Near Linear Time Construction of an Approximate Index for All Maximum Consecutive Sub-Sums of a Sequence

Ferdinando Cicalese Eduardo Laber
Oren Weimann Raphael Yuster

The Maximum Consecutive Subsums Problem (MCSP)

Given a sequence X of n non-negative integers

$$X = x_1\ x_2\ x_3\ \ldots\ x_n$$

let s_ℓ be the maximum subsum of ℓ consecutive elements

$$s_\ell = \max_{j=1\ldots n} (x_j + x_{j+1} + \ldots + x_{j+\ell-1})$$

We want to compute all maximum consecutive subsums: $s_1, s_2, \ldots, s_n$

The Problem: an example

Consider the following sequence of length 8

$X = 2\ 3\ 3\ 2\ 1\ 0\ 4\ 3$

The Problem: an example

Consider the following sequence of length 8

$X = 2\ 3\ 3\ 2\ 1\ 0\ 4\ 3$

The Problem: an example

Consider the following sequence of length 8

$X = 2\ 3\ 3\ 2\ 1\ 0\ 4\ 3$

The Problem: an example

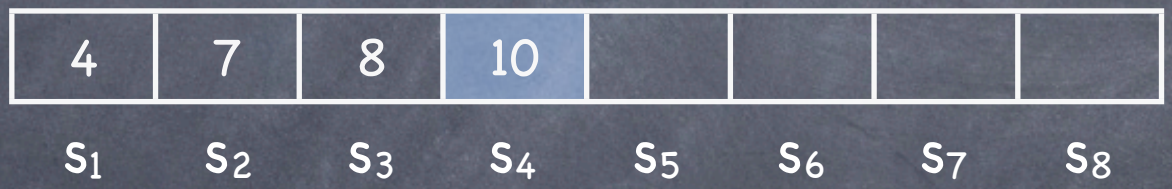
Consider the following sequence of length 8

$X = 2\ 3\ 3\ 2\ 1\ 0\ 4\ 3$

The Problem: an example

Consider the following sequence of length 8

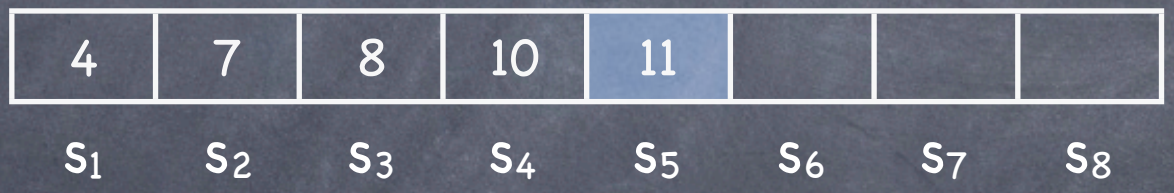
$X = 2\ 3\ 3\ 2\ 1\ 0\ 4\ 3$



The Problem: an example

Consider the following sequence of length 8

$X = 2\ 3\ 3\ 2\ 1\ 0\ 4\ 3$



The Problem: an example

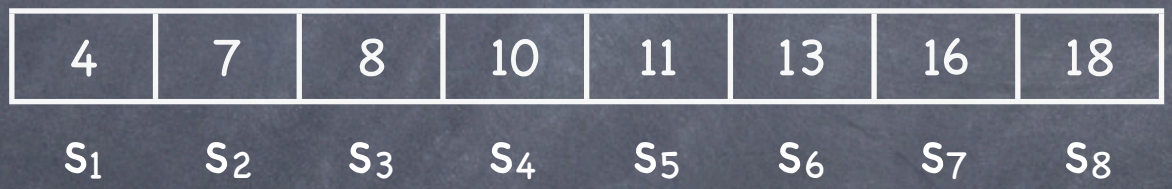
Consider the following sequence of length 8

$X = 2\ 3\ 3\ 2\ 1\ 0\ 4\ 3$

The Problem: an example

Consider the following sequence of length 8

$X = 2\ 3\ 3\ 2\ 1\ 0\ 4\ 3$



The Problem: an example

Consider the following sequence of length 8

$X = 2\ 3\ 3\ 2\ 1\ 0\ 4\ 3$

4	7	8	10	11	13	16	18
s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8

The table can be trivially computed in $O(n^2)$ time

each maximum subsum can be computed in $O(n)$ by a linear scan

State of the Art

The Maximum Consecutive Subsums Problem

Given a sequence **X** of n non-negative integers $\mathbf{X} = x_1\ x_2\ x_3\ \ldots\ x_n$

compute the table of all maximum consecutive subsums $s_1, s_2, \ldots, s_n$

The best known constructions are $\Theta(n^2/\text{polylog } n)$

- $O(n^2/\log n)$ via (min,+)-convolution
- $O(n^2/\log^2 n)$ in the word RAM model

State of the Art

The Maximum Consecutive Subsums Problem

Given a sequence **X** of n non-negative integers $\mathbf{X} = x_1\ x_2\ x_3\ \ldots\ x_n$

compute the table of all maximum consecutive subsums $s_1, s_2, \ldots, s_n$

The best known constructions are $\Theta(n^2/\text{polylog } n)$

- $O(n^2/\log n)$ via (min,+)-convolution
- $O(n^2/\log^2 n)$ in the word RAM model

The only known lower bound is $\Omega(n)$

State of the Art

The Maximum Consecutive Subsums Problem

Given a sequence X of n non-negative integers $X = x_1\ x_2\ x_3\ \ldots\ x_n$
compute the table of all maximum consecutive subsums $s_1, s_2, \ldots, s_n$

The best known constructions are $\Theta(n^2/\text{polylog } n)$

- $O(n^2/\log n)$ via (min,+)-convolution
- $O(n^2/\log^2 n)$ in the word RAM model

The only known lower bound is $\Omega(n)$

Can we close this gap?

Our Result: a nearly linear approximation

The Maximum Consecutive Subsums Problem

Given a sequence **X** of n non-negative integers $\mathbf{X} = x_1\ x_2\ x_3\ \ldots\ x_n$
compute the table of all maximum consecutive subsums $s_1, s_2, \ldots, s_n$

Theorem
For any $\varepsilon, \eta > 0$, we can compute in time $O(k\ n^{1+\eta})$ values $\hat{s}_\ell$ $(\ell=1,\ldots, n)$ such that $s_\ell \le \hat{s}_\ell \le (1+\varepsilon)\ s_\ell$ where k depends only on ε and η.

The Basic Idea:
A golden ratio approx in $O(n^{3/2})$ time

We compute a table of Approximate Subsums $\hat{s}_\ell$ (ℓ=1,..., n) such that

$$1 \le \hat{s}_\ell / s_\ell \le \alpha = (1+\sqrt{5})/2$$

The Basic Idea:
A golden ratio approx in $O(n^{3/2})$ time

We compute a table of Approximate Subsums $\hat{s}_\ell$ (ℓ=1,..., n) such that

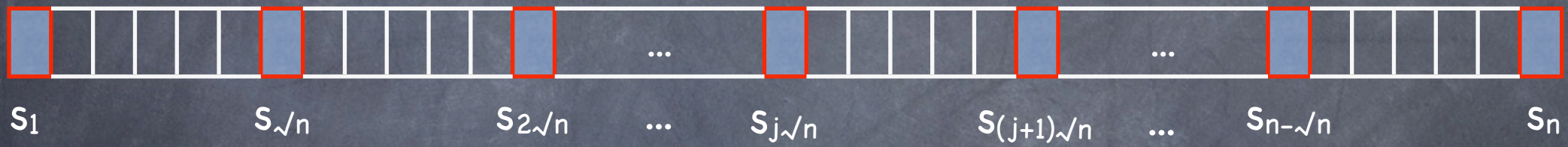
$$1 \le \hat{s}_\ell / s_\ell \le \alpha = (1+\sqrt{5})/2$$

We partition the Subsum table into $\sqrt{n}$ intervals

The Basic Idea:
A golden ratio approx in $O(n^{3/2})$ time

We compute a table of Approximate Subsums $\hat{s}_\ell$ (ℓ=1,..., n) such that

$$1 \le \hat{s}_\ell / s_\ell \le \alpha = (1+\sqrt{5})/2$$



We partition the Subsum table into $\sqrt{n}$ intervals

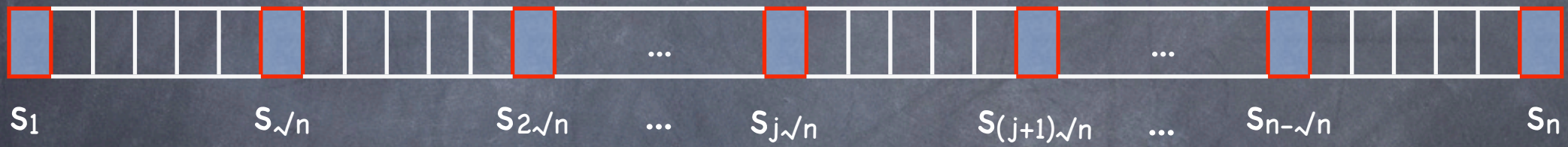
We compute exactly the values s_ℓ at the extremes of each interval

($\ell = \sqrt{n}, 2\sqrt{n}, 3\sqrt{n}, \ldots, n$)

The Basic Idea:
A golden ratio approx in $O(n^{3/2})$ time

We compute a table of Approximate Subsums $\hat{s}_\ell$ (ℓ=1,..., n) such that

$$1 \le \hat{s}_\ell / s_\ell \le \alpha = (1+\sqrt{5})/2$$



We partition the Subsum table into $\sqrt{n}$ intervals

We compute exactly the values s_ℓ at the extremes of each interval

($\ell = \sqrt{n}, 2\sqrt{n}, 3\sqrt{n}, \ldots, n$)

Time = $O(n \times \sqrt{n})$

The Basic Idea:
A golden ratio approx in $O(n^{3/2})$ time

We compute a table of Approximate Subsums $\hat{s}_\ell$ (ℓ=1,..., n) such that

$$1 \le \hat{s}_\ell / s_\ell \le \alpha = (1+\sqrt{5})/2$$

The Basic Idea:
A golden ratio approx in $O(n^{3/2})$ time

We compute a table of Approximate Subsums $\hat{s}_\ell$ (ℓ=1,..., n) such that

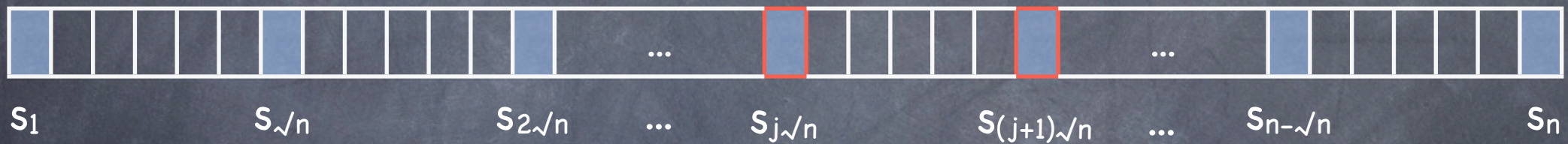
$$1 \le \hat{s}_\ell / s_\ell \le \alpha = (1+\sqrt{5})/2$$

For filling the remaining entries:

The Basic Idea:
A golden ratio approx in $O(n^{3/2})$ time

We compute a table of Approximate Subsums $\hat{s}_\ell$ (ℓ=1,..., n) such that

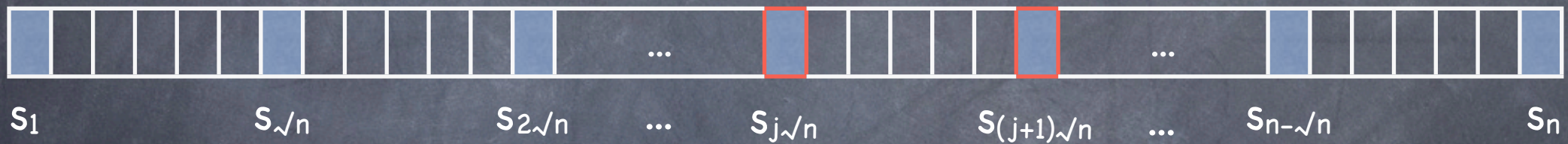
$1 \le \hat{s}_\ell / s_\ell \le \alpha = (1+\sqrt{5})/2$



The Basic Idea:
A golden ratio approx in $O(n^{3/2})$ time

We compute a table of Approximate Subsums $\hat{s}_\ell$ (ℓ=1,..., n) such that

$$1 \le \hat{s}_\ell / s_\ell \le \alpha = (1+\sqrt{5})/2$$

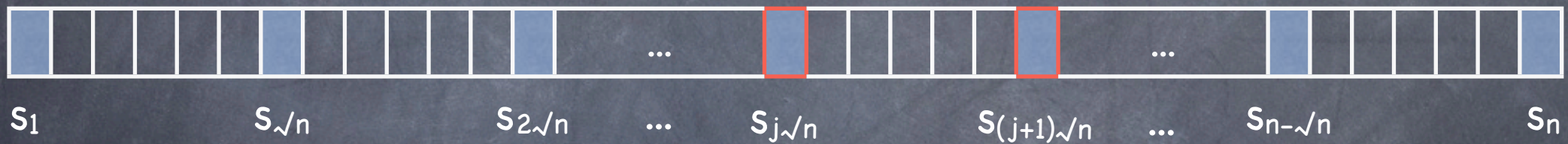


For filling the remaining entries:

The Basic Idea:
A golden ratio approx in $O(n^{3/2})$ time

We compute a table of Approximate Subsums $\hat{s}_\ell$ (ℓ=1,..., n) such that

$$1 \le \hat{s}_\ell / s_\ell \le \alpha = (1+\sqrt{5})/2$$



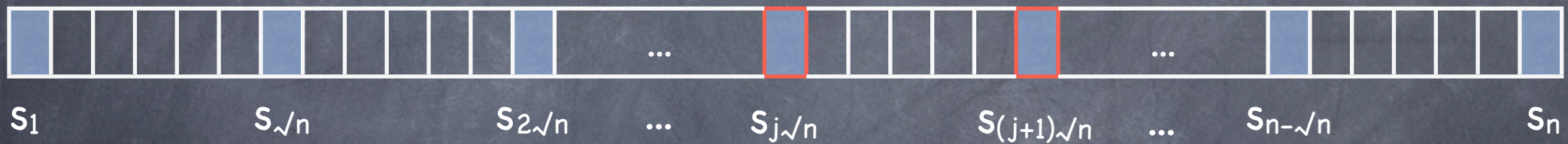
For filling the remaining entries:

1. If $s_{(j+1)\sqrt{n}} / s_{j\sqrt{n}} \le \alpha$

The Basic Idea:
A golden ratio approx in $O(n^{3/2})$ time

We compute a table of Approximate Subsums $\hat{s}_\ell$ (ℓ=1,..., n) such that

$$1 \le \hat{s}_\ell / s_\ell \le \alpha = (1+\sqrt{5})/2$$



For filling the remaining entries:

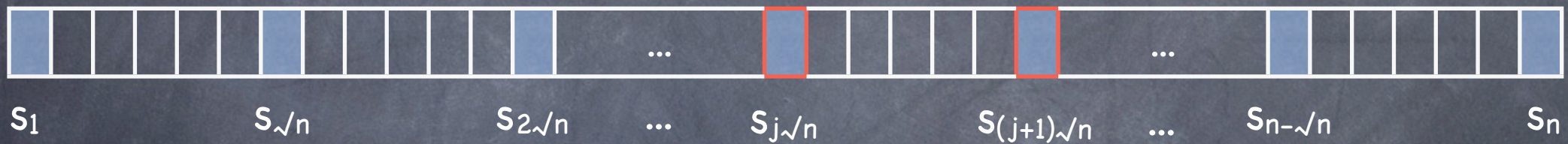
1. If $s_{(j+1)\sqrt{n}} / s_{j\sqrt{n}} \le \alpha$

   set $\hat{s}_\ell = s_{(j+1)\sqrt{n}}$ for each ℓ in the interval

The Basic Idea:
A golden ratio approx in $O(n^{3/2})$ time

We compute a table of Approximate Subsums $\hat{s}_\ell$ (ℓ=1,..., n) such that

$$1 \le \hat{s}_\ell / s_\ell \le \alpha = (1+\sqrt{5})/2$$



For filling the remaining entries:

1. If $s_{(j+1)\sqrt{n}} / s_{j\sqrt{n}} \le \alpha$

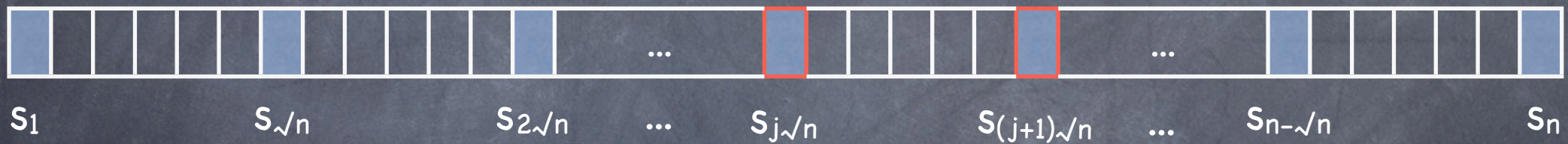
   set $\hat{s}_\ell = s_{(j+1)\sqrt{n}}$ for each ℓ in the interval

   therefore $\hat{s}_\ell / s_\ell = s_{(j+1)\sqrt{n}} / s_\ell \le s_{(j+1)\sqrt{n}} / s_{j\sqrt{n}} \le \alpha$

The Basic Idea:
A golden ratio approx in $O(n^{3/2})$ time

We compute a table of Approximate Subsums $\hat{s}_\ell$ (ℓ=1,..., n) such that

$$1 \le \hat{s}_\ell / s_\ell \le \alpha = (1+\sqrt{5})/2$$



For filling the remaining entries:

1. If $s_{(j+1)\sqrt{n}} / s_{j\sqrt{n}} \le \alpha$

   set $\hat{s}_\ell = s_{(j+1)\sqrt{n}}$ for each ℓ in the interval

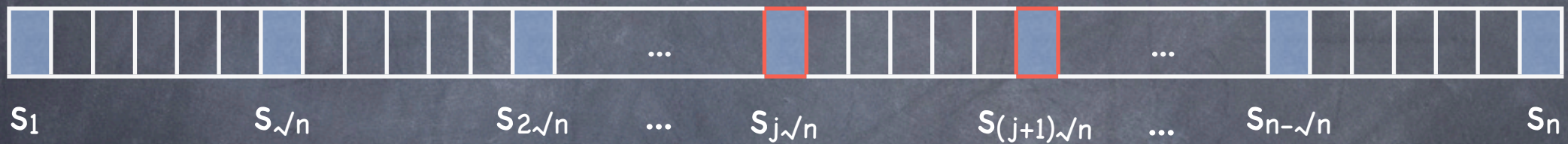
   therefore $\hat{s}_\ell / s_\ell = s_{(j+1)\sqrt{n}} / s_\ell \le s_{(j+1)\sqrt{n}} / s_{j\sqrt{n}} \le \alpha$

2. If $s_{(j+1)\sqrt{n}} / s_{j\sqrt{n}} > \alpha$ we compute exactly s_ℓ for each ℓ in the interval

The Basic Idea:
A golden ratio approx in $O(n^{3/2})$ time

We compute a table of Approximate Subsums $\hat{s}_\ell$ (ℓ=1,..., n) such that

$$1 \le \hat{s}_\ell / s_\ell \le \alpha = (1+\sqrt{5})/2$$



For filling the remaining entries:

1. If $s_{(j+1)\sqrt{n}} / s_{j\sqrt{n}} \le \alpha$

   set $\hat{s}_\ell = s_{(j+1)\sqrt{n}}$ for each ℓ in the interval

   therefore $\hat{s}_\ell / s_\ell = s_{(j+1)\sqrt{n}} / s_\ell \le s_{(j+1)\sqrt{n}} / s_{j\sqrt{n}} \le \alpha$

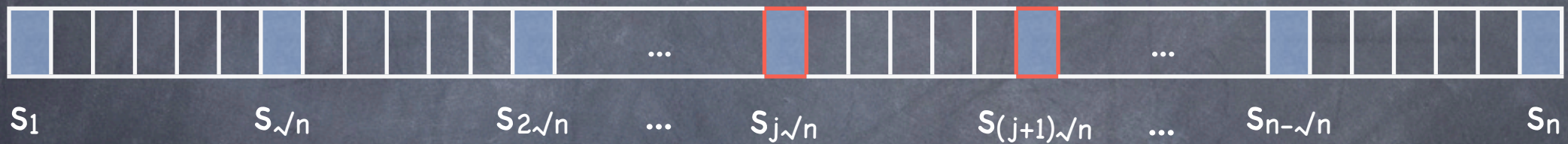
2. If $s_{(j+1)\sqrt{n}} / s_{j\sqrt{n}} > \alpha$ we compute exactly s_ℓ for each ℓ in the interval

Each big gaps (case 2.) requires $O(n\sqrt{n})$

The Basic Idea:
A golden ratio approx in $O(n^{3/2})$ time

We compute a table of Approximate Subsums $\hat{s}_\ell$ (ℓ=1,..., n) such that

$$1 \le \hat{s}_\ell / s_\ell \le \alpha = (1+\sqrt{5})/2$$



For filling the remaining entries:

1. If $s_{(j+1)\sqrt{n}} / s_{j\sqrt{n}} \le \alpha$

   set $\hat{s}_\ell = s_{(j+1)\sqrt{n}}$ for each ℓ in the interval

   therefore $\hat{s}_\ell / s_\ell = s_{(j+1)\sqrt{n}} / s_\ell \le s_{(j+1)\sqrt{n}} / s_{j\sqrt{n}} \le \alpha$

2. If $s_{(j+1)\sqrt{n}} / s_{j\sqrt{n}} > \alpha$ we compute exactly s_ℓ for each ℓ in the interval

Big gaps (case 2.) happen at most once!

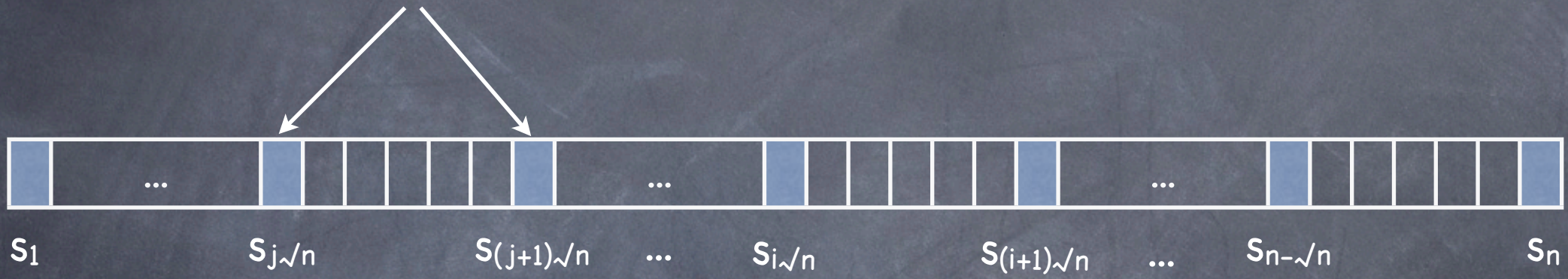
The Basic Idea:
A golden ratio approx in $O(n^{3/2})$ time

Big gap: $s_{j\sqrt{n}} / s_{(j+1)\sqrt{n}} < 1/\alpha$

There is at most one Big Gap

The Basic Idea:
A golden ratio approx in $O(n^{3/2})$ time

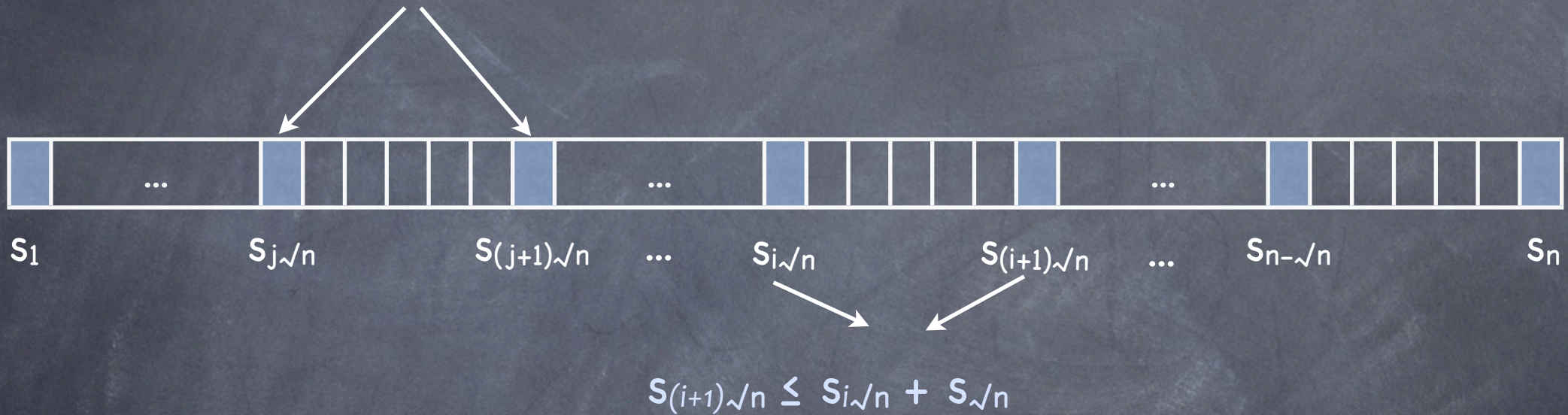
Big gap: $s_{j\sqrt{n}} / s_{(j+1)\sqrt{n}} < 1/\alpha$ --> $s_{\sqrt{n}} / s_{i\sqrt{n}} < 1/\alpha$



There is at most one Big Gap

The Basic Idea:
A golden ratio approx in $O(n^{3/2})$ time

Big gap: $s_{j\sqrt{n}} / s_{(j+1)\sqrt{n}} < 1/\alpha$ --> $s_{\sqrt{n}} / s_{i\sqrt{n}} < 1/\alpha$

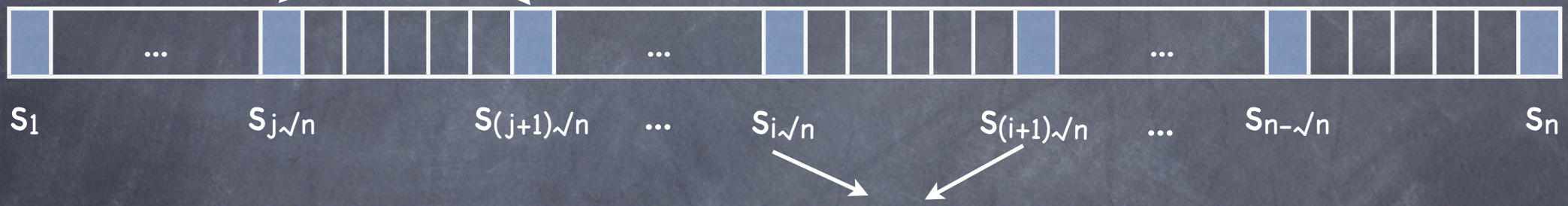


$s_{(i+1)\sqrt{n}} \leq s_{i\sqrt{n}} + s_{\sqrt{n}}$

There is at most one Big Gap

The Basic Idea:
A golden ratio approx in $O(n^{3/2})$ time

Big gap: $s_{j\sqrt{n}} / s_{(j+1)\sqrt{n}} < 1/\alpha$ --> $s_{\sqrt{n}} / s_{i\sqrt{n}} < 1/\alpha$



$s_{(i+1)\sqrt{n}} \le s_{i\sqrt{n}} + s_{\sqrt{n}}$

-->

$s_{(i+1)\sqrt{n}} / s_{i\sqrt{n}} \le 1 + s_{\sqrt{n}} / s_{i\sqrt{n}} \le 1 + 1/\alpha = \alpha$

There is at most one Big Gap

The better approximation and time bounds

The better approximation and time bounds

- We recursively apply the above ideas

The better approximation and time bounds

- We recursively apply the above ideas
  - let k s.t. the solution to $\alpha = 1+1/\alpha^k$ is $\leq 1+\varepsilon$

The better approximation and time bounds

- We recursively apply the above ideas
  - let k s.t. the solution to $\alpha = 1+1/\alpha^k$ is $\leq 1+\varepsilon$
    - the above argument gives now $(1+\varepsilon)$-approximation but we need to process up to k big gaps

The better approximation and time bounds

- We recursively apply the above ideas
  - let k s.t. the solution to $\alpha = 1+1/\alpha^k$ is $\leq 1+\varepsilon$
    - the above argument gives now $(1+\varepsilon)$-approximation but we need to process up to k big gaps
  - We partition into $n^{1/a}$ intervals of size $n^{1-1/a}$

The better approximation and time bounds

- We recursively apply the above ideas
  - let k s.t. the solution to $\alpha = 1+1/\alpha^k$ is $\leq 1+\varepsilon$
    - the above argument gives now $(1+\varepsilon)$-approximation but we need to process up to k big gaps
  - We partition into $n^{1/a}$ intervals of size $n^{1-1/a}$
    - partition big gaps into $n^{1/a}$ subintervals and recurse (up to a-1 times, a>1) ---> $O(k^a n^{1+1/a})$

Related Work and Applications

- Approximate Pattern Matching
  Constant time for Parikh vector membership queries in binary strings
  [Burci et. al., FUN 2010, Moosa-Rahman, JDA 2012]

- Finding large empty regions in data sets
  [Berkvist-Damaschke, Pattern Recognition 2006]

Indexes for Parikh vector membership queries

- Given a sequence **X** of n bits,
  - let $s_1, \ldots, s_n$ (resp. $r_1, \ldots, r_n$) be the maximum (minimum) subsums
- then **X** has a substring of length m with exactly k ones if and only if

$$r_m \leq k \leq s_m$$

Indexes for Parikh vector membership queries

- Given a sequence **X** of n bits,
  - let $s_1, \ldots, s_n$ (resp. $r_1, \ldots, r_n$) be the maximum (minimum) subsums
- then **X** has a substring of length m with exactly k ones if and only if

$$r_m \le k \le s_m$$

Let $\hat{s}_1, \ldots, \hat{s}_n$ (resp. $\hat{r}_1, \ldots, \hat{r}_n$) be the ε-approximate max (min) subsums.

- Then, **X** has a substring of length m with exactly k ones if

$$\hat{r}_m/(1-\varepsilon) \le k \le \hat{s}_m/(1+\varepsilon)$$

and only if

$$\hat{r}_m \le k \le \hat{s}_m$$

Conclusions and Open Problems

- The existence of exact linear algorithms remains open even in the binary case
- Lower bounds
- Parikh vector membership queries for binary strings
  - can we turn our approach into a Las Vegas algorithm?
  - extension to the case of general alphabets

Thank You