^4(n)}{I(Q)}\right) + O\left(\frac{\log(1/\delta)}{I(Q)}\right)$$

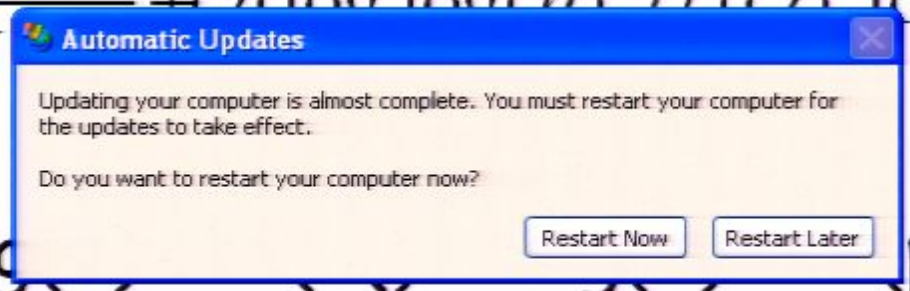
- Communication complexity lower bound is

$$\frac{\log(n)}{I(Q)} - \frac{\log(1/\delta)}{I(Q)}$$

Composite Comparisons

- Assuming k comparisons at once and a probability for the right answer p

$$\frac{\log(n)}{\log(k)-H} + O(\log\log(n)\, H(Q)\, \log(1/\delta))$$



- I(Q) = log(k) - H(p) - (1-p)log(k-1)

$$\frac{\log(n)}{I(Q)} + O\left(\frac{\log\log^4(n)}{I(Q)}\right) + O\left(\frac{\log(1/\delta)}{I(Q)}\right)$$

- Communication complexity lower bound is

$$\frac{\log(n)}{I(Q)} - \frac{\log(1/\delta)}{I(Q)}$$

Composite Comparisons

- Assuming k comparisons at once and a probability for the right answer p

$$\frac{\log(n)}{\log(k)-H(Q)}+O(\log\log(n),H(Q),\log(1/\delta))$$

- I(Q) = log(k) - H(Q) = log(k) - H(p) - (1-p)log(k-1)

$$\frac{\log(n)}{I(Q)}+O\left(\frac{\log\log^4(n)}{I(Q)}\right)+O\left(\frac{\log(1/\delta)}{I(Q)}\right)$$

- Communication complexity lower bound is

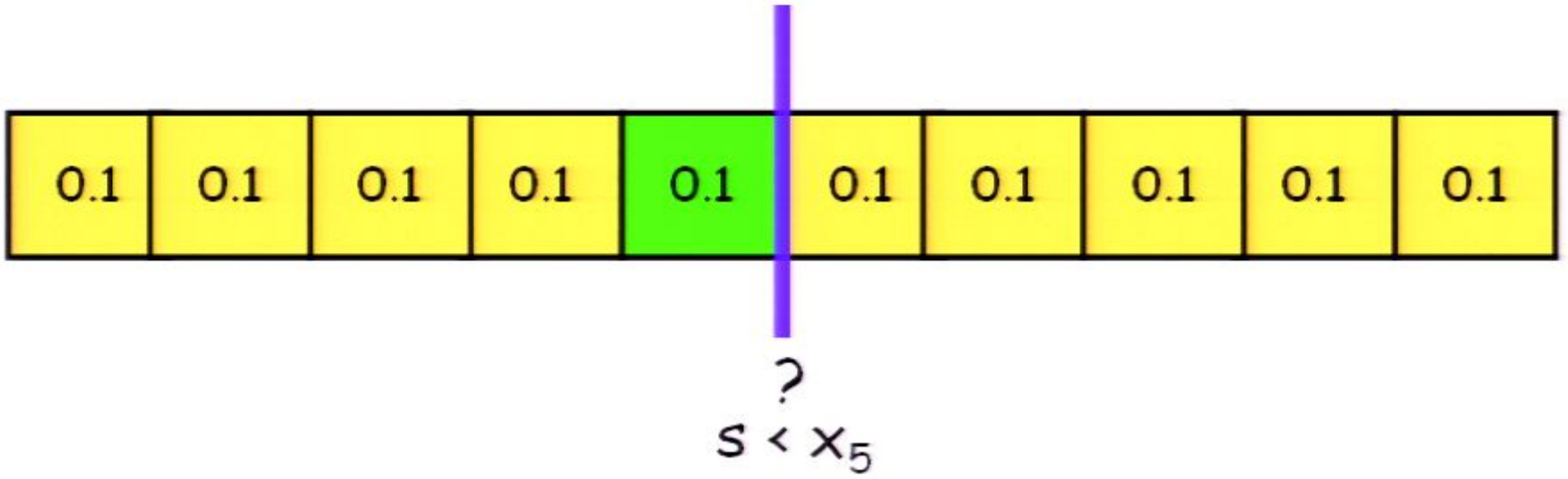
$$\frac{\log(n)}{I(Q)}-\frac{\log(1/\delta)}{I(Q)}$$

Talk Structure

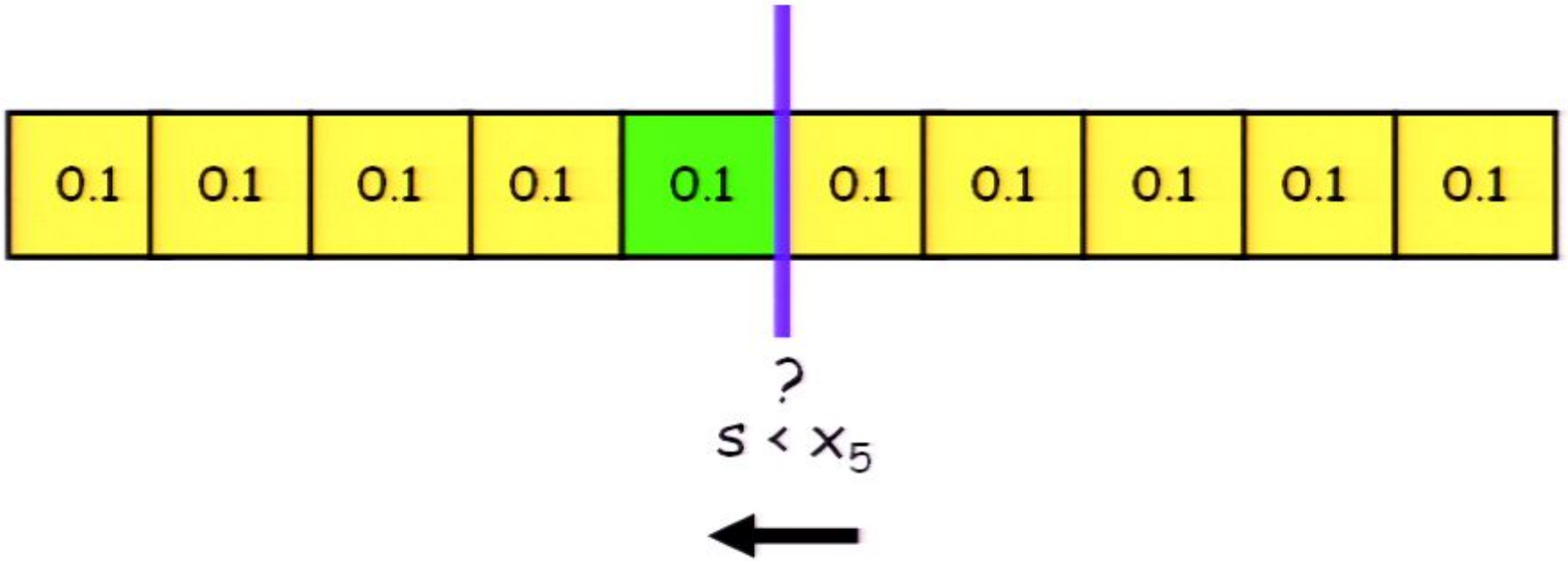
- A survey of noisy binary search
- Classic algorithm
- Proof of the main lemma
- Lower bounds
- Quantum results

Example n=10, p=0.6

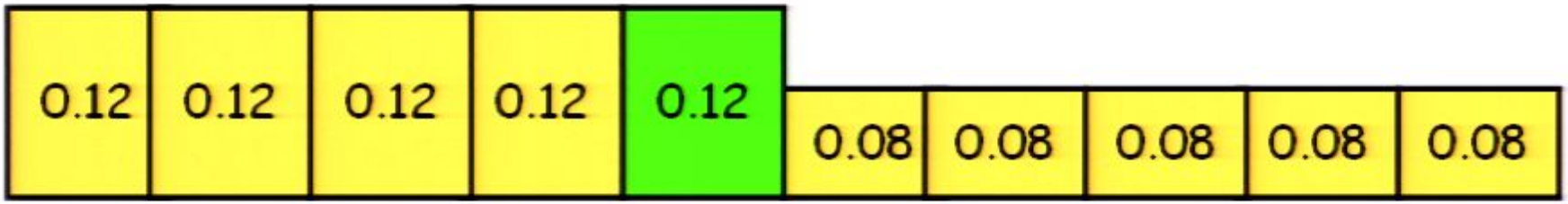
Example n=10, p=0.6



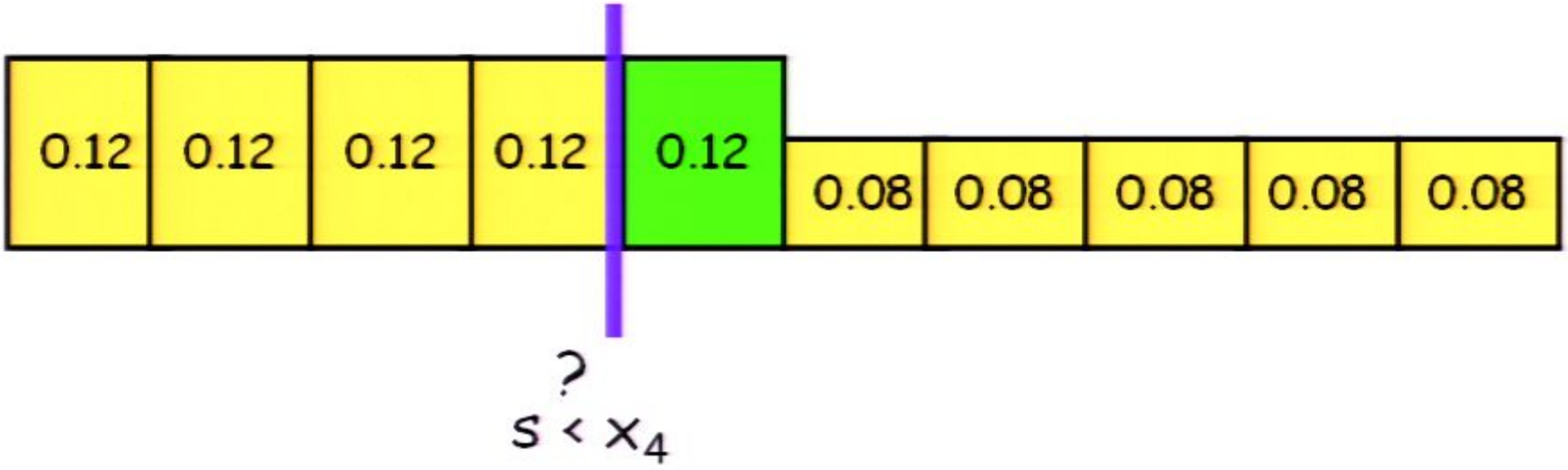
Example n=10, p=0.6



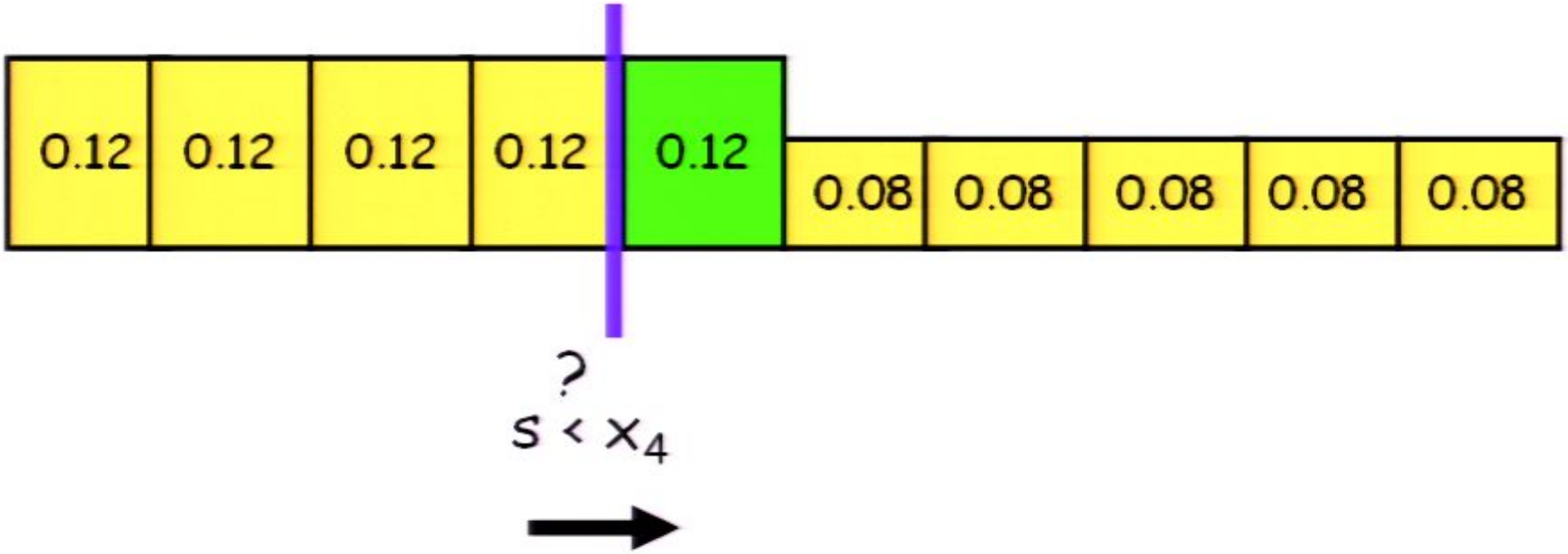
Example n=10, p=0.6



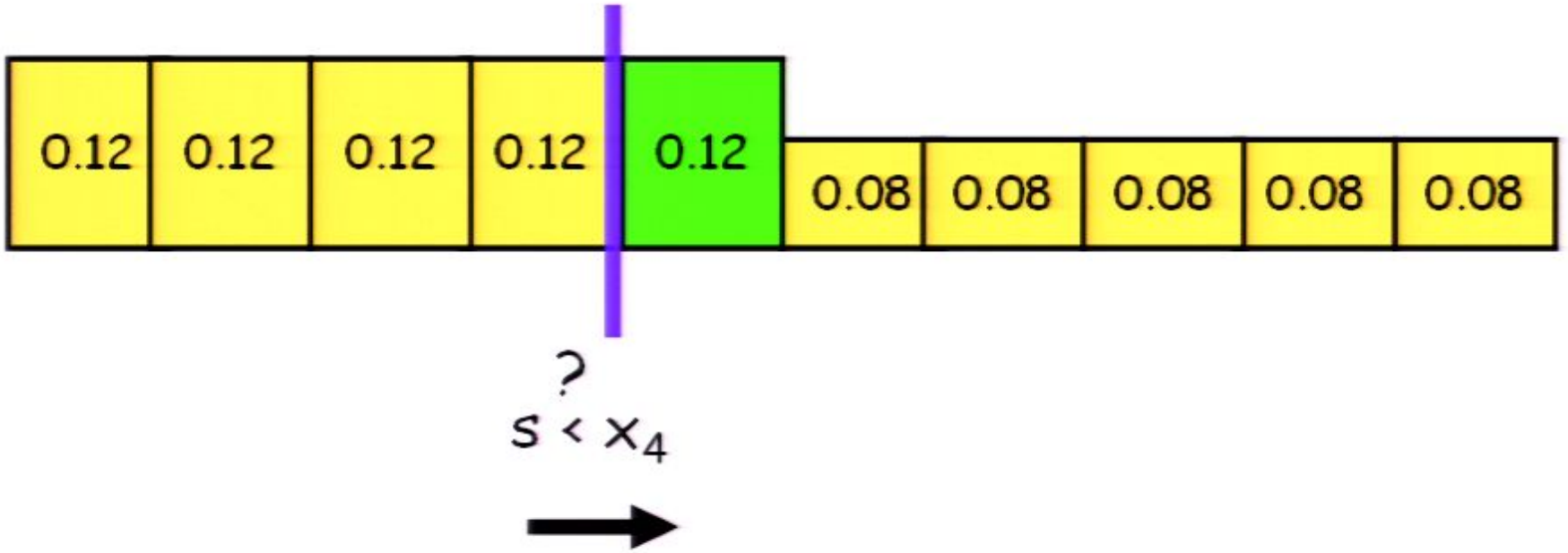
Example n=10, p=0.6



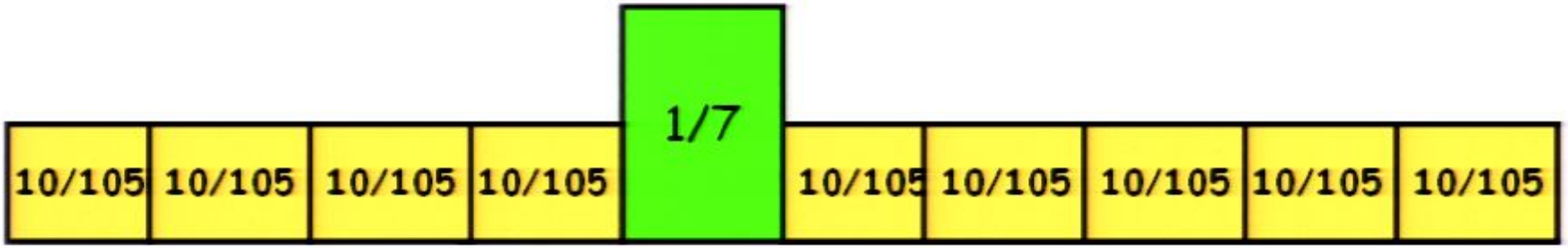
Example n=10, p=0.6



Example n=10, p=0.6



Example n=10, p=0.6



The Algorithm

1. Let $a_1 \ldots a_n$ be an array of probabilities, such that a_i is the probability that $x_i \leq s \leq x_{i+1}$. Initialize by $a_i = 1/n$
2. If $a_i > \varepsilon_{par}$ for some i, search recursively in the vicinity (l_{sur}) of x_i
3. In a manner similar to Bayesian learning, choose a cell k such that the probability that the correct element is larger than x_k is ½

$$\sum_{i=k+1}^{n} a_i \geq 1/2 \qquad \sum_{i=1}^{k} a_i = t \leq 1/2$$

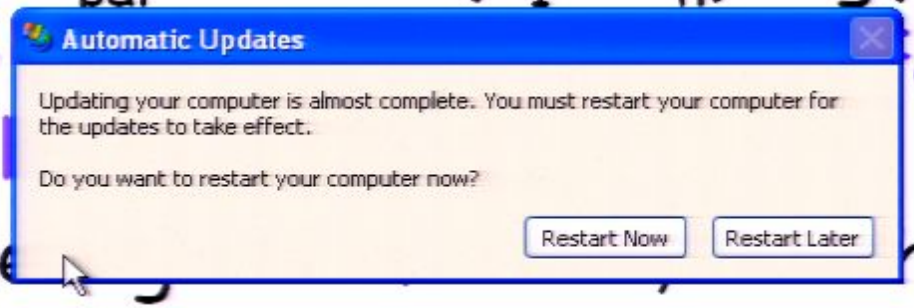
4. Compare to x_k and update all probabilities using Bayes' formula. Return to 2

Why Does this Work?

- Proof follows $H(a_1 \ldots a_n)$, and uses three lemmas
  1. If $\forall i, a_i < \varepsilon_{par}$ then $H(a_1 \ldots a_n) > \log(1/\varepsilon_{par})$
  2. At every stage the expected information rise is (almost) 1-H(p)=I(p)
  3. When the algorithm halts, with high probability $x_{i-1} < s < x_{i+1}$
- Some adjustments are needed if δ is small

Why Does this Work?

- Proof follows $H(a_1 \ldots a_n)$, and uses three lemmas
  1. If $\forall i, a_i < \varepsilon_{par}$ then $H(a_1 \ldots a_n) > \log(1/\varepsilon_{par})$
  2. At every [illegible] formation rise is (a[illegible]
  3. When the [illegible] high probability $x_{i-1} < s < x_{i+1}$
- Some adjustments are needed if δ is small



Why Does this Work?

- Proof follows $H(a_1 \ldots a_n)$, and uses three lemmas
  1. If $\forall i, a_i < \varepsilon_{par}$ then $H(a_1 \ldots a_n) > \log(1/\varepsilon_{par})$
  2. At every stage the expected information rise is (almost) $1-H(p)=I(p)$
  3. When the algorithm halts, with high probability $x_{i-1} < s < x_{i+1}$
- Some adjustments are needed if δ is small

Lemma 1

If $\forall i, a_i < \varepsilon_{par}$ then $H(a_1 \ldots a_n) > \log(1/\varepsilon_{par})$

Proof:

$$H(a_1 \ldots a_n) = \sum -a_i \log(a_i) \geq \sum -a_i \log(\varepsilon_{par})$$
$$= \log(1/\varepsilon_{par}) \sum -a_i = \log(1/\varepsilon_{par})$$

Definitions for lemma 2

- If we compare x_k and s:
  - $x_k > s \rightarrow f(k)=0$
  - $x_k \leq s \rightarrow f(k)=1$
- Remember $a_1 \ldots a_n$ are the probabilities of the elements. Let $b_1 \ldots b_n$ be (the random variables of) the probabilities after the update
- We compare to x_k

$$\sum_{1}^{k} a_i = t, |t - 1/2| < \varepsilon_{par}$$

Lemma 2

The expected information rise at each stage is (1-H(p))(1-1/log(n))

Proof:

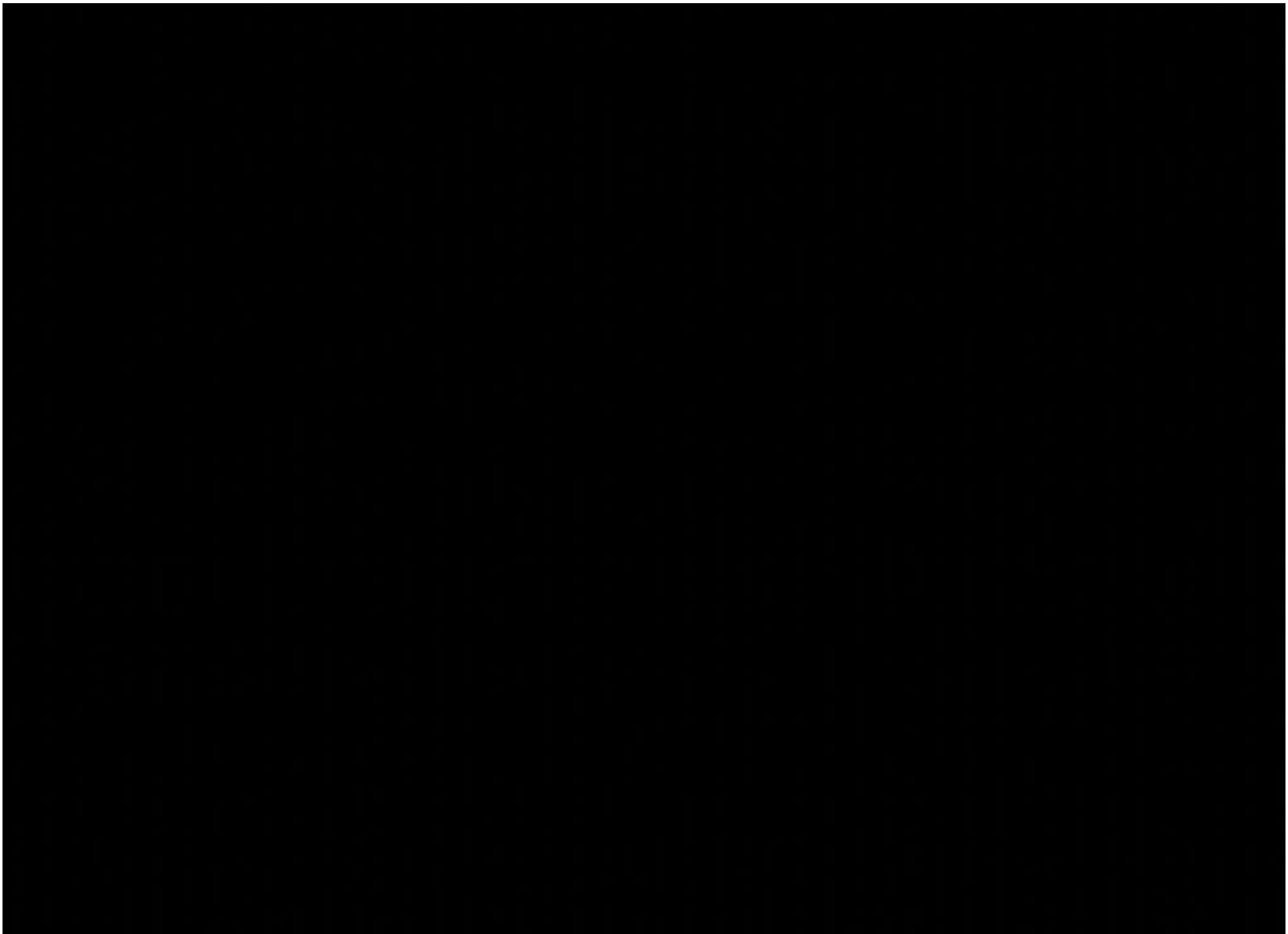
$$H(b_1 \ldots b_n) =$$
$$\Pr(f(k)=0)H(b_1 \ldots b_n \mid f(k)=0) + \Pr(f(k)=1)H(b_1 \ldots b_n \mid f(k)=1) =$$
$$H(a_1 \ldots a_n) - H(pt+(1-p)(1-t)) + H(p)$$

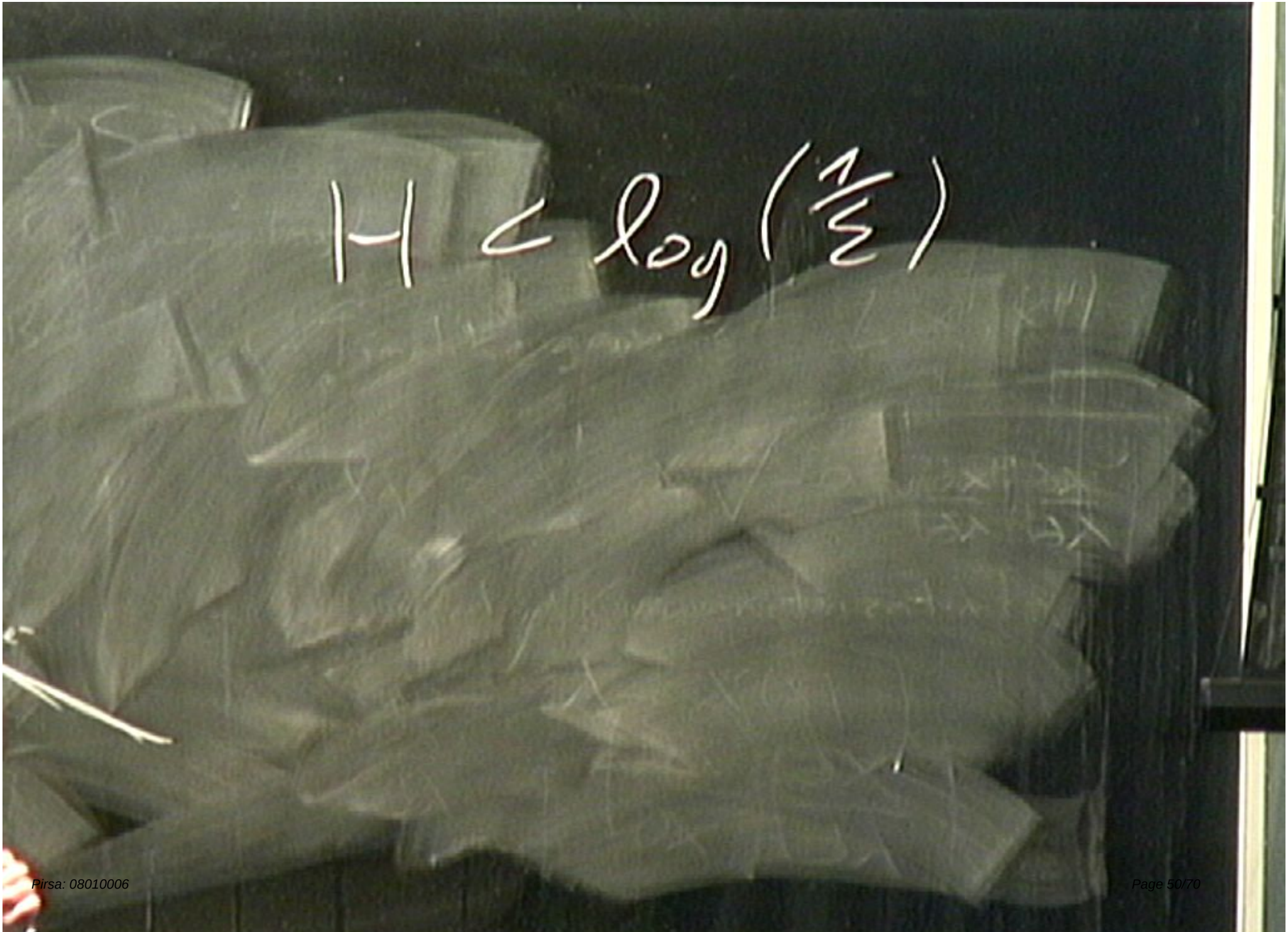
$$|t-1/2| < \varepsilon_{par}$$
$$H(pt+(1-p)(1-t)) > 1-4\varepsilon_{par}^2(1-2p)^2$$

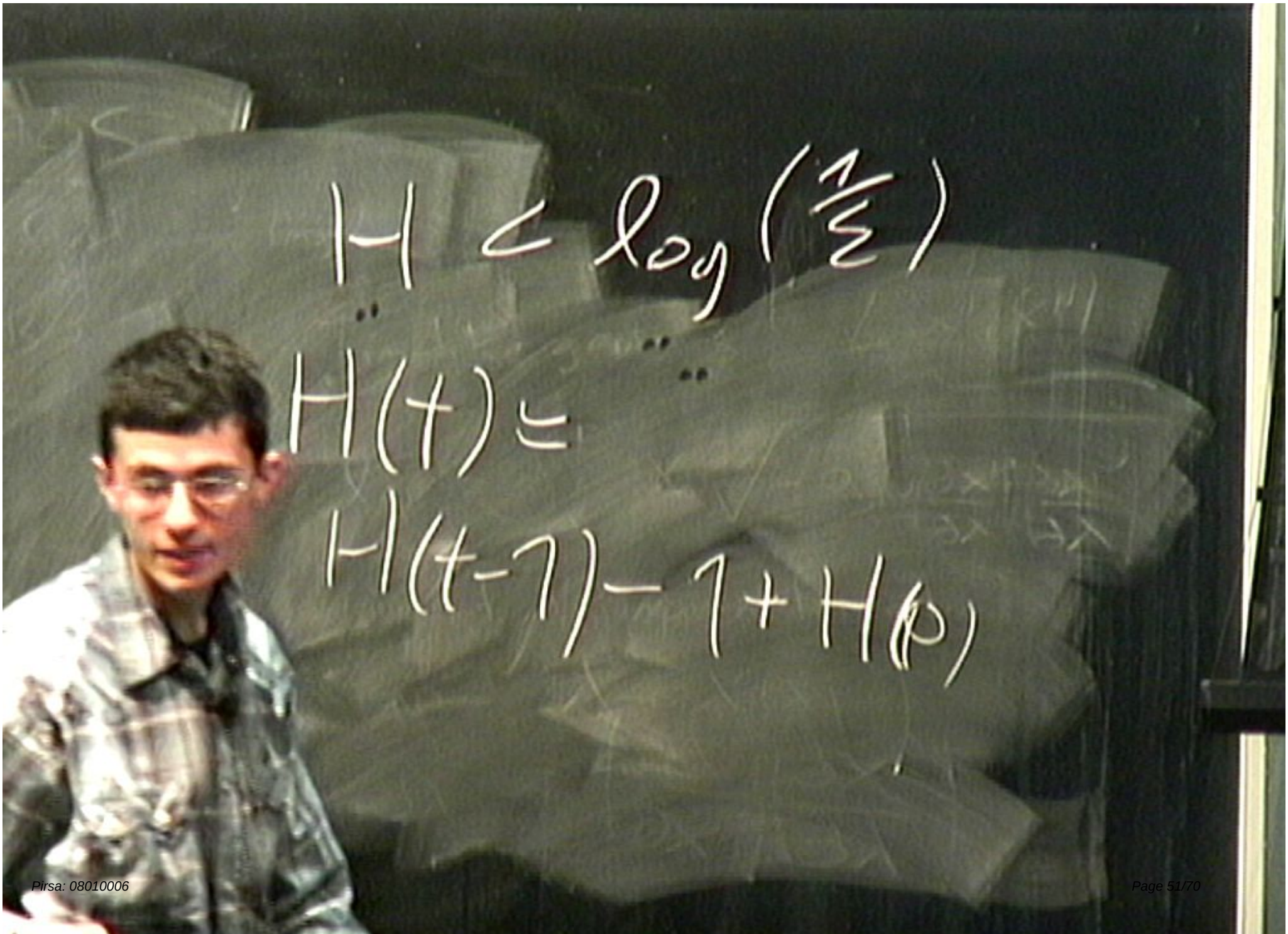
Using a small ε_{par} of $O(1/\log(n))$ gives an average information increase of $I(p)-O(I(p)/\log(n)) \approx 1-H(p)$

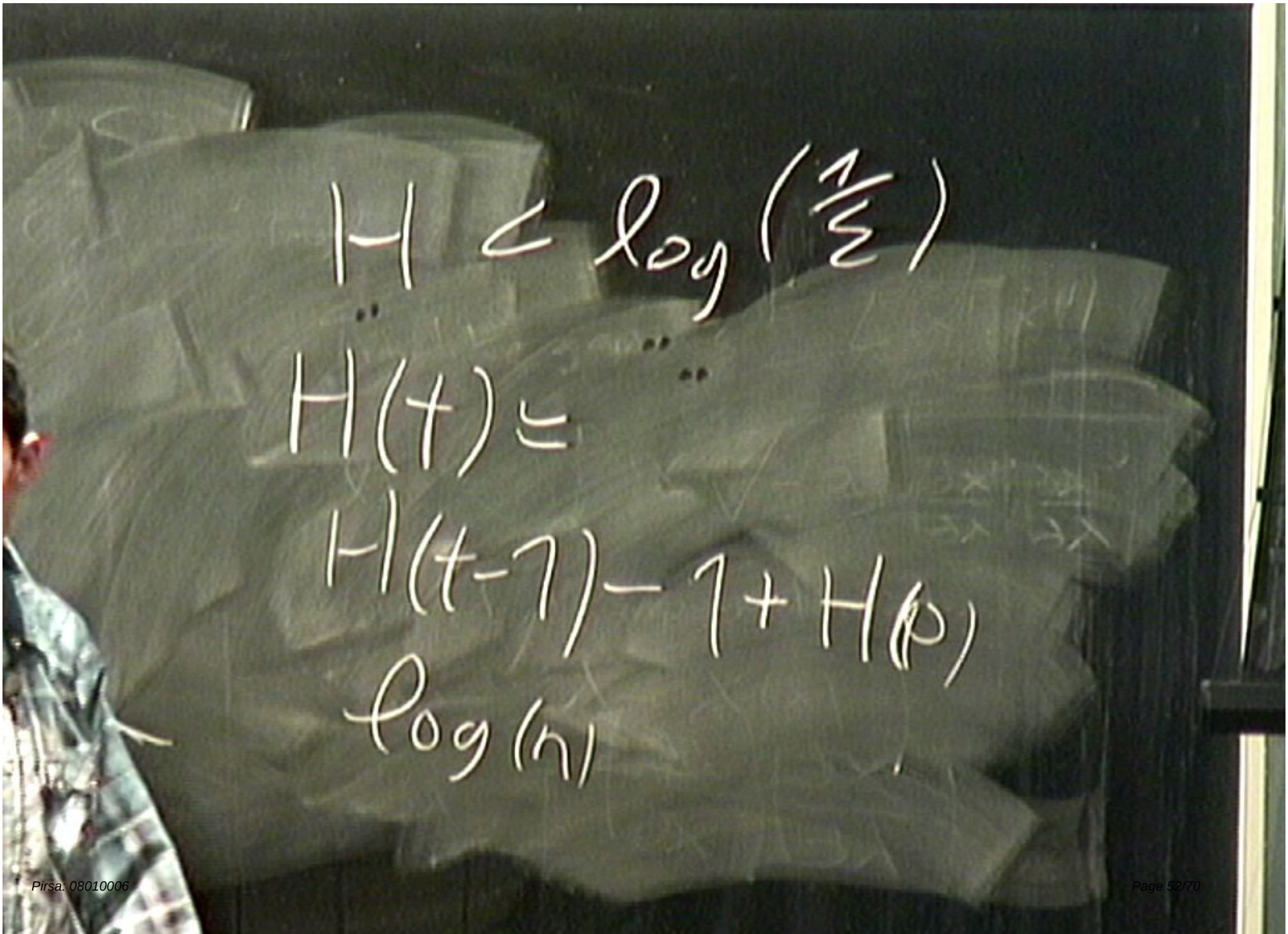
Intuition

Details



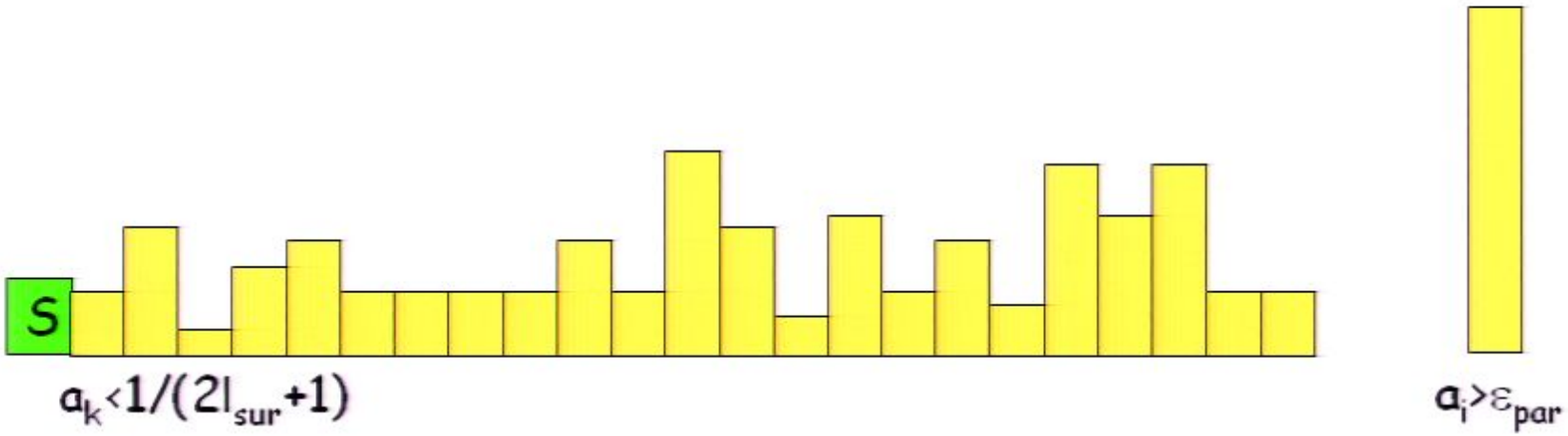

H < log(1/2)


H ≤ log(7/2)
H(t) ≤
H(t-7) - 7 + H(p)


H ≤ log(1/ε)
H(t) ≤
H(t-1) - 1 + H(p)
log(n)

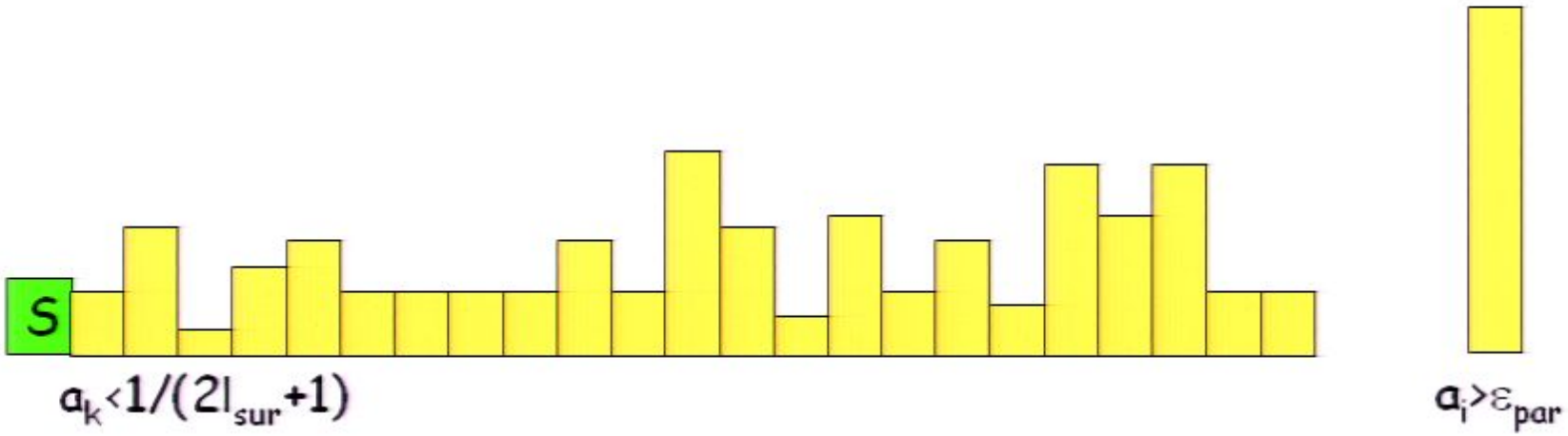
Lemma 3 - Intuition

When the algorithm halts, with high probability the large element is in the vicinity of s



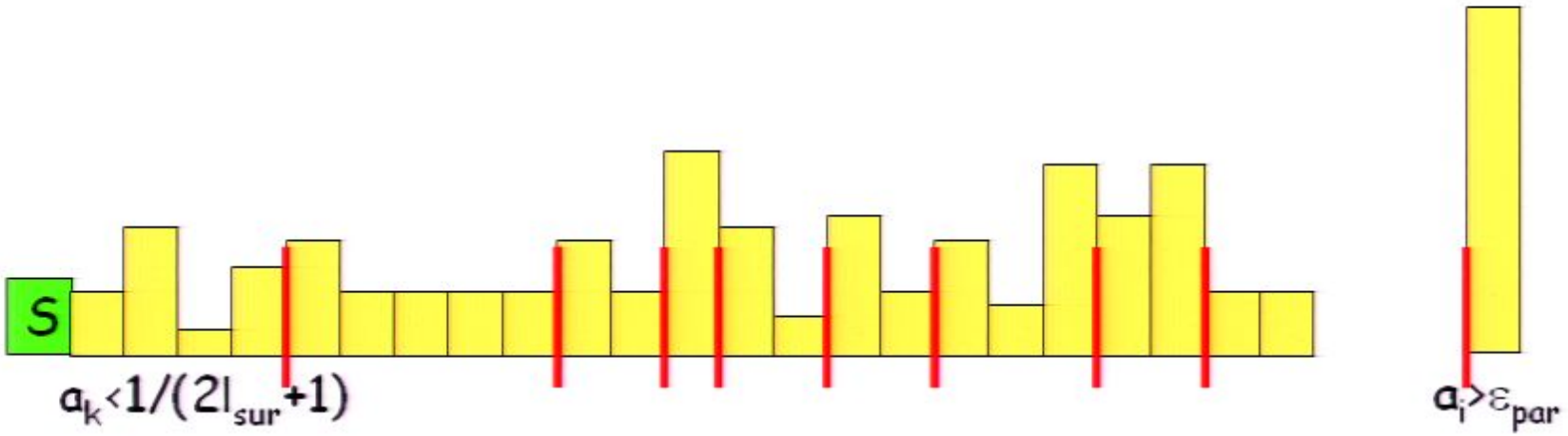
Lemma 3 - Intuition

When the algorithm halts, with high probability the large element is in the vicinity of s



Lemma 3 - Intuition

When the algorithm halts, with high probability the large element is in the vicinity of s

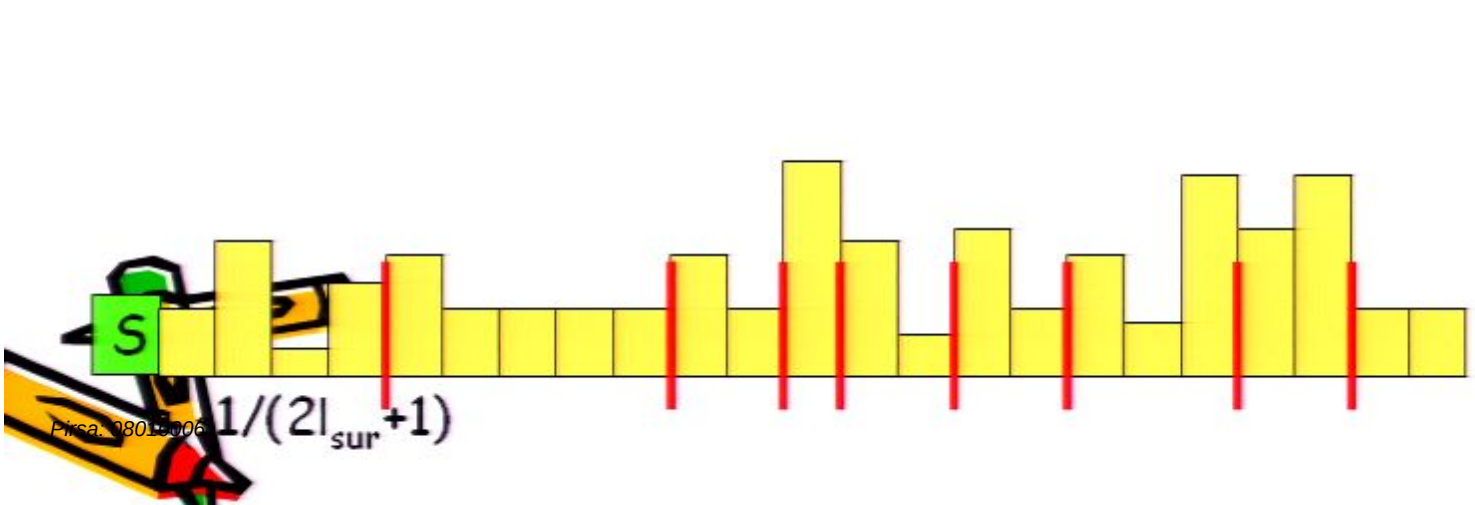


Lemma 3 - Intuition

When the algorithm halts, with high probability the large element is in the vicinity of s

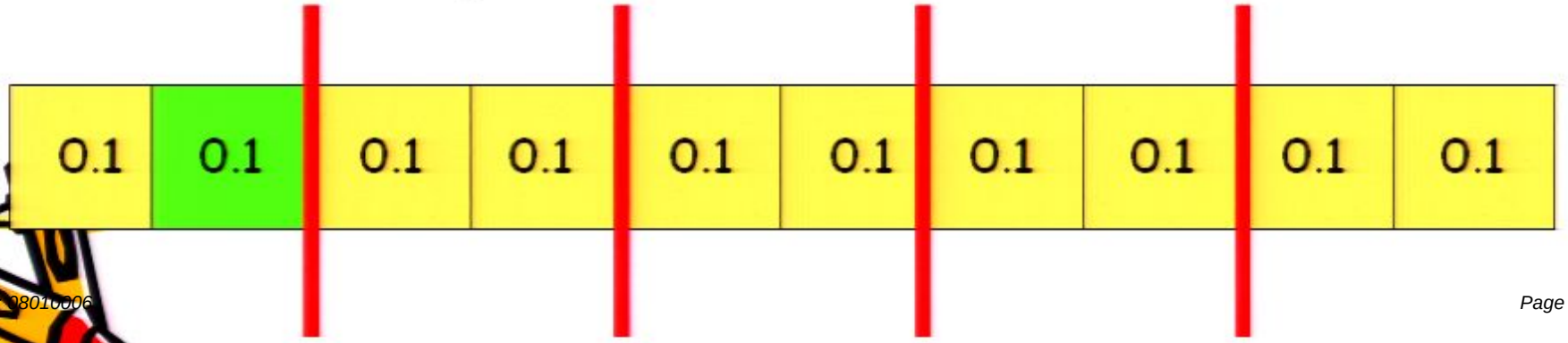
Let $r = \frac{p(1-p)\log^2(1/\delta)}{2p-1}, l_{sur} = \left(\frac{p}{1-p}\right)^r \frac{1}{\varepsilon_{par}}$

Then $\Pr(x_{i-l_{sur}} < s < x_{i+l_{sur}}) \geq 1-\delta$



Generalized Model

- Assume we can query k elements each time, and know where is our element
- Easy generalization – ask about a_0, $a_{n/k}$, $a_{2n/k}$,…,$a_{(k-1)n/k}$
- Can be combined with the noisy model
- Our algorithm deals with this situation optimally (information theoretic)

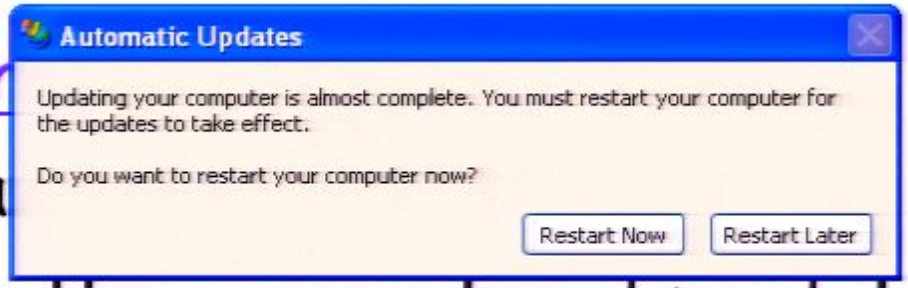


More Details

- Small δ
- Proof of the communication complexity bound
- Implementation
- Getting a flat prior
- Proof for the generalized model
- Discussing the generalized lower bounds
- Bounding the variance of the runtime

More Details

- Small δ
- Proof of the communication complexity bound
- Implemer
- Getting a
- Proof for the generalized model
- Discussing the generalized lower bounds
- Bounding the variance of the runtime



More Details

- Small δ
- Proof of the communication complexity bound
- Implementation
- Getting a flat prior
- Proof for the generalized model
- Discussing the generalized lower bounds
- Bounding the variance of the runtime

Talk Structure

- A survey of noisy binary search
- Classic algorithm
- Proof of the main lemma
- Lower bounds
- Quantum results

Quantum Results

- E. Farhi, J. Goldstone, S. Gutmann and M. Sipser [FGGS99] presented a translationally invariant greedy algorithm which tries to minimize in every query the angle between the state and the required element. The algorithm has good numerical results
- Another exact algorithm by Farhi et al. which finds the correct element out of 52 by 3 queries. Recursive use leads to 0.53log(n)

Quantum Results

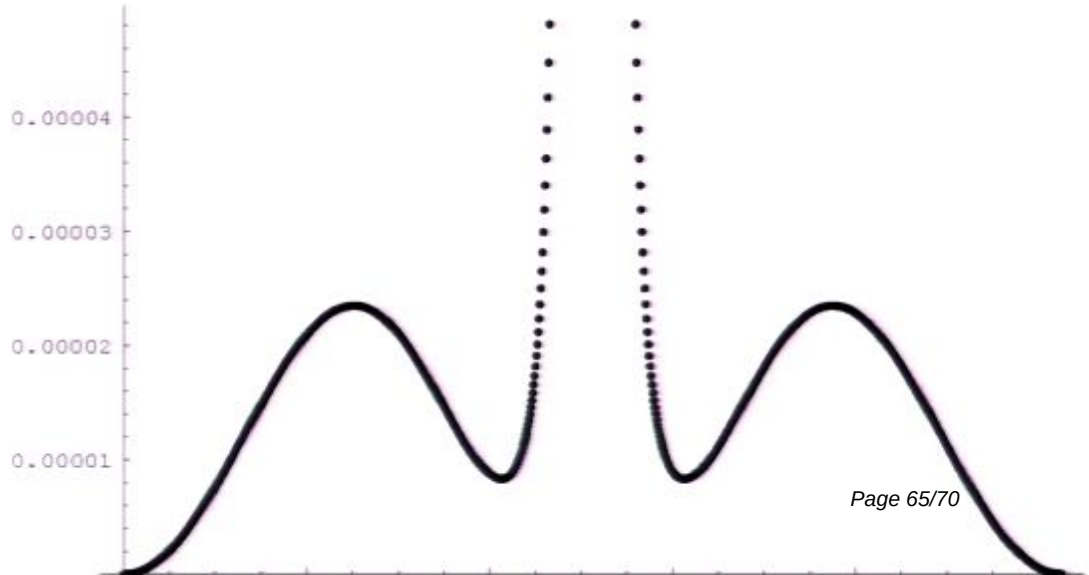
- E. Farhi, J. Goldstone, S. Gutmann and M. Sipser [FGGS99] presented a translationally invariant greedy algorithm which tries to minimize in every query the angle between the state and the required element. The algorithm has good numerical results
- Jacokes, Landahl and Brooks presented an algorithm using 0.44log(n) queries [JLB05]. Childs, Landahl and Parrilo improved this to 0.433log(n), by finding the right element out of 605 using 4 comparisons [CLP07]

Quantum Results

- E. Farhi, J. Goldstone, S. Gutmann and M. Sipser [FGGS99] presented a translationally invariant greedy algorithm which tries to minimize in every query the angle between the state and the required element. The algorithm has good numerical results
- Jacokes, Landahl and Brooks presented an algorithm using 0.44log(n) queries [JLB05]. Childs, Landahl and Parrilo improved this to 0.433log(n), by finding the right element out of 605 using 4 comparisons [CLP07]
- A lower bound of $1/\pi \ln(n) \approx 0.22 \log(n)$ queries by P. Høyer J. Neerbek and Y. Shi [HNS02]

Our Quantum Algorithm

- We use the quantum greedy algorithm by Farhi et al. for searching 2^{26} elements using 7 queries. This gives 22.31 bits of information
  - The probability for success in our subroutine is just 0.6 – so we need a good classical solution
- Combining this with our generalized noisy binary search obtains an algorithm which uses 0.313 log(n) queries



Quantum Lower Bounds

- [HNS02]: Quantum binary search with success probability 1-δ requires

$$\left(1-2\sqrt{\delta(1-\delta)}\right)\frac{1}{\pi}\ln(n)$$

- We prove a matching bound for noisy search

$$\left(1-2\sqrt{\delta(1-\delta)}\right)\frac{1}{\pi}\frac{\ln(n)}{I(P)}$$

- If the algorithm is not noisy, this can be improved to

$$\frac{\ln(2)}{\pi}(1-\delta)\log(n)-O(\delta)$$

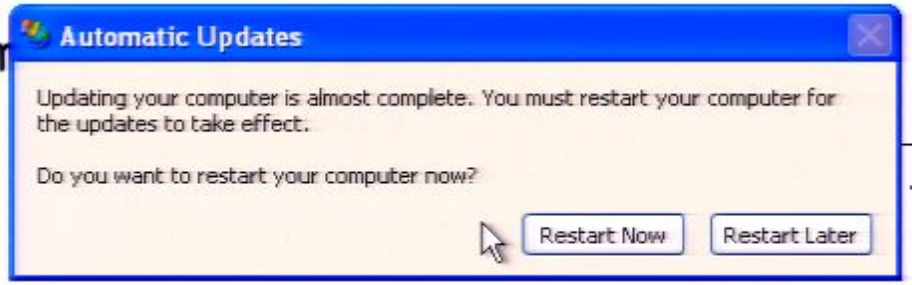
- Which is meaningful also for large δ

Quantum Lower Bounds

- [HNS02]: Quantum binary search with success probability 1-δ requires

$$\left(1-2\sqrt{\delta(1-\delta)}\right)\frac{1}{\pi}\ln(n)$$

- We prove a m



$$-\delta)\right)\frac{1}{\pi}\frac{\ln(n)}{I(P)}$$

- If the algorithm is not noisy, this can be improved to

$$\frac{\ln(2)}{\pi}(1-\delta)\log(n)-O(\delta)$$

- Which is meaningful also for large δ

Quantum Lower Bounds

- [HNS02]: Quantum binary search with success probability 1-δ requires

$$\left(1-2\sqrt{\delta(1-\delta)}\right)\frac{1}{\pi}\ln(n)$$

- We prove a matching bound for noisy search

$$\left(1-2\sqrt{\delta(1-\delta)}\right)\frac{1}{\pi}\frac{\ln(n)}{I(P)}$$

- If the algorithm is not noisy, this can be improved to

$$\frac{\ln(2)}{\pi}(1-\delta)\log(n)-O(\delta)$$

- Which is meaningful also for large δ

Open Questions

- Asymptotic analysis of the greedy algorithm by Farhi et al. Is this better than our algorithm?
- Trying entropy-greedy algorithms with our scheme
- Proving the optimal classical solution to a composite comparison algorithm in which k queries are answered at once
- A. Childs showed that translationally invariant algorithms with error probability less than O(1/N) can be made exact by two more queries. Can this be combined with our results (perhaps by strengthening both results?)

Thank You