Dynamic Elias-Fano Representation

Giulio Ermanno Pibiri
University of Pisa and ISTI-CNR
Pisa, Italy
giulio.pibiri@di.unipi.it

Rossano Venturini
University of Pisa and ISTI-CNR
Pisa, Italy
rossano.venturini@unipi.it

The 28-th Annual Symposium on
Combinatorial Pattern Matching (CPM 2017)

Warsaw, Poland

06/07/2017

Introduction

A **dynamic ordered set** S is a data structure representing n objects and supporting the following operations:

- Insert(x) inserts x in S
- Delete(x) deletes x from S
- Search(x) checks whether x belongs to S
- Minimum() returns the minimum element of S
- Maximum() returns the maximum element of S
- Predecessor(x) returns $\max\{y \in S : y < x\}$
- Successor(x) returns $\min\{y \in S : y \geq x\}$

Introduction

A **dynamic ordered set** S is a data structure representing n objects and supporting the following operations:

- Insert(x) inserts x in S
- Delete(x) deletes x from S
- Search(x) checks whether x belongs to S
- Minimum() returns the minimum element of S
- Maximum() returns the maximum element of S
- Predecessor(x) returns $\max\{y \in S : y < x\}$
- Successor(x) returns $\min\{y \in S : y \geq x\}$

In the **comparison model** this is solved optimally by any self-balancing tree data structure in $O(\log n)$ time and $O(n)$ space.

Introduction

A **dynamic ordered set** S is a data structure representing n objects and supporting the following operations:

- Insert(x) inserts x in S
- Delete(x) deletes x from S
- Search(x) checks whether x belongs to S
- Minimum() returns the minimum element of S
- Maximum() returns the maximum element of S
- Predecessor(x) returns $\max\{y \in S : y < x\}$
- Successor(x) returns $\min\{y \in S : y \geq x\}$

In the **comparison model** this is solved optimally by any self-balancing tree data structure in $O(\log n)$ time and $O(n)$ space.

More efficient solutions there exist if the considered objects are **integers** drawn from a bounded universe of size u.

Introduction

A **dynamic ordered set** S is a data structure representing n objects and supporting the following operations:

- Insert(x) inserts x in S
- Delete(x) deletes x from S
- Search(x) checks whether x belongs to S
- Minimum() returns the minimum element of S
- Maximum() returns the maximum element of S
- Predecessor(x) returns $\max\{y \in S : y < x\}$
- Successor(x) returns $\min\{y \in S : y \geq x\}$

In the **comparison model** this is solved optimally by any self-balancing tree data structure in $O(\log n)$ time and $O(n)$ space.

More efficient solutions there exist if the considered objects are **integers** drawn from a bounded universe of size u.

Challenge

How to **optimally** solve the **integer** dynamic ordered set problem in **compressed space**?

Motivation

Integer Data Structures

- van Emde Boas Trees
- X/Y-Fast Tries
- Fusion Trees
- Exponential Search Trees
- …

Elias-Fano Encoding

- EF($S(n,u)$) = $n \log(u/n) + 2n$ bits to encode an ordered integer sequence S
- O(1) **Access**
- O($1 + \log(u/n)$) **Predecessor**

Motivation

Integer Data Structures

- van Emde Boas Trees
- X/Y-Fast Tries
- Fusion Trees
- Exponential Search Trees
- …

Elias-Fano Encoding

- EF($S(n,u)$) = $n \log(u/n) + 2n$ bits to encode an ordered integer sequence S
- O(1) **Access**
- $O(1 + \log(u/n))$ **Predecessor**

+ time
- space
+ dynamic

Motivation

Integer Data Structures

- van Emde Boas Trees
- X/Y-Fast Tries
- Fusion Trees
- Exponential Search Trees
- …

+ time
- space
+ dynamic

Elias-Fano Encoding

- $\text{EF}(S(n,u)) = n \log(u/n) + 2n$ bits to encode an ordered integer sequence S
- $O(1)$ **Access**
- $O(1 + \log(u/n))$ **Predecessor**

+ time
+ space
- static

Motivation

Integer Data Structures

- van Emde Boas Trees
- X/Y-Fast Tries
- Fusion Trees
- Exponential Search Trees
- …

\+ time
\- space
\+ dynamic

Elias-Fano Encoding

- $\text{EF}(S(n,u)) = n \log(u/n) + 2n$ bits to encode an ordered integer sequence S
- $O(1)$ **Access**
- $O(1 + \log(u/n))$ **Predecessor**

\+ time
\+ space
\- static

Motivation

Integer Data Structures

- van Emde Boas Trees
- X/Y-Fast Tries
- Fusion Trees
- Exponential Search Trees
- …

?

↔

Elias-Fano Encoding

- $\mathrm{EF}(S(n,u)) = n \log(u/n) + 2n$ bits to encode an ordered integer sequence S
- O(1) **Access**
- $O(1 + \log(u/n))$ **Predecessor**

\+ time
\- space
\+ dynamic

\+ time
\+ space
\- static

Motivation

Integer Data Structures

- van Emde Boas Trees
- X/Y-Fast Tries
- Fusion Trees
- Exponential Search Trees
- …

?

Elias-Fano Encoding

- $EF(S(n,u)) = n \log(u/n) + 2n$ bits to encode an ordered integer sequence S
- $O(1)$ **Access**
- $O(1 + \log(u/n))$ **Predecessor**

\+ time
\- space
\+ dynamic

\+ time
\+ space
\- static

Can we grab the best from both?

Objectives

Extend the *static* Elias-Fano representation of S as to support

1. **Predecessor**

2. **Access/Insert/Delete**

in **optimal time** and using $n \log(u/n) + 2n$ bits

Objectives

Extend the *static* Elias-Fano representation of S as to support

1. **Predecessor**
2. **Access/Insert/Delete**

sublinear redundancy

in **optimal time** and using $n \log(u/n) + 2n$ bits + $o(n)$ bits

Objectives

Extend the *static* Elias-Fano representation of S as to support

1. **Predecessor**

2. **Access/Insert/Delete**

sublinear redundancy

in **optimal time** and using $n \log(u/n) + 2n$ bits + $o(n)$ bits

What is optimal?

Objectives

Extend the *static* Elias-Fano representation of S as to support

1. **Predecessor**

2. **Access/Insert/Delete**

sublinear redundancy

in **optimal time** and using $n \log(u/n) + 2n$ bits + o(n) bits

What is optimal?

Lower bounds

Objectives

Extend the *static* Elias-Fano representation of S as to support

1. **Predecessor**
2. **Access/Insert/Delete**

sublinear redundancy

in **optimal time** and using $n \log(u/n) + 2n$ bits + $o(n)$ bits

What is optimal?

Lower bounds

1. [Patrascu and Thorup, STC 2007]

- Optimal space/time trade-off
- m bits, where $a = \log(m/n) - \log w$

$$\Theta\left(\min\left\{\log_w n, \log\frac{w - \log n}{a}, \frac{\log\frac{w}{a}}{\log(\frac{a}{\log n}\log\frac{w}{a})}, \frac{\log\frac{w}{a}}{\log(\log\frac{w}{a} / \log\frac{\log n}{a})}\right\}\right)$$

Objectives

Extend the *static* Elias-Fano representation of S as to support

→ 1. **Predecessor**

→ 2. **Access/Insert/Delete**

sublinear redundancy

in **optimal time** and using $n \log(u/n) + 2n$ bits + $o(n)$ bits

What is optimal?

Lower bounds

1. [Patrascu and Thorup, STC 2007]

- Optimal space/time trade-off
- m bits, where $a = \log(m/n) - \log w$

$$\Theta\left(\min\left\{\log_w n, \log\frac{w-\log n}{a}, \frac{\log\frac{w}{a}}{\log(\frac{a}{\log n}\log\frac{w}{a})}, \frac{\log\frac{w}{a}}{\log(\log\frac{w}{a} / \log\frac{\log n}{a})}\right\}\right)$$

2. [Fredman and Saks, STC 1989]

Dynamic List Representation Problem:

- Access/Insert/Delete in $\Omega(\log n / \log\log n)$ amortized time
- $w \leq \log^{\gamma} n$ for some γ

Objectives

Extend the *static* Elias-Fano representation of S as to support

→ 1. **Predecessor**

→ 2. **Access/Insert/Delete**

sublinear redundancy

in **optimal time** and using $n \log(u/n) + 2n$ bits + $o(n)$ bits

What is optimal?

Lower bounds - polynomial universes

1. [Patrascu and Thorup, STC 2007]

- Optimal space/time trade-off
- m bits, where $a = \log(m/n) - \log w$

$$\Theta\left(\min\left\{\log_w n, \log\frac{w-\log n}{a}, \frac{\log\frac{w}{a}}{\log\left(\frac{a}{\log n}\log\frac{w}{a}\right)}, \frac{\log\frac{w}{a}}{\log\left(\log\frac{w}{a} / \log\frac{\log n}{a}\right)}\right\}\right)$$

2. [Fredman and Saks, STC 1989]

Dynamic List Representation Problem:

- Access/Insert/Delete in $\Omega(\log n / \log\log n)$ amortized time
- $w \leq \log^{\gamma} n$ for some γ

Objectives

Extend the *static* Elias-Fano representation of S as to support

→ 1. **Predecessor**

→ 2. **Access/Insert/Delete**

sublinear redundancy

in **optimal time** and using $n \log(u/n) + 2n$ bits + $o(n)$ bits

What is optimal?

Lower bounds - polynomial universes

1. [Patrascu and Thorup, STC 2007]

- Optimal space/time trade-off
- m bits, where $a = \log(m/n) - \log w$

$$\Theta\left(\min\left\{\log_w n, \log\frac{w - \log n}{a}, \frac{\log\frac{w}{a}}{\log(\frac{a}{\log n}\log\frac{w}{a})}, \frac{\log\frac{w}{a}}{\log(\log\frac{w}{a} / \log\frac{\log n}{a})}\right\}\right)$$

For Elias-Fano, $a = \log(\log(u/n) + 2)$ bits:

the second branch becomes $O(\log\log n)$

2. [Fredman and Saks, STC 1989]

Dynamic List Representation Problem:

- Access/Insert/Delete in $\Omega(\log n / \log\log n)$ amortized time
- $w \leq \log^{\gamma} n$ for some γ

Results - Static Elias-Fano Optimal Predecessor Queries

For $u = n^\gamma$, $\gamma = \Theta(1)$:

- EF($\mathcal{S}(n,u)$) + o(n) bits
- O(1) Access
- O(min{1+log(u/n), loglog n}) Predecessor

Results - Static Elias-Fano Optimal Predecessor Queries

For $u = n^{\gamma}$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + o(n) bits
- O(1) Access
- O(min{1+log(u/n), loglog n}) Predecessor

EF($S(n,u)$) = n log(u/n) + 2n bits

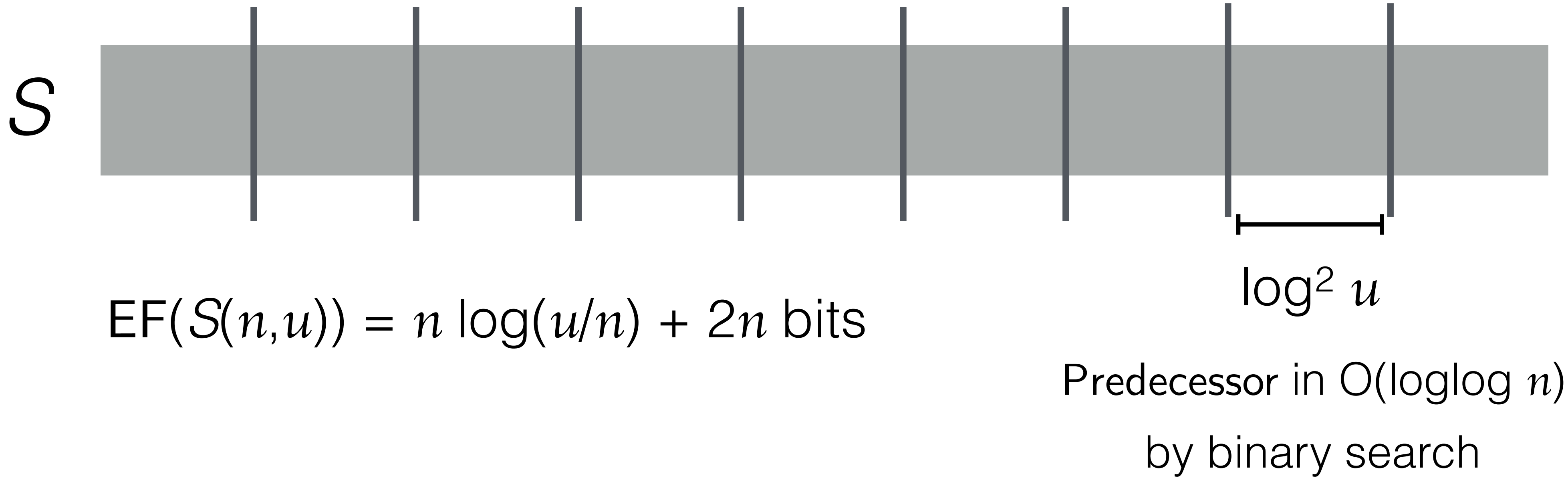
For $u = n^\gamma$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + o(n) bits
- O(1) Access
- O(min{1+log(u/n), loglog n}) Predecessor

EF($S(n,u)$) = n log(u/n) + 2n bits

For $u = n^{\gamma}$, $\gamma = \Theta(1)$:

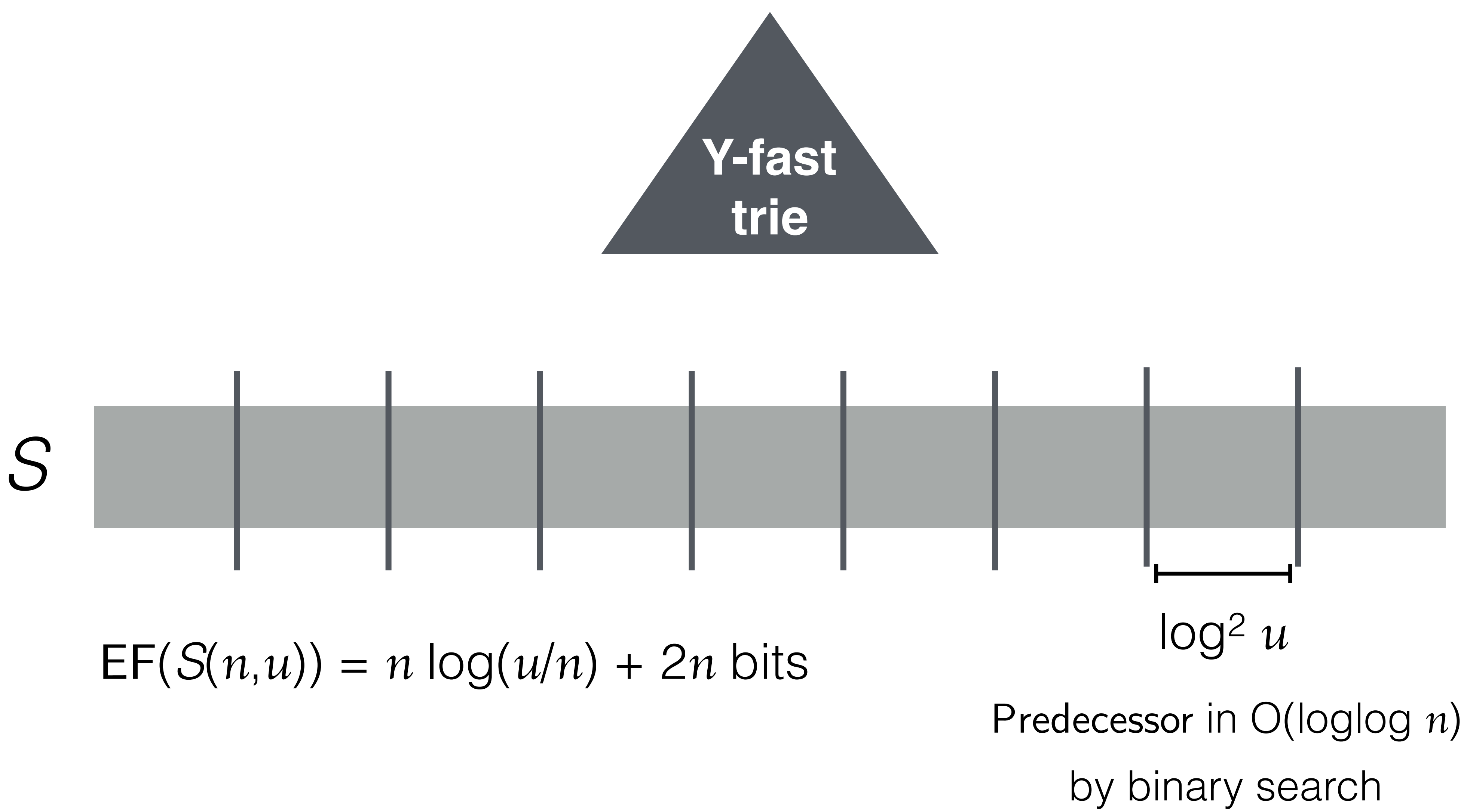
- EF($S(n,u)$) + o(n) bits
- O(1) Access
- O(min{1+log(u/n), loglog n}) Predecessor



Results - Static Elias-Fano Optimal Predecessor Queries 1

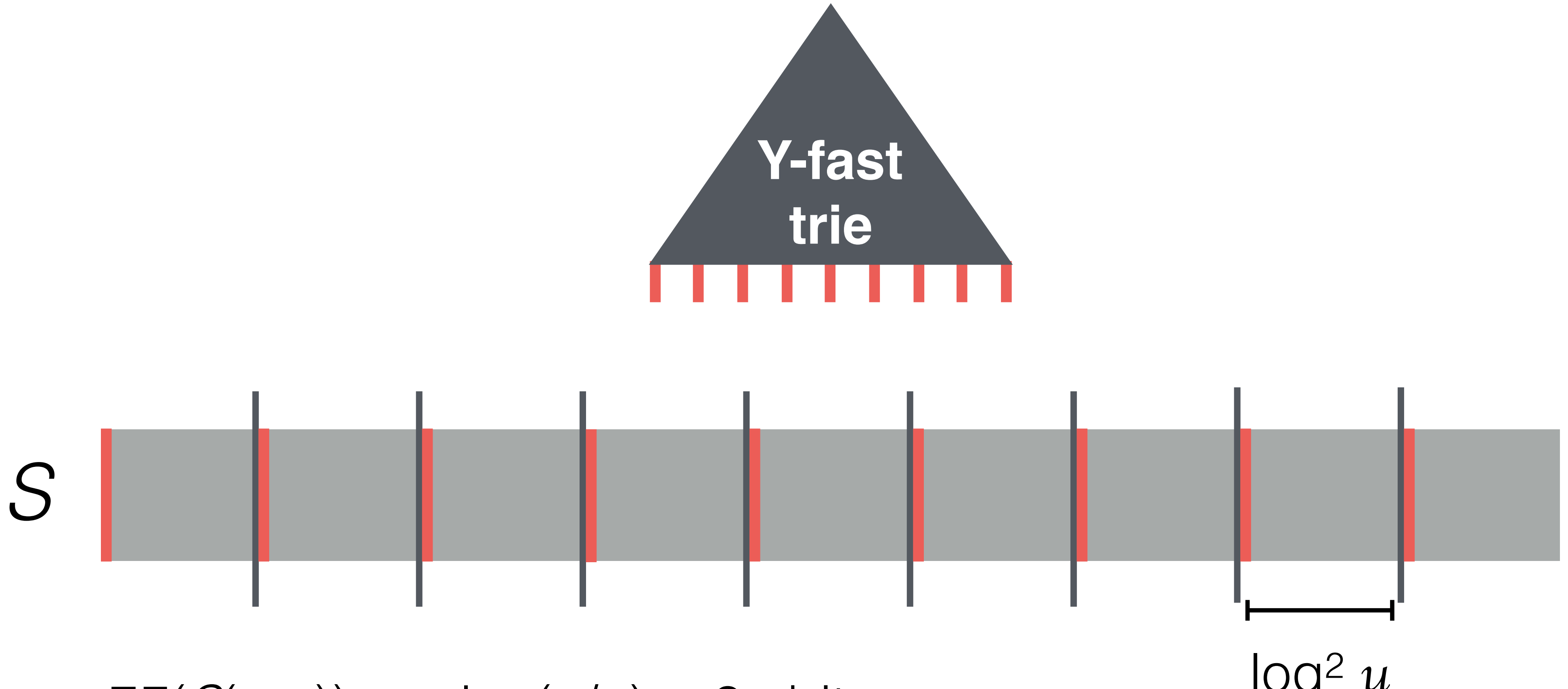
For $u = n^{\gamma}$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + o(n) bits
- O(1) Access
- O(min{1+log(u/n), loglog n}) Predecessor



For $u = n^\gamma$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + o(n) bits
- O(1) Access
- O(min{1+log(u/n), loglog n}) Predecessor



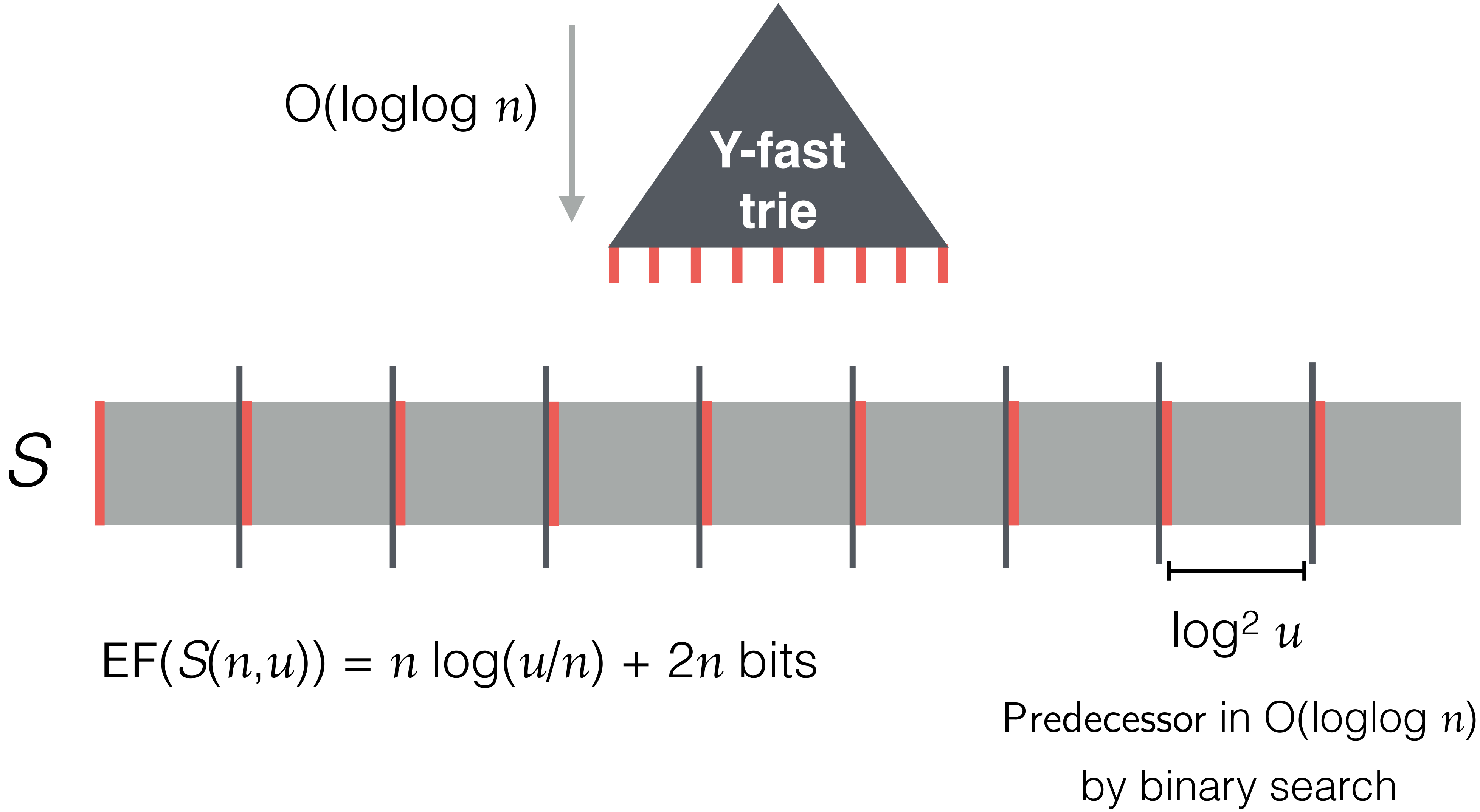
EF($S(n,u)$) = $n \log(u/n) + 2n$ bits

Predecessor in O(loglog n) by binary search

Results - Static Elias-Fano Optimal Predecessor Queries

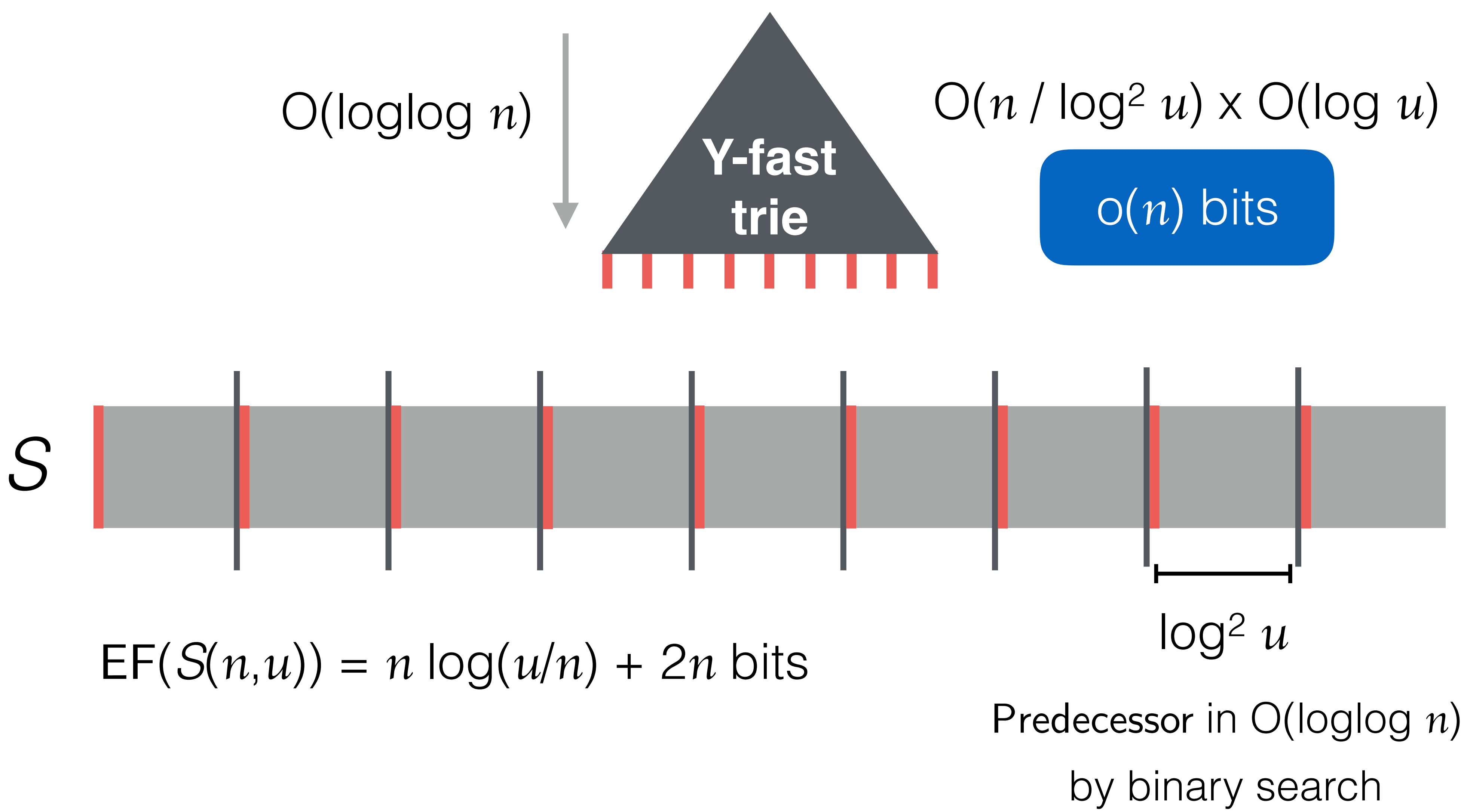
For $u = n^\gamma$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + o(n) bits
- O(1) Access
- O(min{1+log(u/n), loglog n}) Predecessor



For $u = n^\gamma$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + o(n) bits
- O(1) Access
- O(min{1+log(u/n), loglog n}) Predecessor

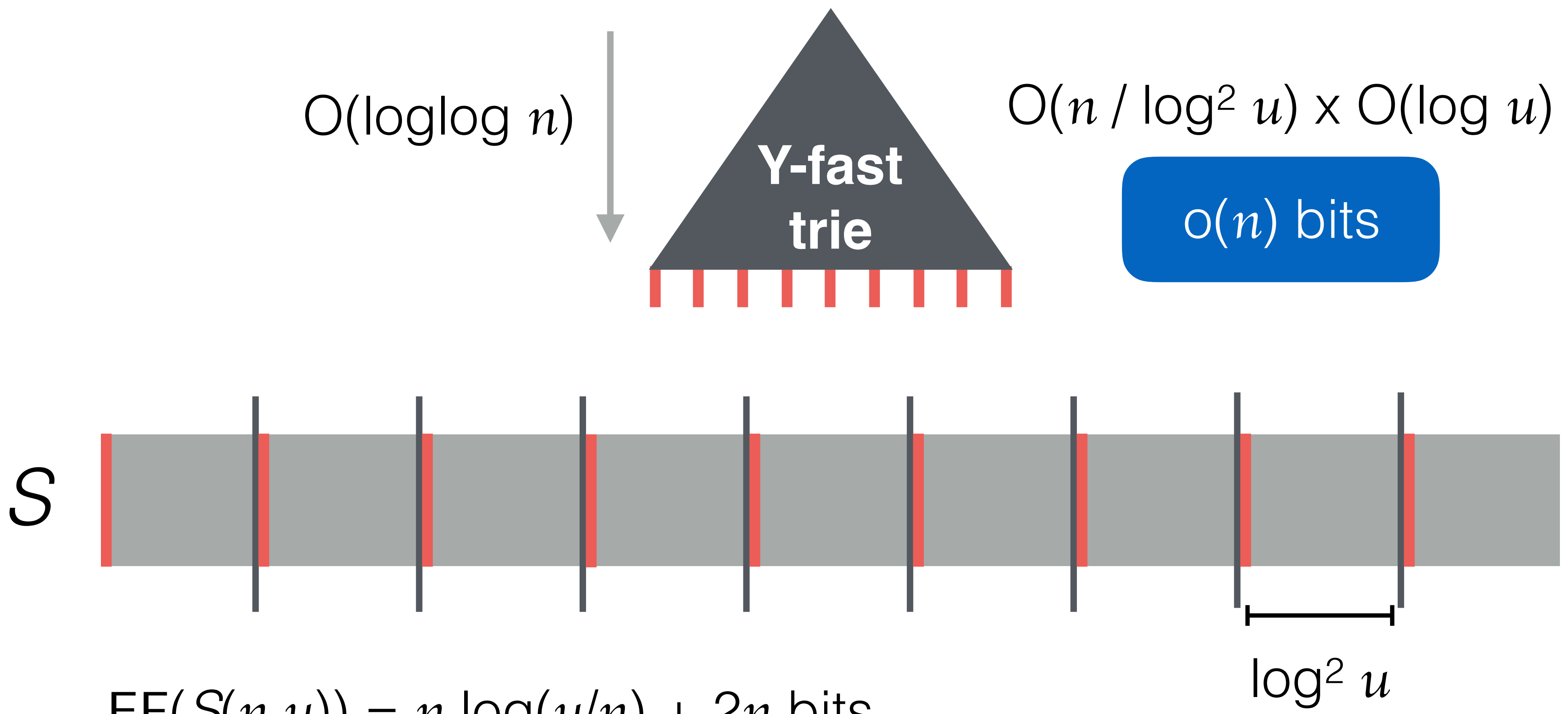


For $u = n^{\gamma}$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + o(n) bits
- O(1) Access
- O(min{1+log(u/n), loglog n}) Predecessor

  for the tiny range

  $1 \leq \gamma \leq 1 + \text{loglog } n / \log n$



EF($S(n,u)$) = n log(u/n) + 2n bits

Predecessor in O(loglog n)

by binary search

Results - Extensible Elias-Fano

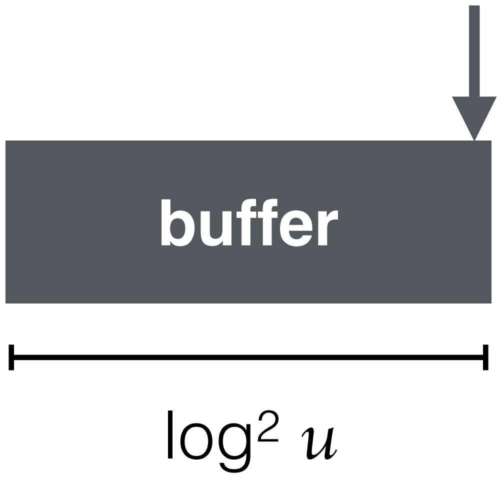
For $u = n^{\gamma}$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + o(n) bits
- O(1) Access
- O(1) Append (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor

Results - Extensible Elias-Fano

For $u = n^\gamma$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + o(n) bits
- O(1) Access
- O(1) Append (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor

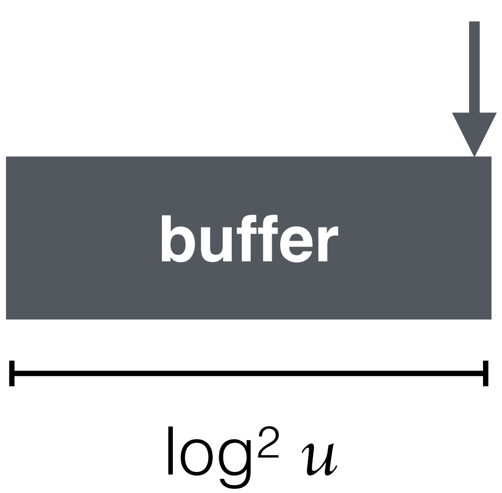


Results - Extensible Elias-Fano

For $u = n^{\gamma}$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + o(n) bits
- O(1) Access
- O(1) Append (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor

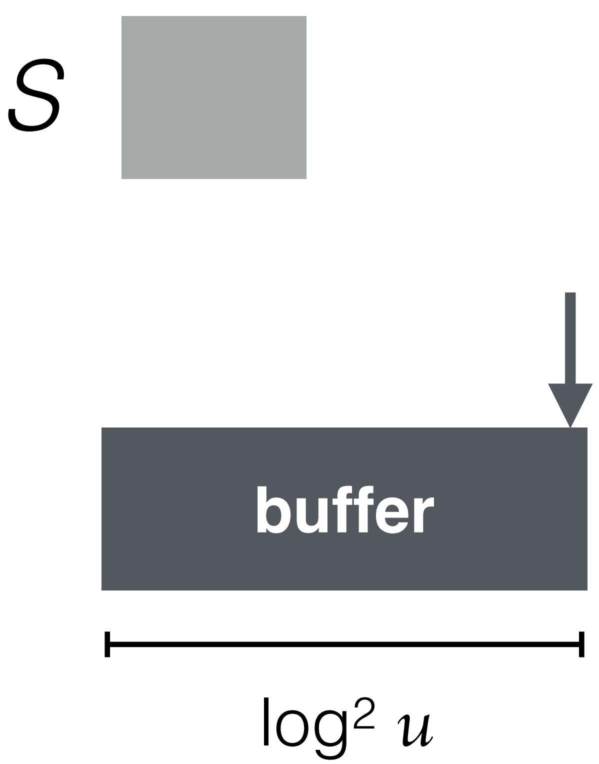
S



Results - Extensible Elias-Fano 2

For $u = n^{\gamma}$, $\gamma = \Theta(1)$:

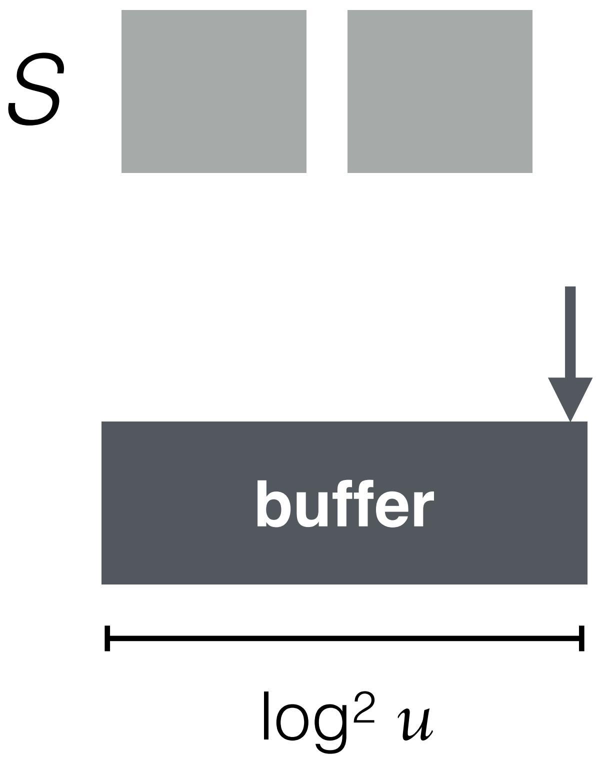
- EF($S(n,u)$) + o(n) bits
- O(1) Access
- O(1) Append (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor



Results - Extensible Elias-Fano

For $u = n^\gamma$, $\gamma = \Theta(1)$:

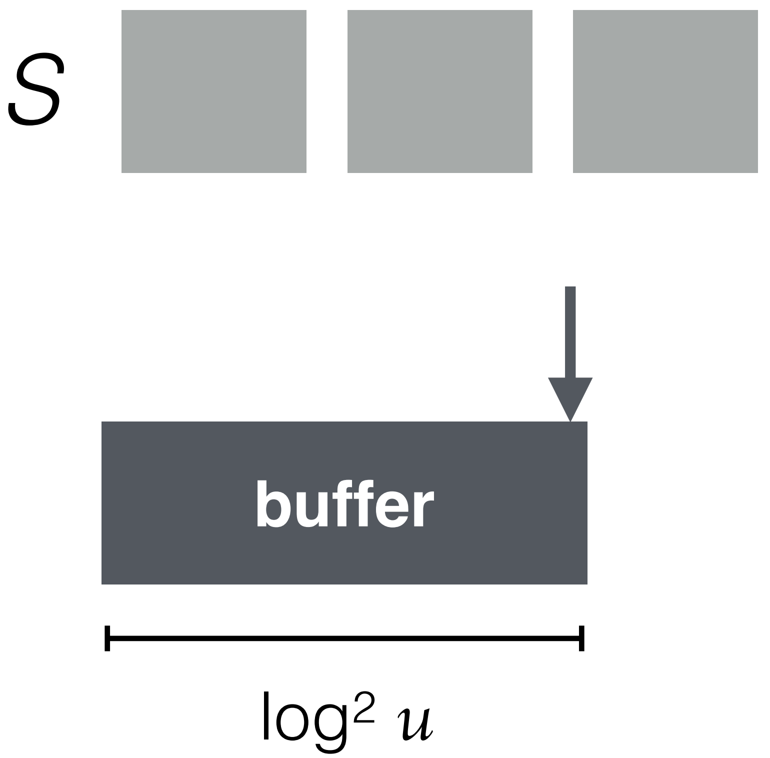
- EF($S(n,u)$) + o(n) bits
- O(1) Access
- O(1) Append (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor



Results - Extensible Elias-Fano

For $u = n^{\gamma}$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + o(n) bits
- O(1) Access
- O(1) Append (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor

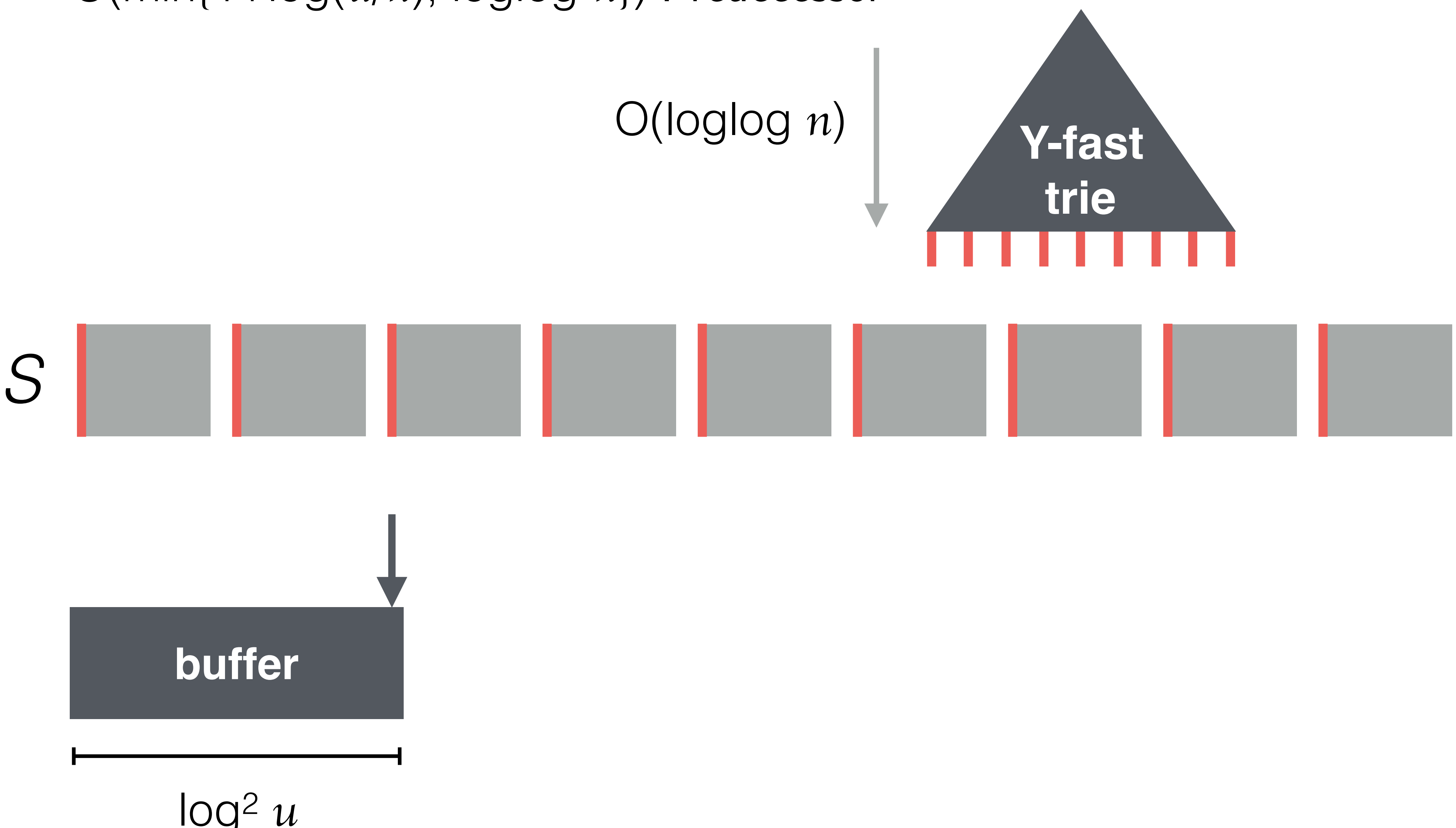


For $u = n^\gamma$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + o(n) bits
- O(1) Access
- O(1) Append (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor

For $u = n^\gamma$, $\gamma = \Theta(1)$:

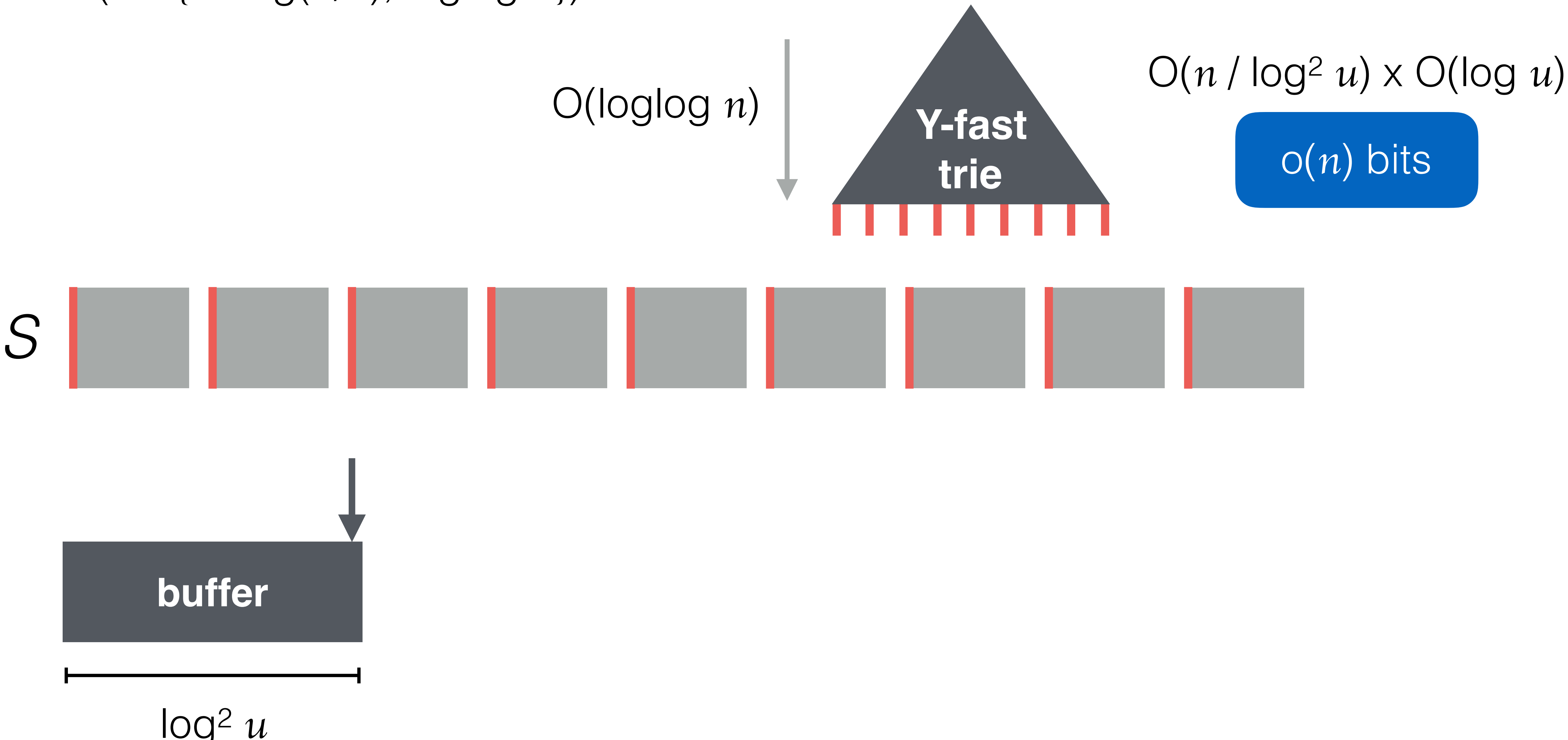
- EF($S(n,u)$) + o(n) bits
- O(1) Access
- O(1) Append (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor



Results - Extensible Elias-Fano

For $u = n^{\gamma}$, $\gamma = \Theta(1)$:

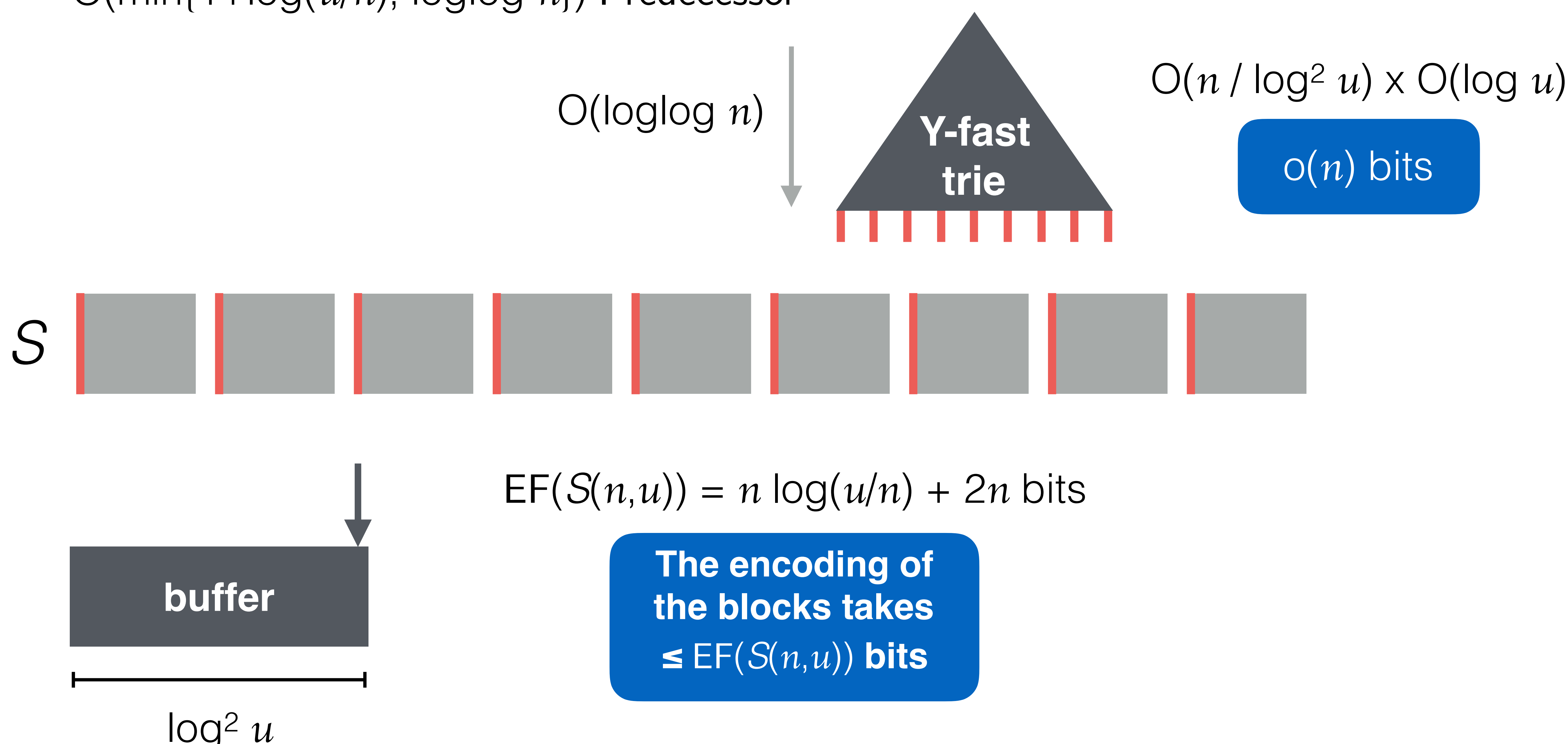
- EF($S(n,u)$) + o(n) bits
- O(1) Access
- O(1) Append (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor



Results - Extensible Elias-Fano

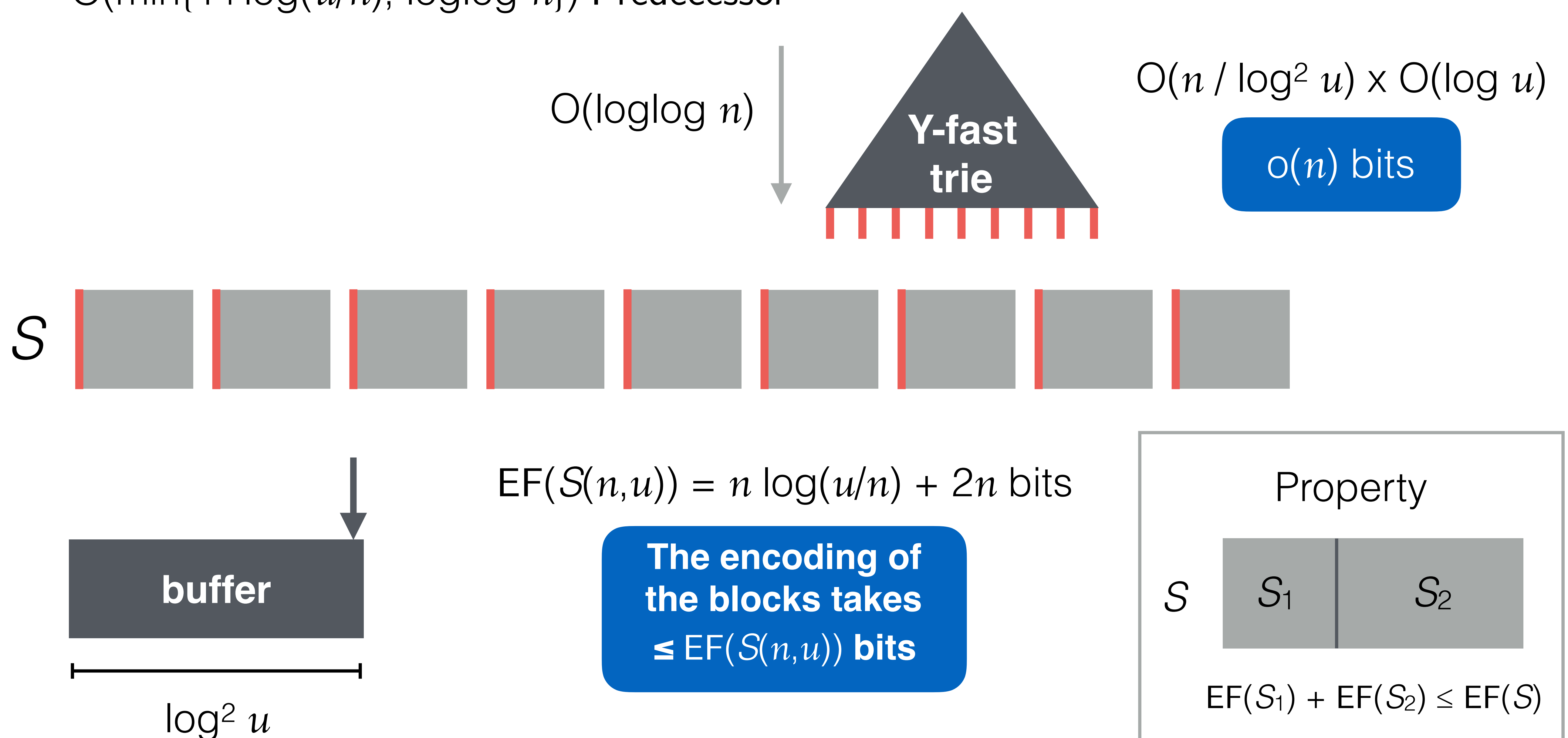
For $u = n^{\gamma}$, $\gamma = \Theta(1)$:

- $EF(S(n,u))$ + o(n) bits
- O(1) Access
- O(1) Append (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor



For $u = n^{\gamma}$, $\gamma = \Theta(1)$:

- $EF(S(n,u)) + o(n)$ bits
- $O(1)$ Access
- $O(1)$ Append (amortized)
- $O(\min\{1+\log(u/n), \log\log n\})$ Predecessor



Results - Dynamic Elias-Fano

For $u = n^\gamma$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + o(n) bits
- O(log n / loglog n) Access
- O(log n / loglog n) Insert/Delete (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor

Results - Dynamic Elias-Fano

For $u = n^{\gamma}$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + o(n) bits
- O(log n / loglog n) Access
- O(log n / loglog n) Insert/Delete (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor

S

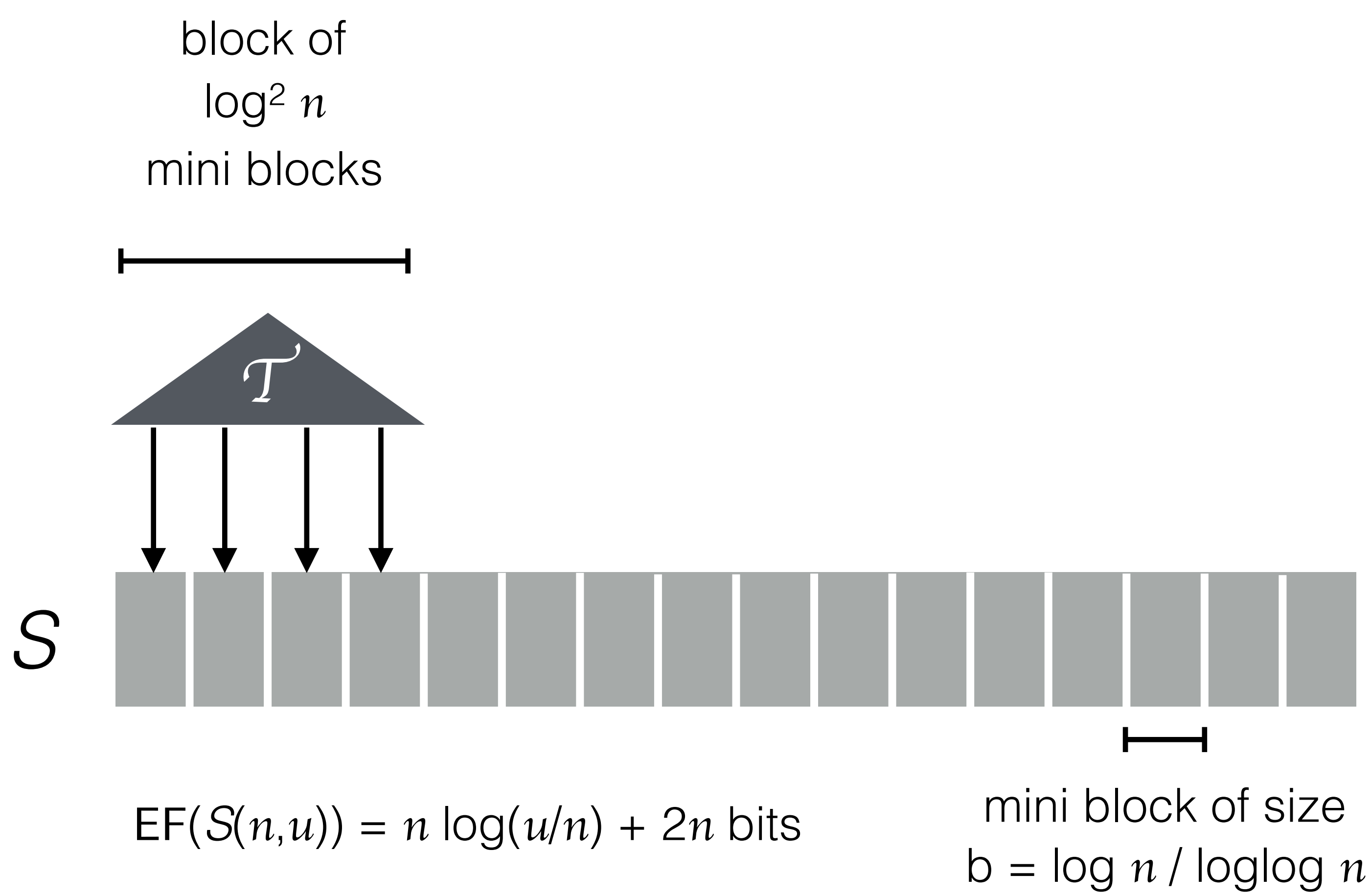
EF($S(n,u)$) = n log(u/n) + 2n bits

mini block of size
b = log n / loglog n

Results - Dynamic Elias-Fano

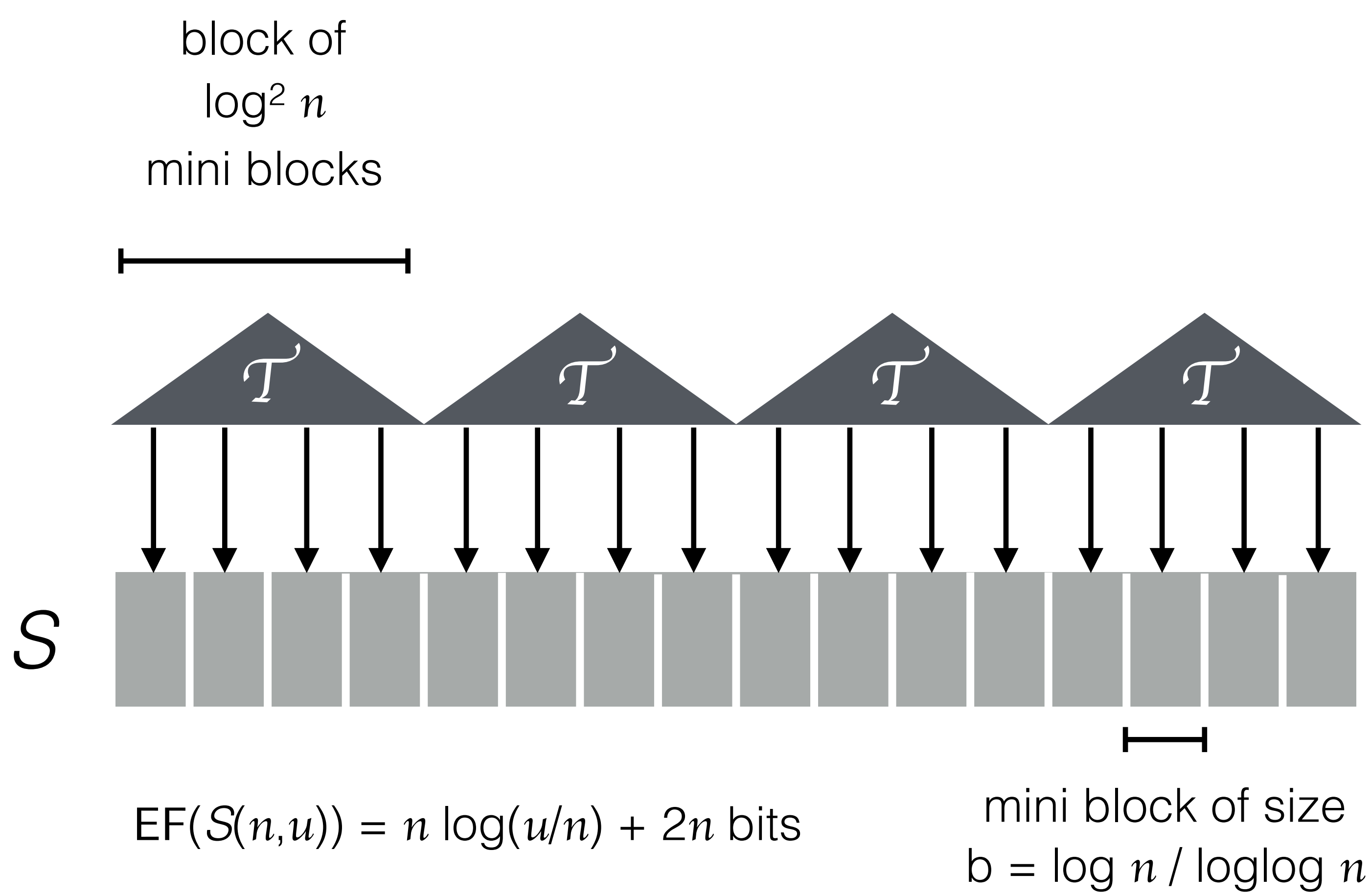
For $u = n^{\gamma}$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + o(n) bits
- O(log n / loglog n) Access
- O(log n / loglog n) Insert/Delete (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor



For $u = n^{\gamma}$, $\gamma = \Theta(1)$:

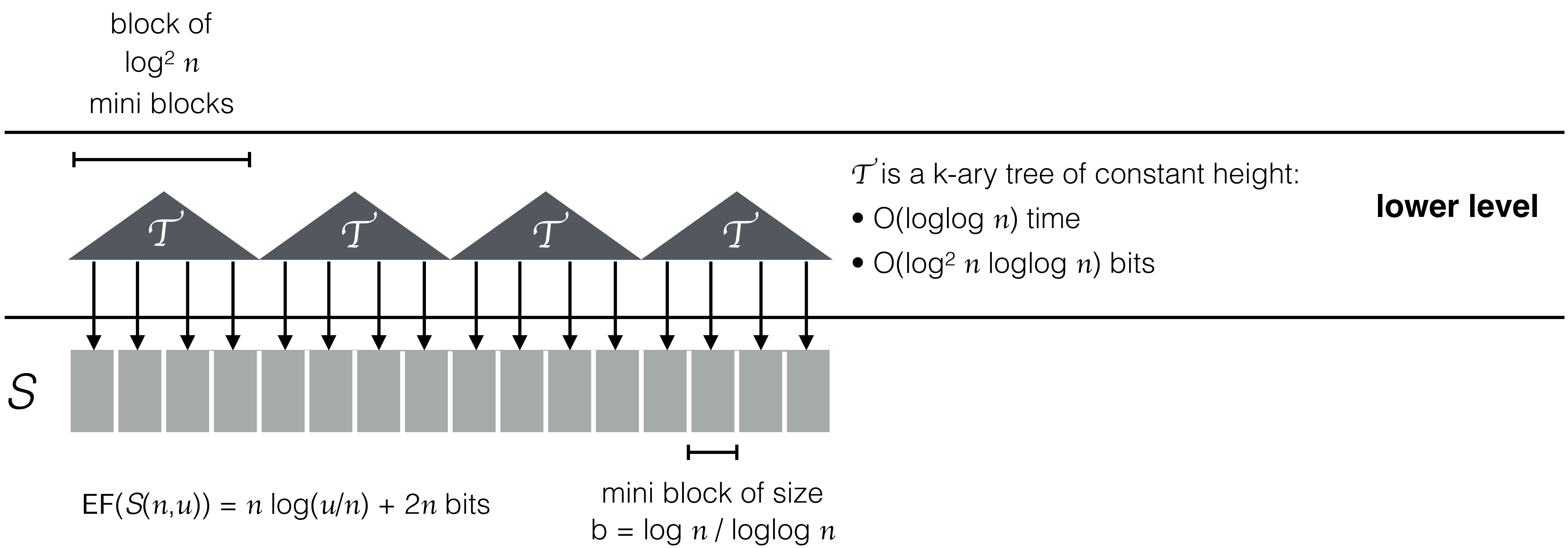
- EF($S(n,u)$) + o(n) bits
- O(log n / loglog n) Access
- O(log n / loglog n) Insert/Delete (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor



Results - Dynamic Elias-Fano

For $u = n^\gamma$, $\gamma = \Theta(1)$:

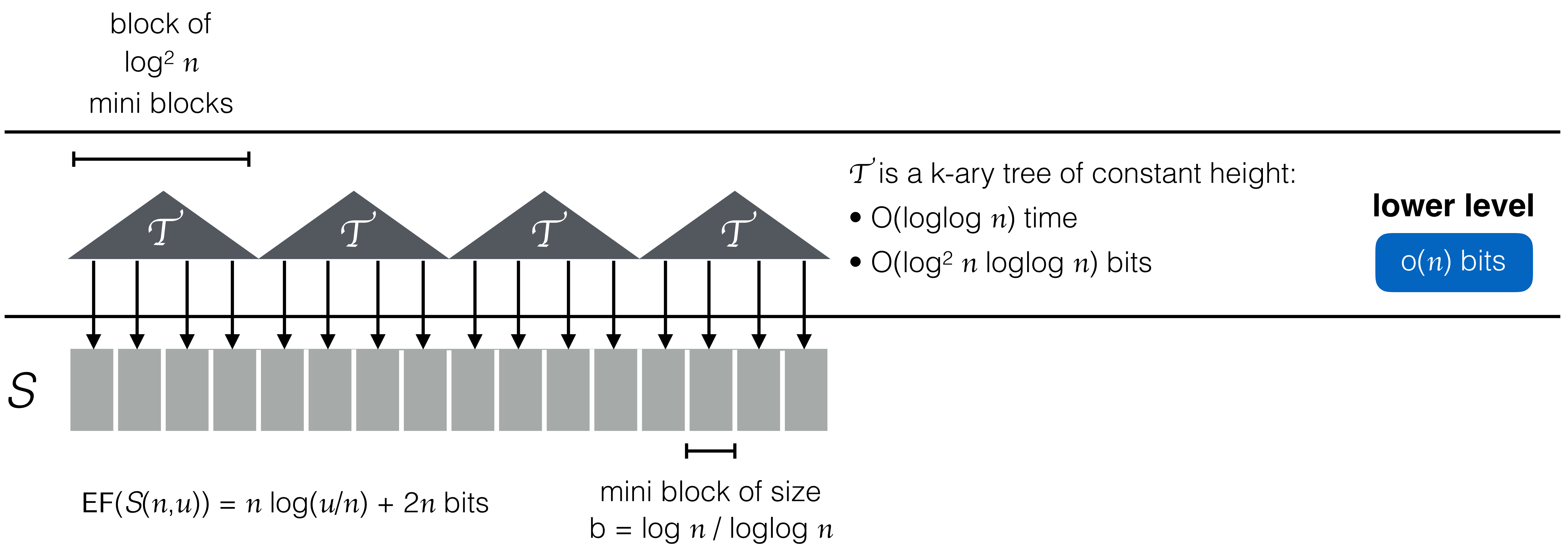
- $\text{EF}(S(n,u)) + o(n)$ bits
- $O(\log n / \log\log n)$ Access
- $O(\log n / \log\log n)$ Insert/Delete (amortized)
- $O(\min\{1+\log(u/n), \log\log n\})$ Predecessor



Results - Dynamic Elias-Fano

For $u = n^{\gamma}$, $\gamma = \Theta(1)$:

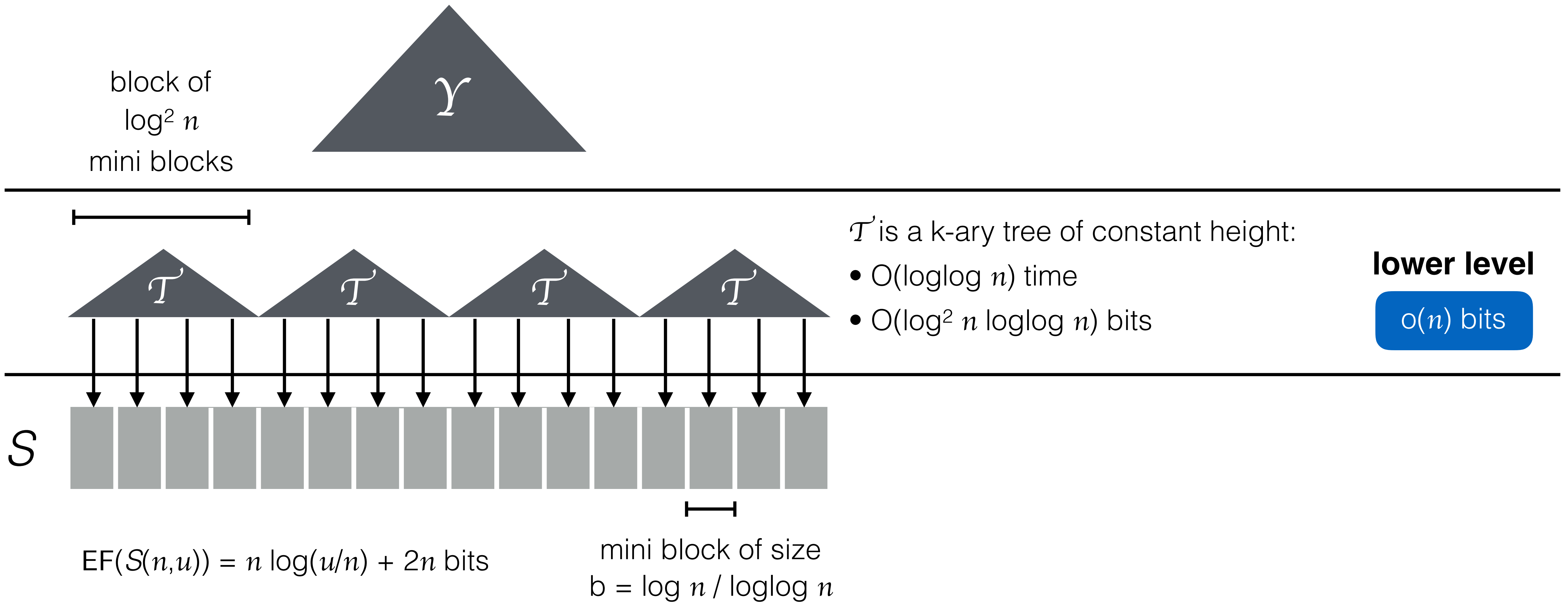
- EF($S(n,u)$) + o(n) bits
- O(log n / loglog n) Access
- O(log n / loglog n) Insert/Delete (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor



Results - Dynamic Elias-Fano

For $u = n^{\gamma}$, $\gamma = \Theta(1)$:

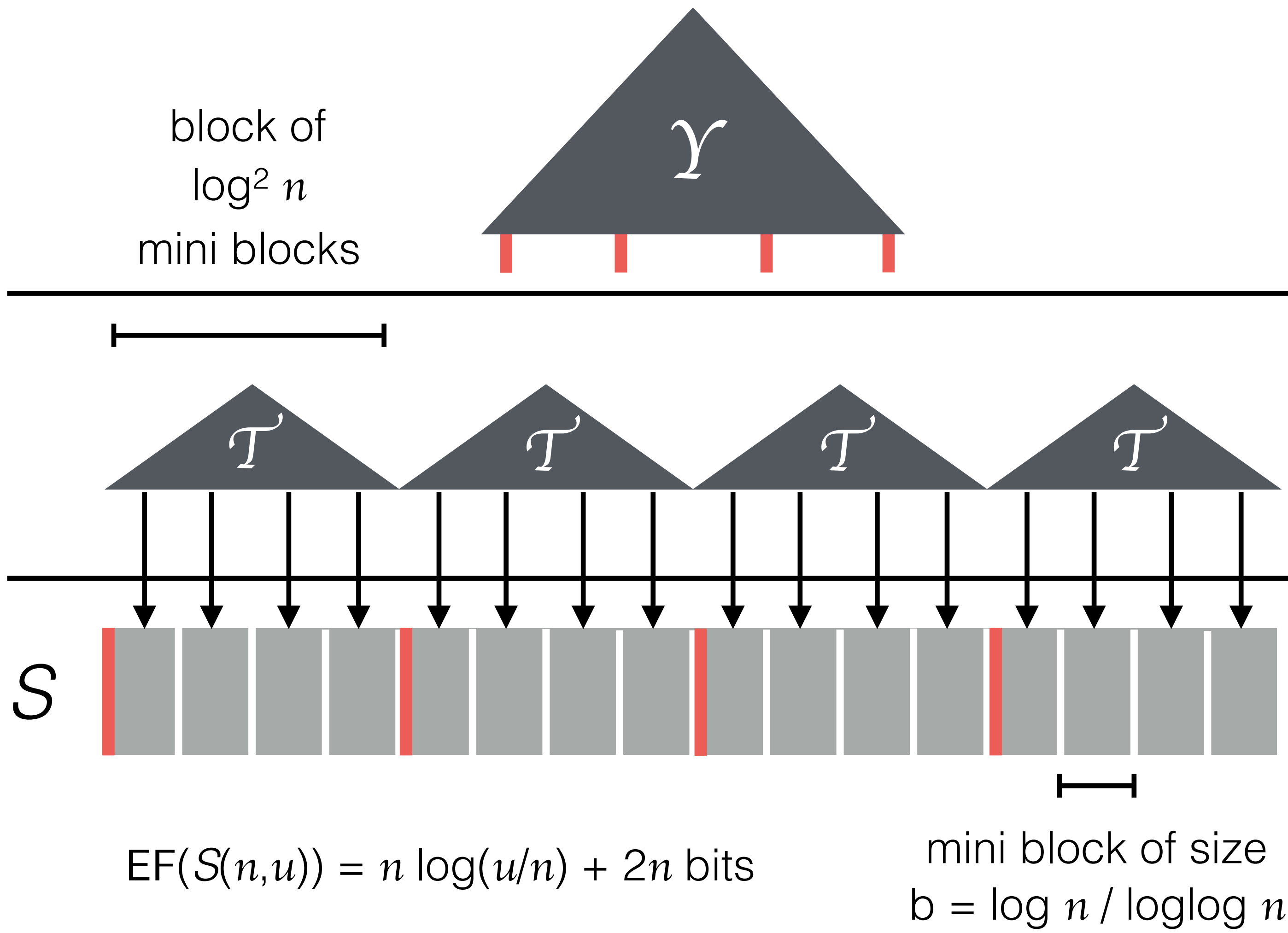
- EF($S(n,u)$) + o(n) bits
- O(log n / loglog n) Access
- O(log n / loglog n) Insert/Delete (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor



Results - Dynamic Elias-Fano

For $u = n^{\gamma}$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + o(n) bits
- O(log n / loglog n) Access
- O(log n / loglog n) Insert/Delete (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor



$\mathcal{T}$ is a k-ary tree of constant height:

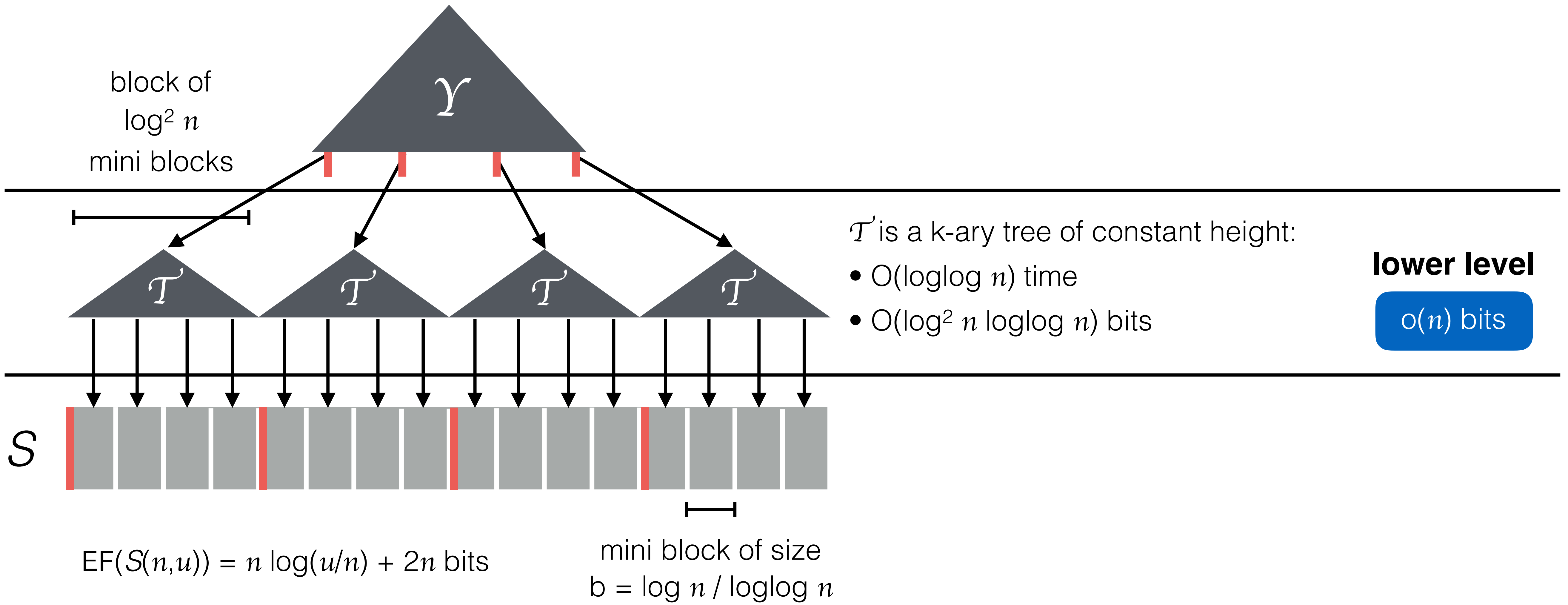
- O(loglog n) time
- O($\log^2 n$ loglog n) bits

lower level

o(n) bits

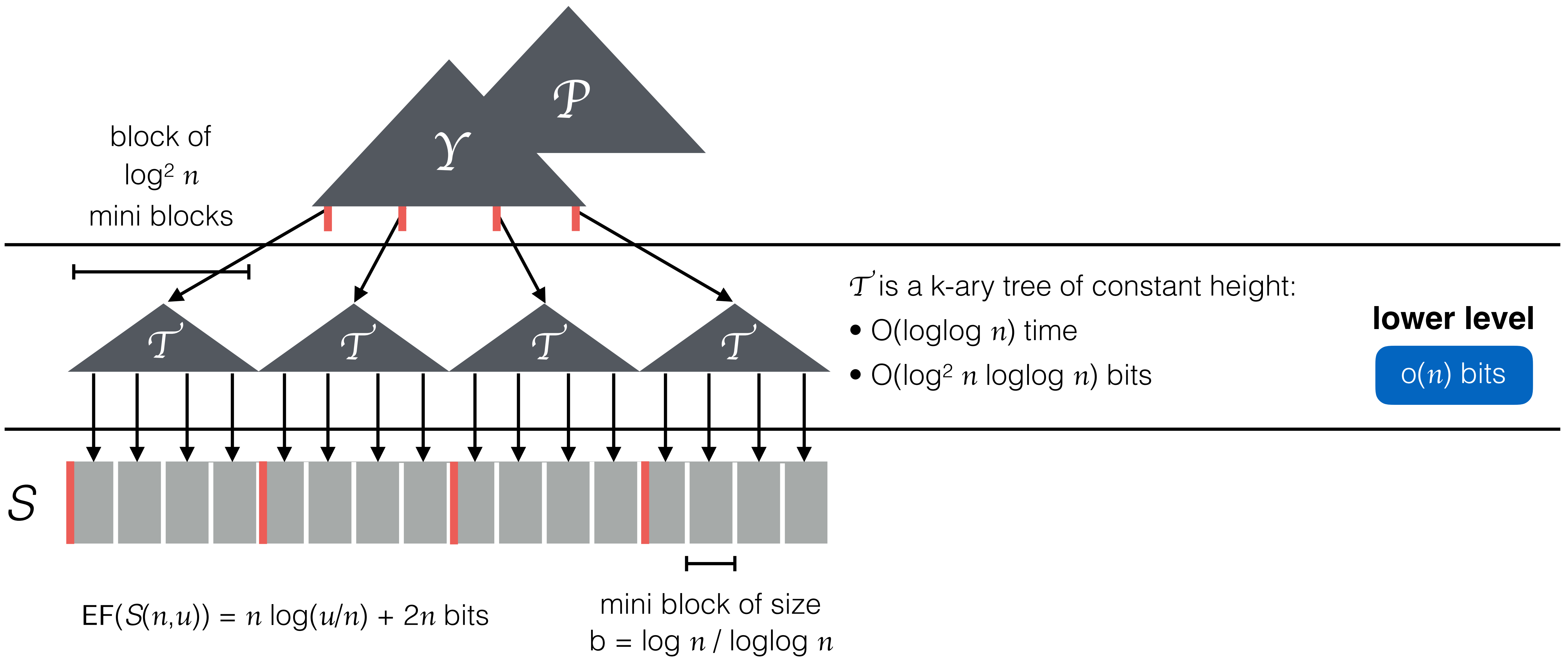
For $u = n^{\Upsilon}$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + o(n) bits
- O(log n / loglog n) Access
- O(log n / loglog n) Insert/Delete (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor



For $u = n^{\gamma}$, $\gamma = \Theta(1)$:

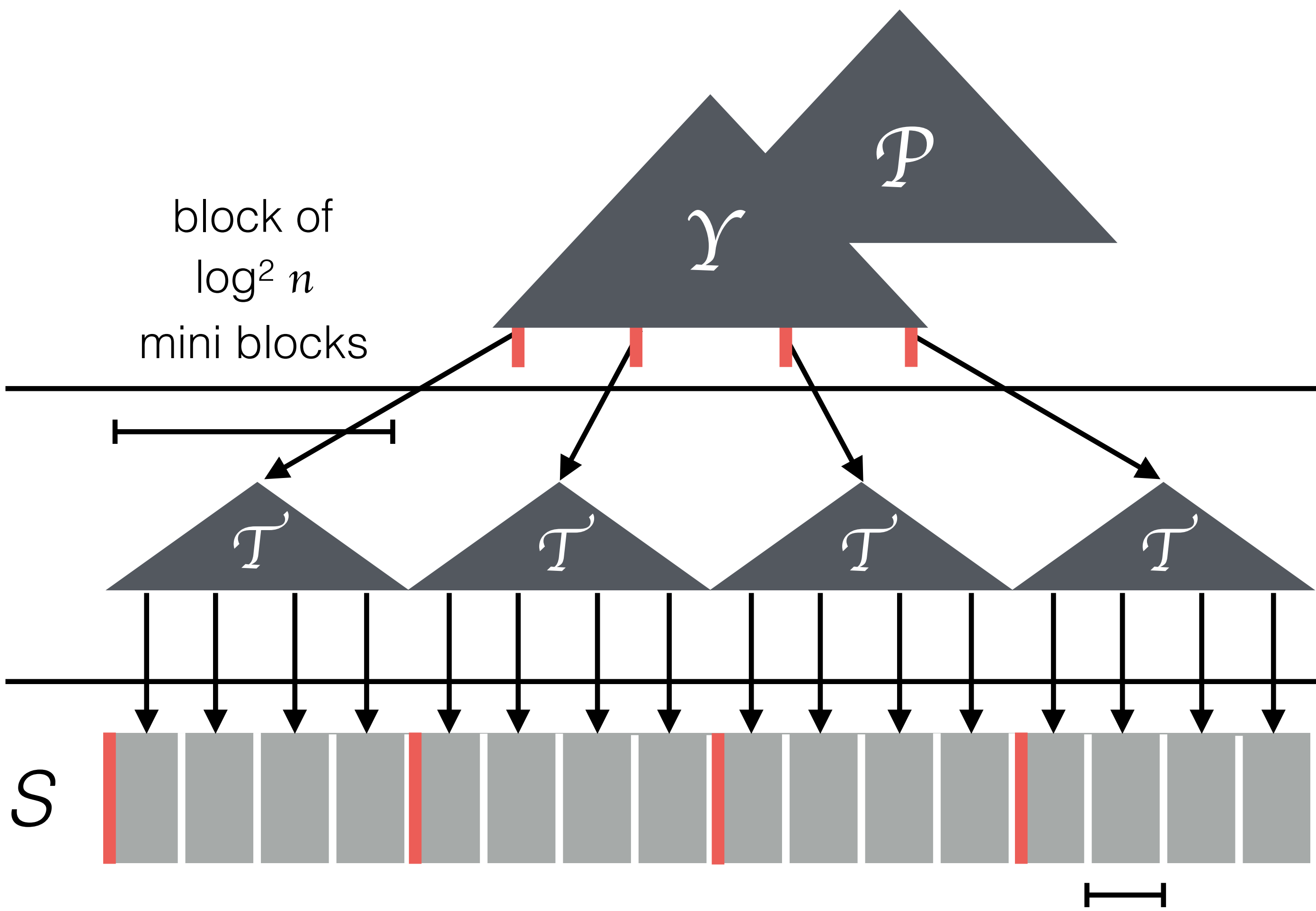
- EF($S(n,u)$) + $o(n)$ bits
- $O(\log n / \log\log n)$ Access
- $O(\log n / \log\log n)$ Insert/Delete (amortized)
- $O(\min\{1+\log(u/n), \log\log n\})$ Predecessor



Results - Dynamic Elias-Fano

For $u = n^{\gamma}$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + o(n) bits
- O(log n / loglog n) Access
- O(log n / loglog n) Insert/Delete (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor



Υ is an Y-fast trie

- O(loglog n) time

$\mathcal{P}$ is a dynamic prefix-sums DS [1]

- O(b) time

O(n / (b x log² n)) x O(log u) = o(n) bits each

upper level

$\mathcal{T}$ is a k-ary tree of constant height:

- O(loglog n) time
- O(log² n loglog n) bits

lower level

o(n) bits

EF($S(n,u)$) = n log(u/n) + 2n bits

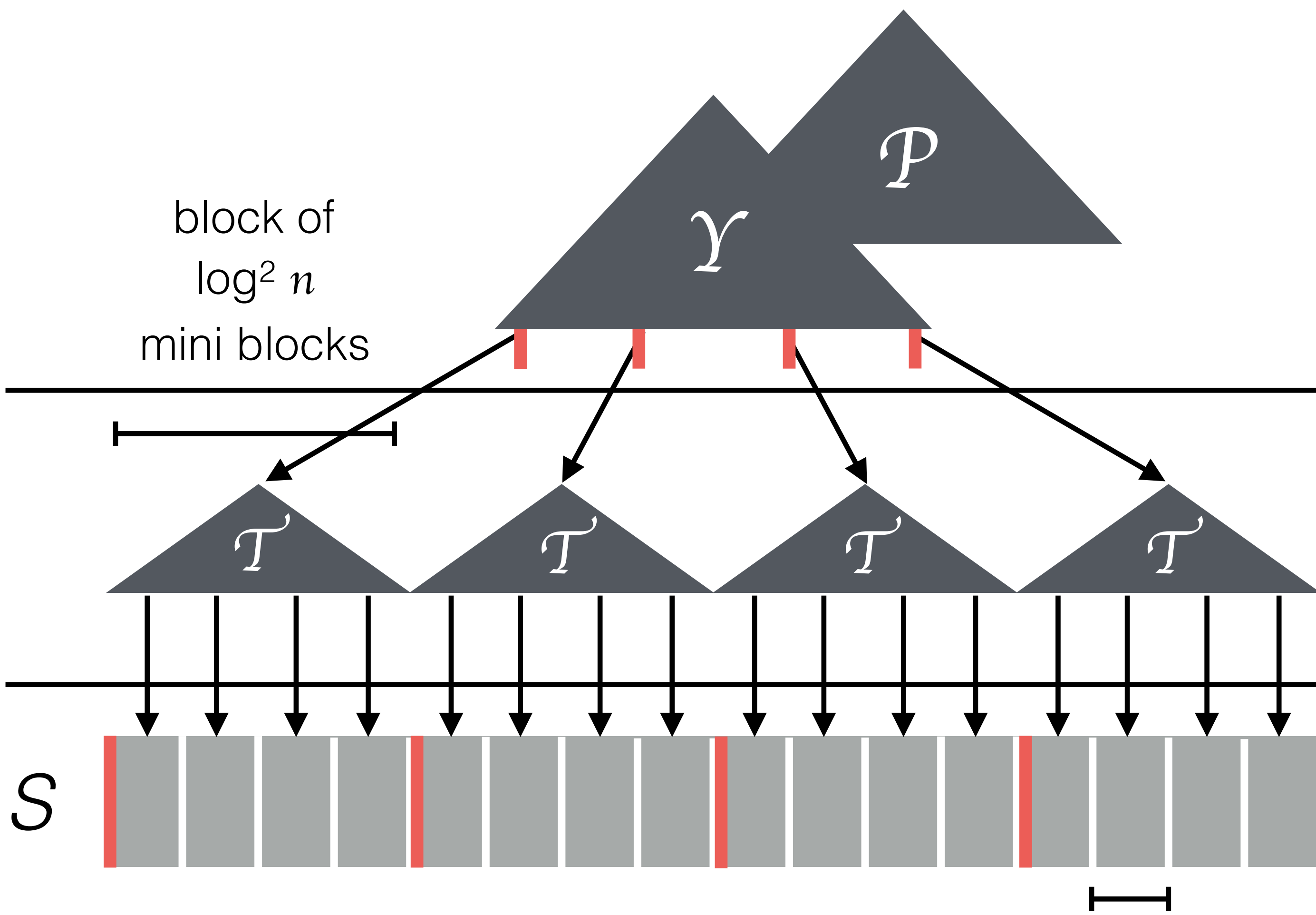
mini block of size
b = log n / loglog n

[1] Philip Bille, Patrick Hagge Cording, Inge Li Gørtz, Frederik Rye Skjoldjensen, Hjalte Wedel Vildhøj, and Søren Vind. *Dynamic relative compression*. *CoRR*, abs/1504.07851, 2015.

Results - Dynamic Elias-Fano

For $u = n^{\Upsilon}$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + $o(n)$ bits
- $O(\log n / \log\log n)$ Access
- $O(\log n / \log\log n)$ Insert/Delete (amortized)
- $O(\min\{1+\log(u/n), \log\log n\})$ Predecessor



Υ is an Y-fast trie

- $O(\log\log n)$ time

$\mathcal{P}$ is a dynamic prefix-sums DS [1]

- O(b) time

$O(n / (b \times \log^2 n)) \times O(\log u) = o(n)$ bits each

upper level

$o(n)$ bits

$\mathcal{T}$ is a k-ary tree of constant height:

- $O(\log\log n)$ time
- $O(\log^2 n \log\log n)$ bits

lower level

$o(n)$ bits

EF($S(n,u)$) = $n \log(u/n) + 2n$ bits

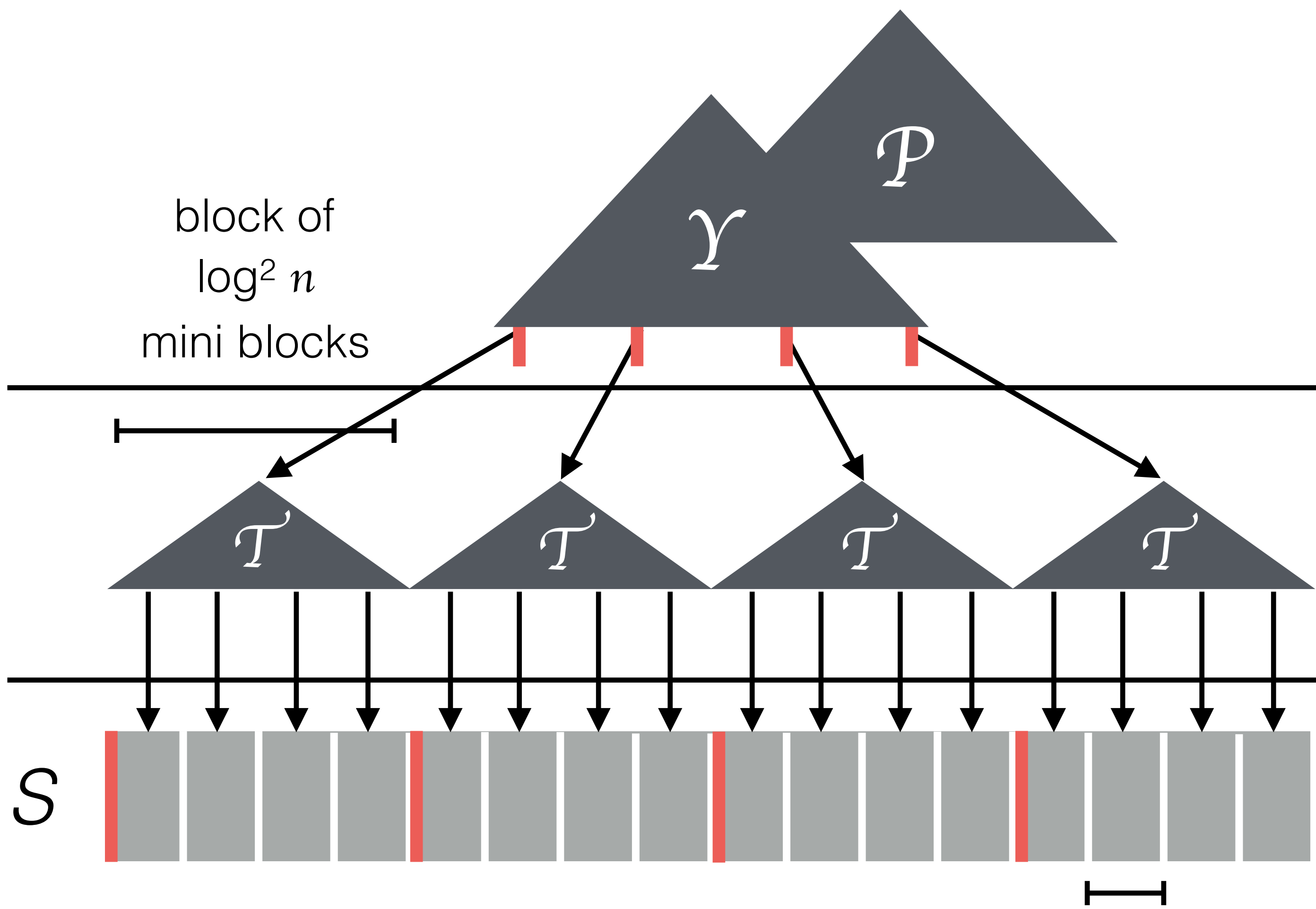
mini block of size
b = $\log n / \log\log n$

[1] Philip Bille, Patrick Hagge Cording, Inge Li Gørtz, Frederik Rye Skjoldjensen, Hjalte Wedel Vildhøj, and Søren Vind. *Dynamic relative compression*. *CoRR*, abs/1504.07851, 2015.

Results - Dynamic Elias-Fano

For $u = n^{\Upsilon}$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + o(n) bits
- O(log n / loglog n) Access
- O(log n / loglog n) Insert/Delete (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor



Υ is an Y-fast trie
- O(loglog n) time

$\mathcal{P}$ is a dynamic prefix-sums DS [1]
- O(b) time

O(n / (b x log² n)) x O(log u) = o(n) bits each

upper level

o(n) bits

$\mathcal{T}$ is a k-ary tree of constant height:
- O(loglog n) time
- O(log² n loglog n) bits

lower level

o(n) bits

mini block of size
b = log n / loglog n

EF($S(n,u)$) = n log(u/n) + 2n bits

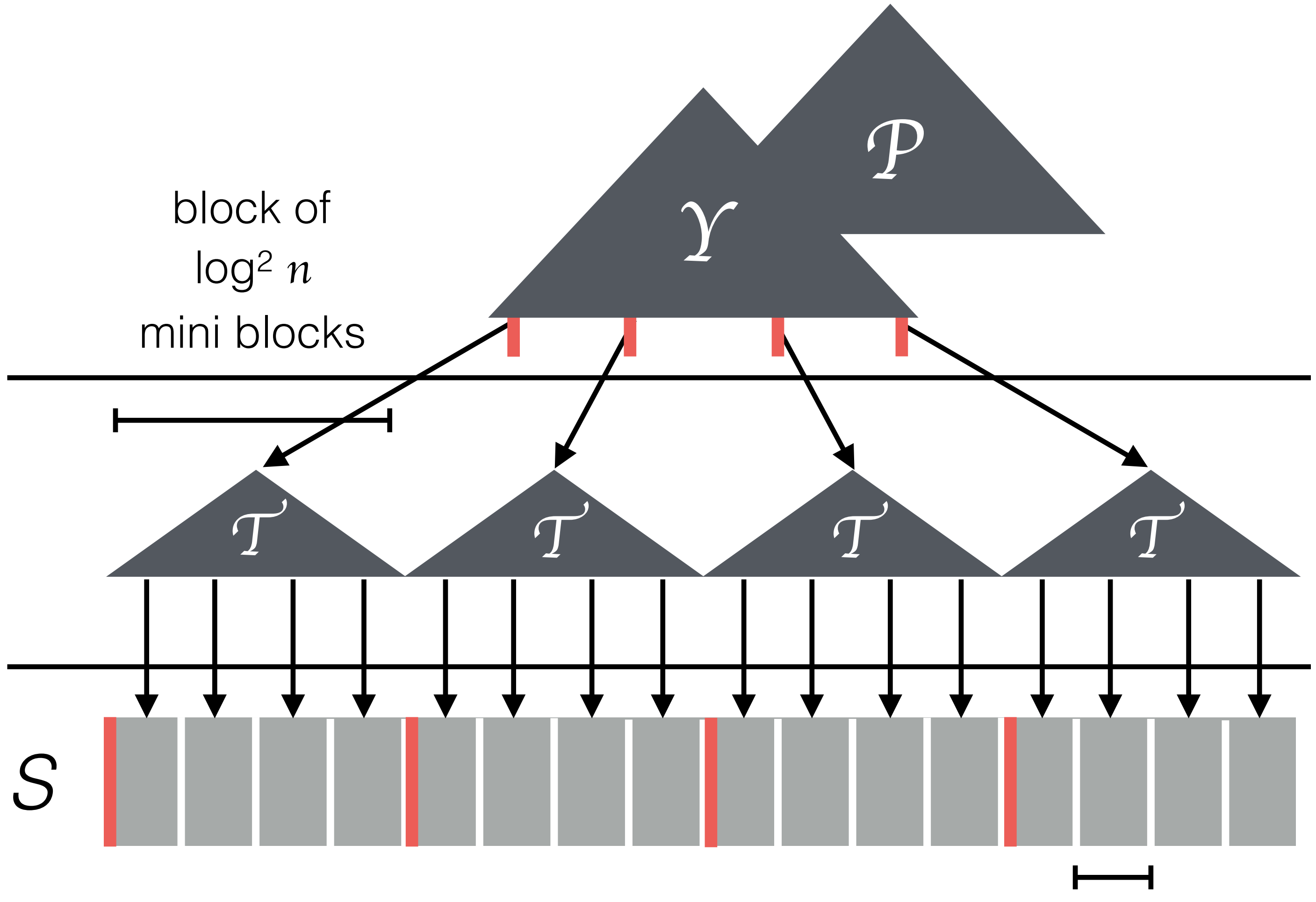
The encoding of the mini blocks takes ≤ EF($S(n,u)$) + o(n) **bits**

[1] Philip Bille, Patrick Hagge Cording, Inge Li Gørtz, Frederik Rye Skjoldjensen, Hjalte Wedel Vildhøj, and Søren Vind. *Dynamic relative compression*. *CoRR*, abs/1504.07851, 2015.

Results - Dynamic Elias-Fano

For $u = n^\gamma$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + o(n) bits
- O(log n / loglog n) Access
- O(log n / loglog n) Insert/Delete (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor



EF($S(n,u)$) = n log(u/n) + 2n bits

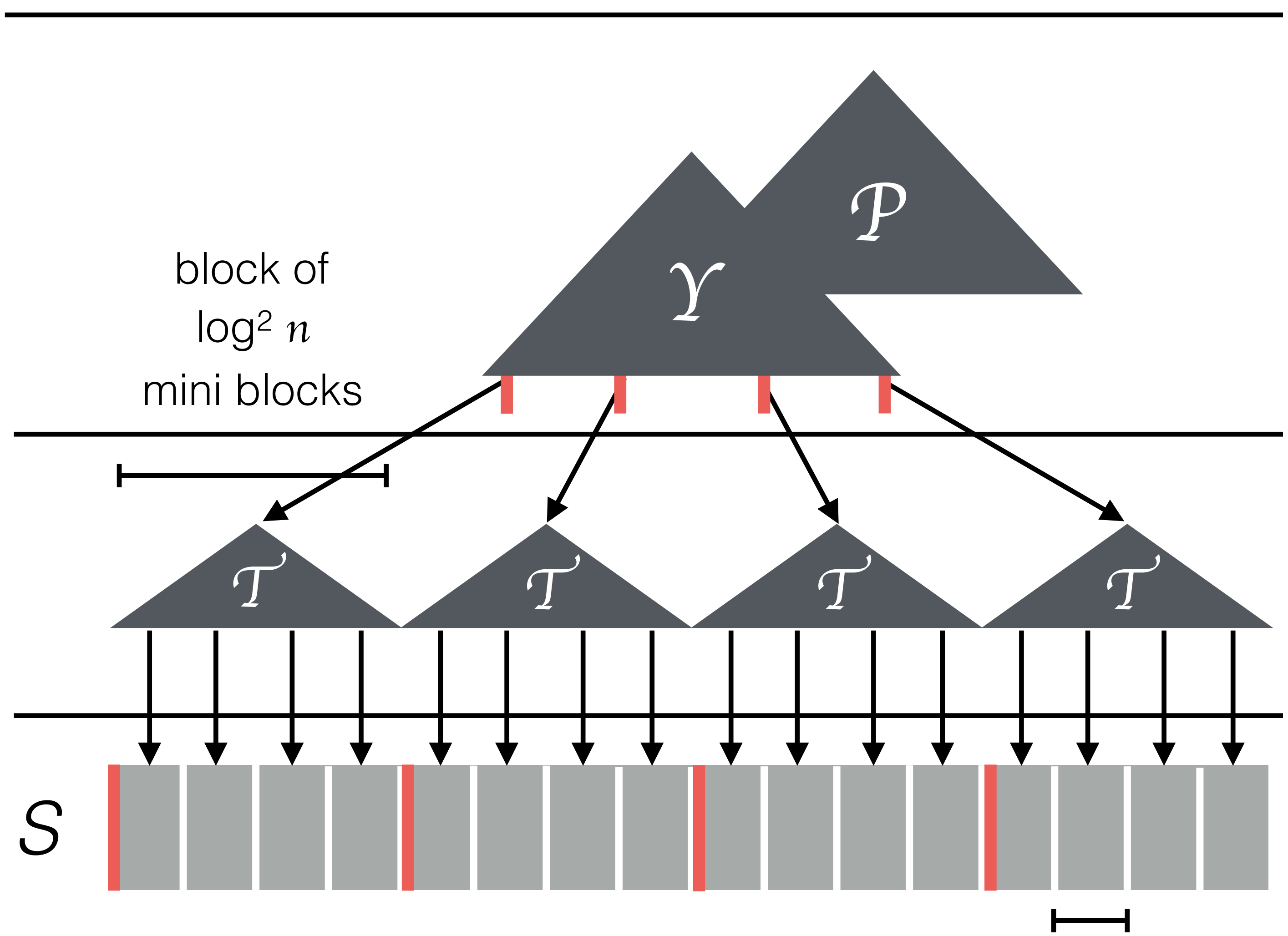
The encoding of the mini blocks takes ≤ EF($S(n,u)$) + o(n) bits

[1] Philip Bille, Patrick Hagge Cording, Inge Li Gørtz, Frederik Rye Skjoldjensen, Hjalte Wedel Vildhøj, and Søren Vind. *Dynamic relative compression*. *CoRR*, abs/1504.07851, 2015.

Results - Dynamic Elias-Fano 3

For $u = n^{\gamma}$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + o(n) bits
- O(log n / loglog n) Access
- O(log n / loglog n) Insert/Delete (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor



EF($S(n,u)$) = n log(u/n) + 2n bits

mini block of size
b = log n / loglog n

The encoding of the mini blocks takes ≤ EF($S(n,u)$) + o(n) bits

Memory management for the mini blocks:

LOW HIGH

b log(u/b) + 2b bits

Corollary 3 from [3]: random Access in O(1).

Theorem 6 from [2]: address and allocate the high part of a mini block in O(1).

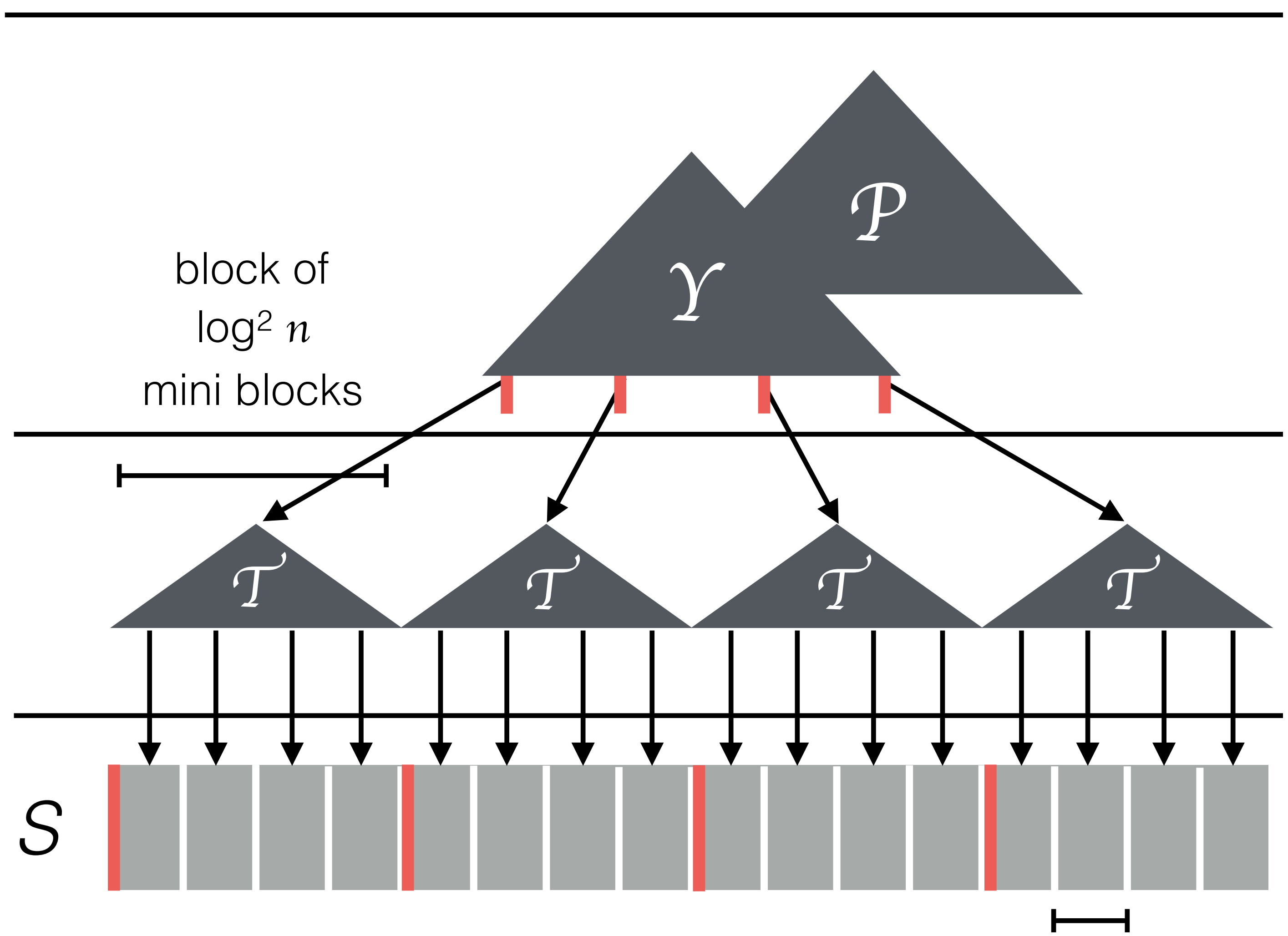
The overall redundancy is o(n) bits.

[1] Philip Bille, Patrick Hagge Cording, Inge Li Gørtz, Frederik Rye Skjoldjensen, Hjalte Wedel Vildhøj, and Søren Vind. *Dynamic relative compression. CoRR*, abs/1504.07851, 2015.

Results - Dynamic Elias-Fano

For $u = n^\gamma$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + o(n) bits
- O(log n / loglog n) Access
- O(log n / loglog n) Insert/Delete (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor



mini block of size
b = log n / loglog n

EF($S(n,u)$) = n log(u/n) + 2n bits

The encoding of the mini blocks takes ≤ EF($S(n,u)$) + o(n) bits

Memory management for the mini blocks:

LOW HIGH

b log(u/b) + 2b bits

Corollary 3 from [3]: random Access in O(1).

Theorem 6 from [2]: address and allocate the high part of a mini block in O(1).

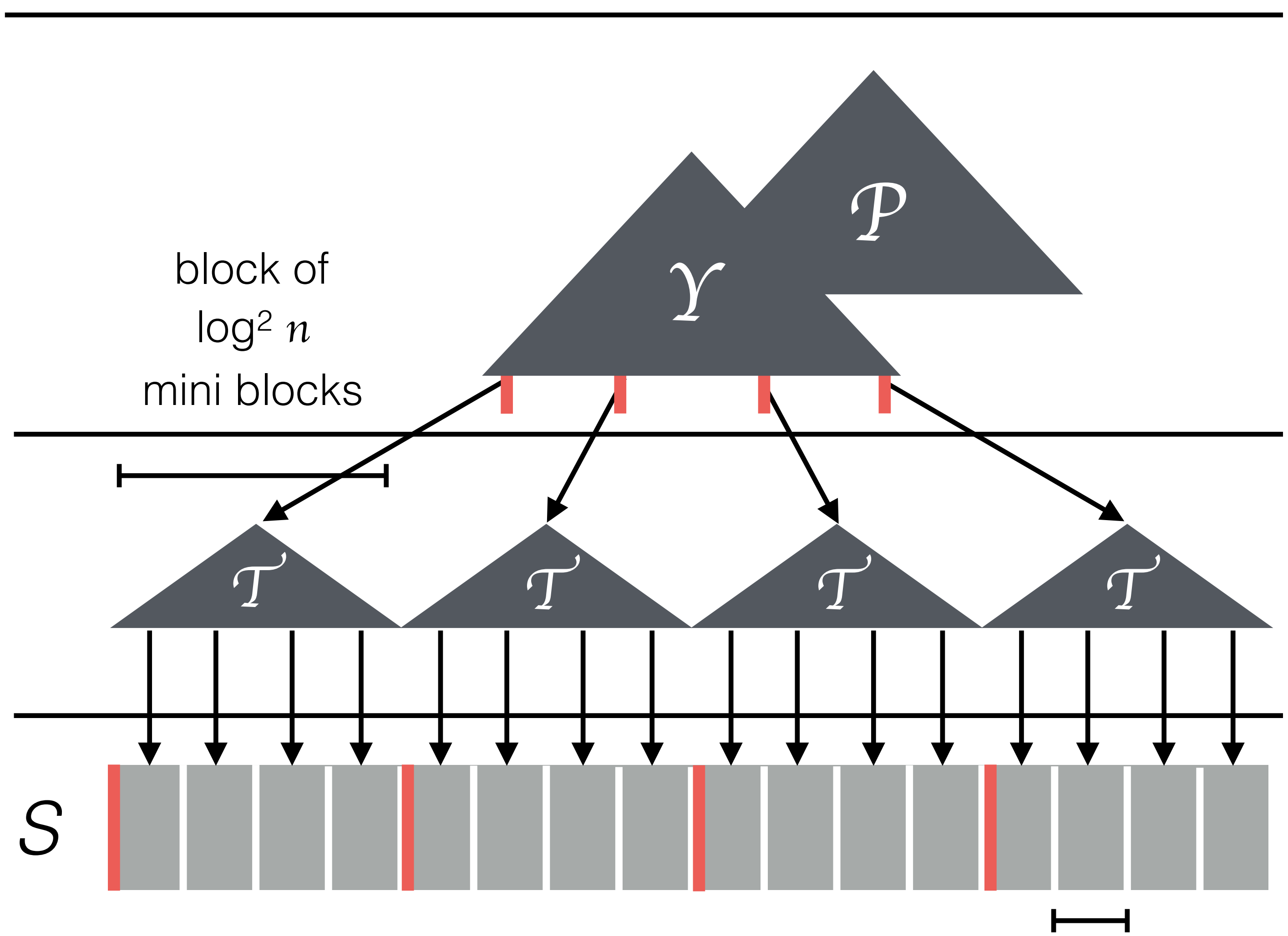
The overall redundancy is o(n) bits.

[1] Philip Bille, Patrick Hagge Cording, Inge Li Gørtz, Frederik Rye Skjoldjensen, Hjalte Wedel Vildhøj, and Søren Vind. *Dynamic relative compression. CoRR*, abs/1504.07851, 2015.
[2] J. Jansson, K. Sadakane, and Wing-Kin Sung. *CRAM: Compressed random access memory*. ICALP 2012.
[3] R. Raman, V. Raman, and S. Srinivasa Rao. *Succinct dynamic data structures.* WADS 2001.

Results - Dynamic Elias-Fano

For $u = n^{\Upsilon}$, $\gamma = \Theta(1)$:

- EF($S(n,u)$) + o(n) bits
- O(log n / loglog n) Access
- O(log n / loglog n) Insert/Delete (amortized)
- O(min{1+log(u/n), loglog n}) Predecessor



EF($S(n,u)$) = n log(u/n) + 2n bits

The encoding of the mini blocks takes ≤ EF($S(n,u)$) + o(n) bits

Memory management for the mini blocks:

LOW: b log(u/b) + HIGH: 2b bits

Corollary 3 from [3]: random Access in O(1).

Theorem 6 from [2]: address and allocate the high part of a mini block in O(1).

The overall redundancy is o(n) bits.

Property

EF(S_1) + EF(S_2) ≤ EF(S)

[1] Philip Bille, Patrick Hagge Cording, Inge Li Gørtz, Frederik Rye Skjoldjensen, Hjalte Wedel Vildhøj, and Søren Vind. *Dynamic relative compression. CoRR*, abs/1504.07851, 2015.
[2] J. Jansson, K. Sadakane, and Wing-Kin Sung. *CRAM: Compressed random access memory*. ICALP 2012.
[3] R. Raman, V. Raman, and S. Srinivasa Rao. *Succinct dynamic data structures*. WADS 2001.

Thanks for your attention, time, patience!

Any questions?