Understanding the Inverse Ackermann Function

Raimund Seidel

Universität des Saarlandes

Ackermann function - Wikipedia, the free encyclopedia - Mozilla Firefox

Datei Bearbeiten Ansicht Gehe Lesezeichen Extras Hilfe

http://en.wikipedia.org/wiki/Ackerman's_function

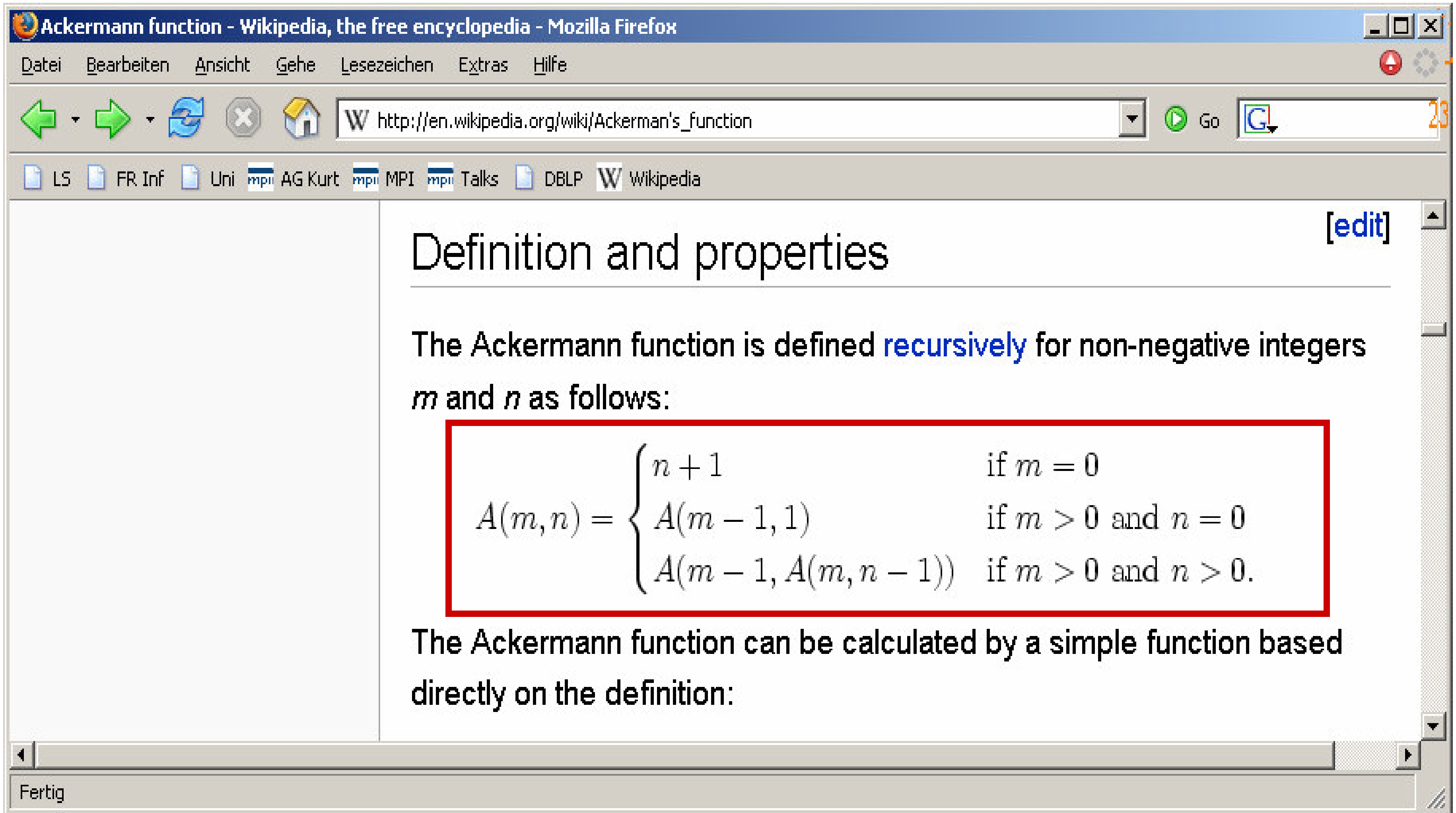
LS FR Inf Uni AG Kurt MPI Talks DBLP Wikipedia

A two-parameter variation of the inverse Ackermann function can be defined as follows:

$$\alpha(m,n) = \min\{i \geq 1 : A(i, \lfloor m/n \rfloor) \geq \log_2 n\}.$$

This function arises in more precise analyses of the algorithms mentioned above, and gives a more refined time bound. In the disjoint-set data structure, *m* represents the number of operations while *n* represents the number of elements; in the minimum spanning tree algorithm, *m* represents the number of edges while *n* represents the number of vertices. Several slightly different definitions of α(*m*, *n*) exist; for example, $\log_2 n$ is sometimes replaced by *n*, and the floor function is sometimes replaced by a ceiling.

Fertig


Ackermann function - Wikipedia, the free encyclopedia - Mozilla Firefox
Datei Bearbeiten Ansicht Gehe Lesezeichen Extras Hilfe
http://en.wikipedia.org/wiki/Ackerman's_function
Go
LS FR Inf Uni AG Kurt MPI Talks DBLP Wikipedia
[edit]
Definition and properties
The Ackermann function is defined recursively for non-negative integers m and n as follows:
$$A(m,n) = \begin{cases} n+1 & \text{if } m = 0 \\ A(m-1,1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1, A(m,n-1)) & \text{if } m > 0 \text{ and } n > 0. \end{cases}$$
The Ackermann function can be calculated by a simple function based directly on the definition:
Fertig

I am not smart enough to understand this easily.

I am not smart enough to understand this easily.

I am not smart enough to come up with proofs (or even reproduce proofs) involving the inverse Ackermann function

based on this definition.

What do I tell my students ?

What do I tell my students ?

$A(m,n)$ grows veeeeery quickly ....

$\alpha(m,n)$ grows veeeeery slowly ....

What do I tell my students ?

$A(m,n)$ grows veeeeery quickly ....

$\alpha(m,n)$ grows veeeeery slowly ....

Let's move on to the next subject !

Goal of this talk:

Goal of this talk:

Convince, that $\alpha()$ is not that complicated after all.

Goal of this talk:

Convince, that $\alpha()$ is not that complicated after all.

2 examples,

where $\alpha()$ arises naturally out of the analysis;

the Ackermann function $A()$ need not be mentioned;

top-down approach;

Goal of this talk:

Convince, that $\alpha()$ is not that complicated after all.

2 examples,

where $\alpha()$ arises naturally out of the analysis;

the Ackermann function $A()$ need not be mentioned;

top-down approach;

Partial sum problem in the semi-group setting

Goal of this talk:

Convince, that $\alpha()$ is not that complicated after all.

2 examples,

where $\alpha()$ arises naturally out of the analysis;

the Ackermann function $A()$ need not be mentioned;

top-down approach;

Partial sum problem in the semi-group setting

Union Find with Path Compression

Divide-and-Conquer Recurrences, Baby Version

Divide-and-Conquer Recurrences, Baby Version

Typical Divide-and-Conquer:

If problem set S has size $n=1$, then nothing to be done.

Otherwise:

* partition S into subproblems of size $< f(n)$
* solve each of the $n/f(n)$ subproblems recursively
* combine subsolutions.

Divide-and-Conquer Recurrences, Baby Version

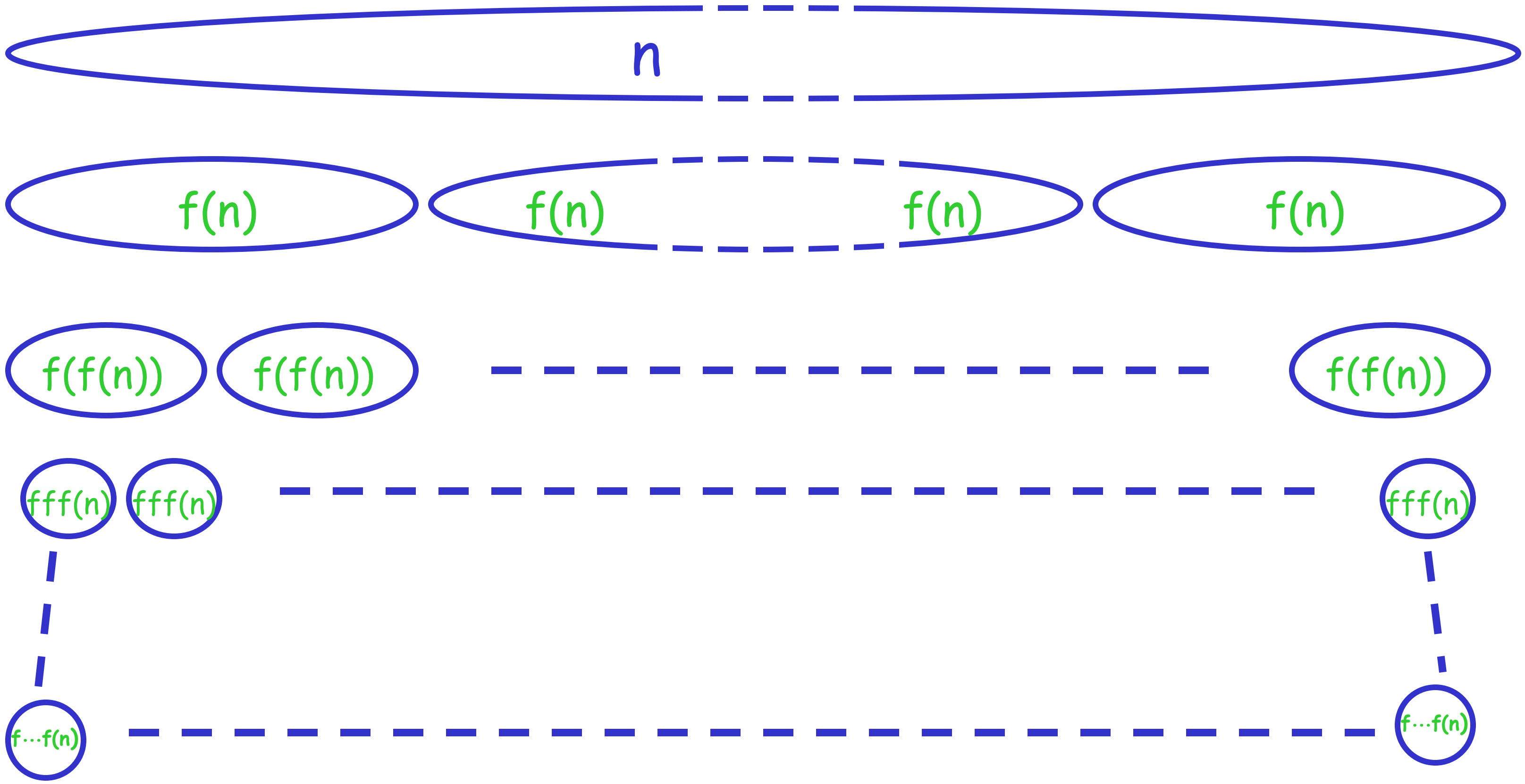
Typical Divide-and-Conquer:

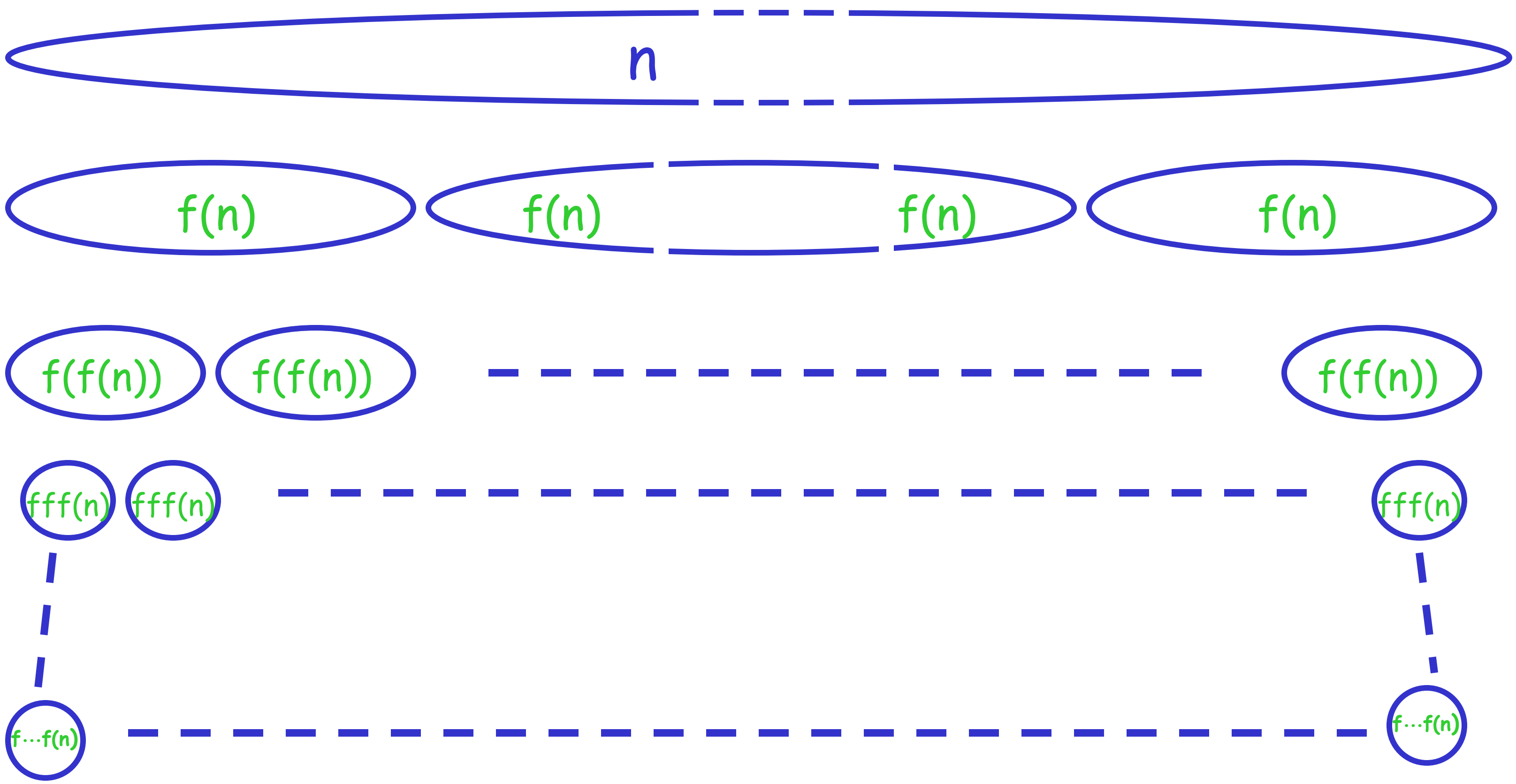
If problem set S has size $n=1$, then nothing to be done.

Otherwise:

* partition S into subproblems of size $< f(n)$
* solve each of the $n/f(n)$ subproblems recursively
* combine subsolutions.

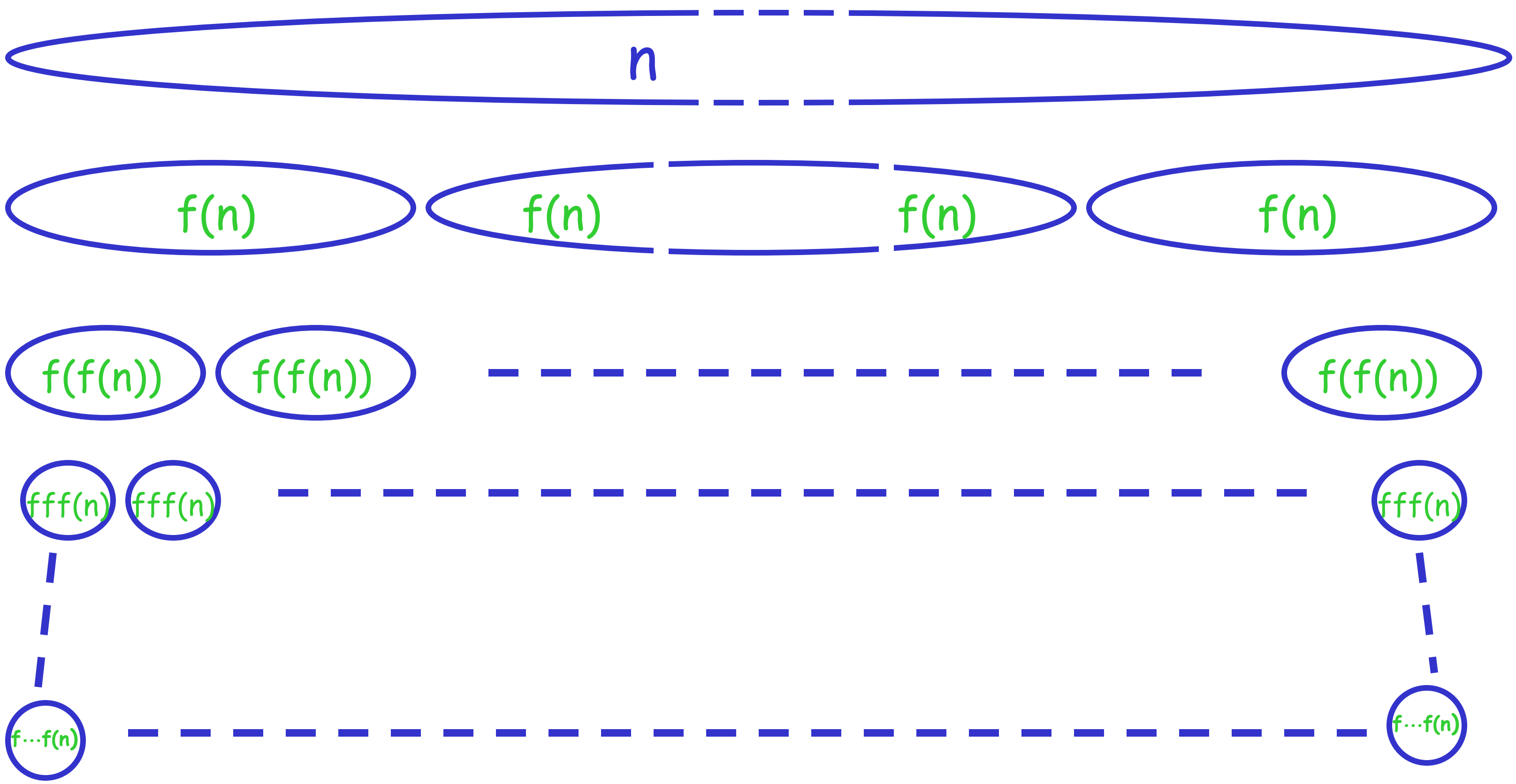
(f needs to satisfy contraction condition $f(n)<n$ for $n>1$.)


n
f(n)
f(n)
f(n)
f(n)
f(f(n))
f(f(n))
f(f(n))
fff(n)
fff(n)
fff(n)
f⋯f(n)
f⋯f(n)



Recurrence:

$$X(n) \le \begin{cases} 0 & \text{if } n \le 1 \\ a \cdot n + \dfrac{n}{f(n)} \cdot X(f(n)) & \text{if } n > 1 \end{cases}$$



Recurrence:
$$X(n) \le \begin{cases} 0 & \text{if } n \le 1 \\ a \cdot n + \dfrac{n}{f(n)} \cdot X(f(n)) & \text{if } n > 1 \end{cases}$$

Solution: $X(n) \le a \cdot n \cdot f^*(n)$

$$f^*(n) = \begin{cases} 0 & \text{if } n \leq 1 \\ 1 + f^*(\ f(n)\ ) & \text{if } n > 1 \end{cases}$$

$$f^*(n) = \begin{cases} 0 & \text{if } n \leq 1 \\ 1 + f^*(\,f(n)\,) & \text{if } n > 1 \end{cases}$$

$$f^*(n) = \min \{\, k \mid \underbrace{f(f(\cdots\cdots f(n)\cdots)}_{k \text{ times}} \leq 1 \,\}$$

$$f^*(n) = \begin{cases} 0 & \text{if } n \le 1 \\ 1 + f^*(\,f(n)\,) & \text{if } n > 1 \end{cases}$$

$$f^*(n) = \min \{\, k \mid \underbrace{f(f(\cdots\cdots f}_{k \text{ times}}(n)\cdots) \le 1 \,\}$$

Properties:

1) $f^*(\,f(n)\,) = f^*(n) - 1$

2) f a "nice" compaction
$\Rightarrow f^*$ a "nice" compaction and
f^* "much smaller" than f

Examples for f^* :

$f(n)$	$f^*(n)$
$n-1$	$n-1$
$n-2$	$n/2$
$n-c$	n/c
$n/2$	$\log_2 n$
n/c	$\log_c n$
$\sqrt{n}$	$\log \log n$
$\log n$	$\log^* n$

Partial sum problem in the semi-group setting

Data: $A_1, A_2, \cdots, A_n \in$ "Semigroup" $(G,+)$

Query: i,j **Answer**: $A_i+A_{i+1}+ \cdots +A_j$

"partial sum"

Partial sum problem in the semi-group setting

Data: $A_1, A_2, \cdots, A_n \in$ "Semigroup" $(G,+)$

Query: i,j **Answer**: $A_i+A_{i+1}+\cdots+A_j$

"partial sum"

Goal: Store "few" values of G so that each query can be answered with little cost

Partial sum problem in the semi-group setting

Data: $A_1, A_2, \cdots, A_n \in$ "Semigroup" $(G,+)$

Query: i,j **Answer**: $A_i + A_{i+1} + \cdots + A_j$

"partial sum"

Goal: Store "few" values of G so that each query can be answered with little cost

of "+" operations

Partial sum problem in the semi-group setting

Data: $A_1, A_2, \cdots, A_n \in$ "Semigroup" $(G,+)$

Query: i,j **Answer**: $A_i + A_{i+1} + \cdots + A_j$

"partial sum"

Goal: Store "few" values of G so that each query can be answered with little cost

\# of "+" operations

$S_k(n)$ = # of values to be stored so that every query can be answered using at most k "+" operations.

Partial sum problem in the semi-group setting

Data: $A_1, A_2, \cdots, A_n \in$ "Semigroup" $(G,+)$

Query: i,j **Answer**: $A_i + A_{i+1} + \cdots + A_j$

"partial sum"

Goal: Store "few" values of G so that each query can be answered with little cost

of "+" operations

$S_k(n)$ = # of values to be stored so that every query can be answered using at most k "+" operations.

$S_0(n) = \binom{n+1}{2}$

Example semi-groups $(G,+)$:

$(\mathbb{R}, \max)$

$(\mathbb{R}^n, \text{componentwise-max})$

(d×d matrices, mult)

Claim: $S_1(n) =$

Claim: $S_1(n) = n \log_2 n$

Claim: $S_1(n) = n \log_2 n$

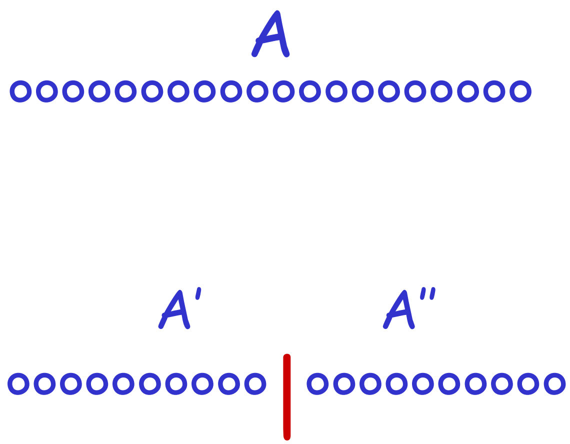
"1-op-structure"

case $n=1$: trivial

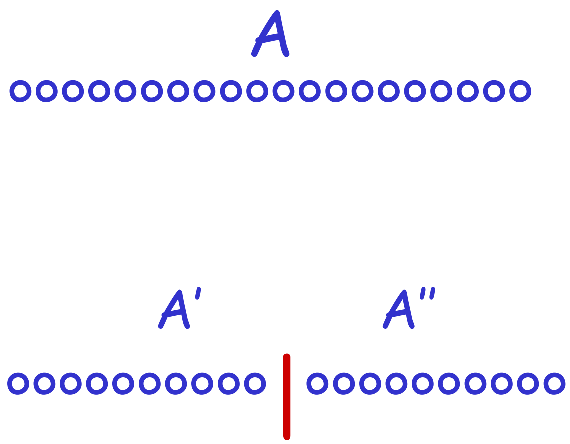
case $n \geq 2$: use recursive construction

A

partition A-sequence into 2 subsequences A' and A'' of length n/2 each

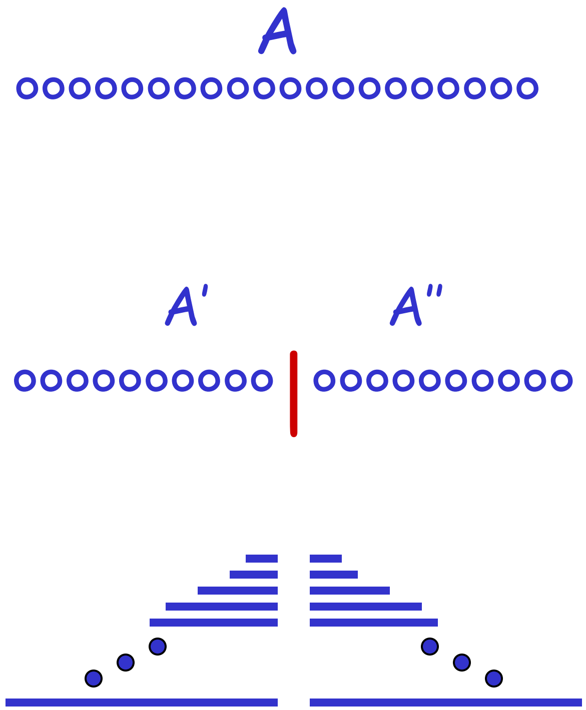


partition A-sequence into
2 subsequences A' and A''
of length n/2 each



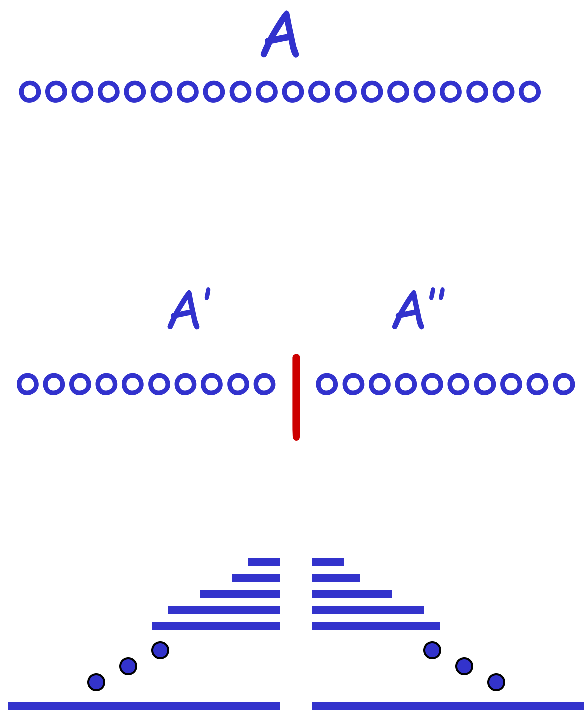
partition A-sequence into 2 subsequences A' and A'' of length $n/2$ each

store each suffix-sum of A'
store each prefix-sum of A''



partition A-sequence into
2 subsequences A' and A''
of length n/2 each

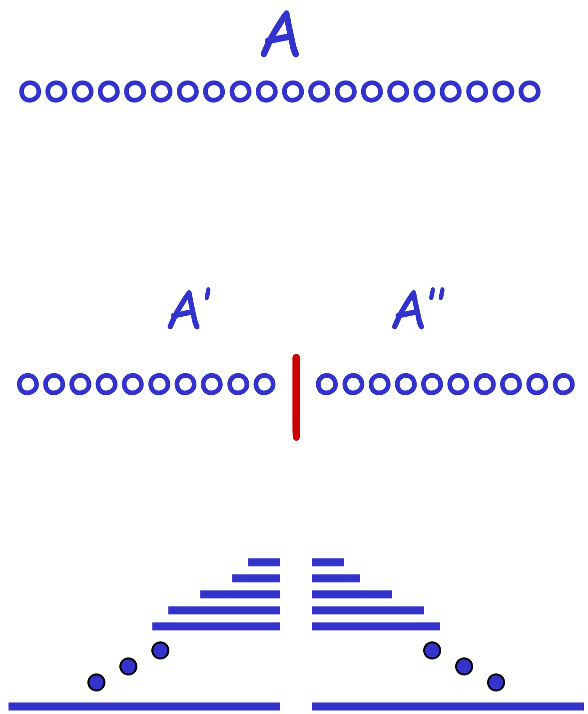
store each suffix-sum of A'
store each prefix-sum of A''



partition A-sequence into
2 subsequences A' and A''
of length $n/2$ each

store each suffix-sum of A'
store each prefix-sum of A''

recursively store a
1-op-structure for A' and a
1-op-structure for A''



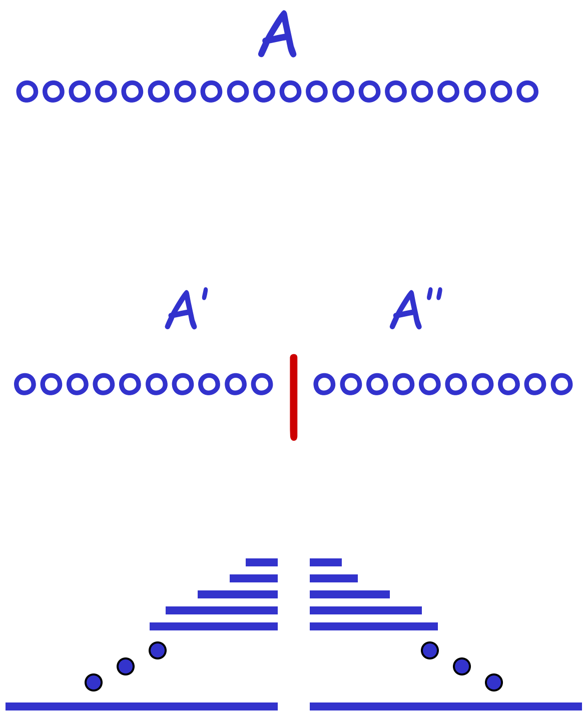
partition A-sequence into
2 subsequences A' and A''
of length $n/2$ each

store each suffix-sum of A'
store each prefix-sum of A''

recursively store a
1-op-structure for A' and a
1-op-structure for A''

Query answering:

either return (suffix-sum)+(prefix-sum)
or use one of the recursive structures

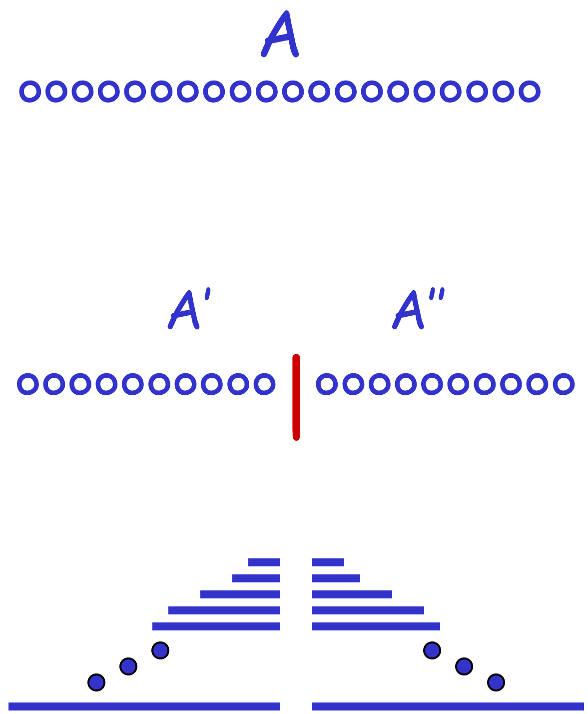


partition A-sequence into
2 subsequences A' and A''
of length $n/2$ each

store each suffix-sum of A'
store each prefix-sum of A''

recursively store a
1-op-structure for A' and a
1-op-structure for A''

$$S_1(n) \leq n + \frac{n}{(n/2)} S_1(\, n/2 \,)$$



partition *A*-sequence into
2 subsequences *A'* and *A''*
of length n/2 each

store each suffix-sum of *A'*
store each prefix-sum of *A''*

recursively store a
1-op-structure for *A'* and a
1-op-structure for *A''*

$$S_1(n) \leq n + \frac{n}{(n/2)} S_1(\, n/2 \,)$$

$$\Rightarrow \; S_1(n) \leq n \cdot (n/2)^* = n \log_2 n$$

$S_3(n) = ?$

$S_3(n) = ?$

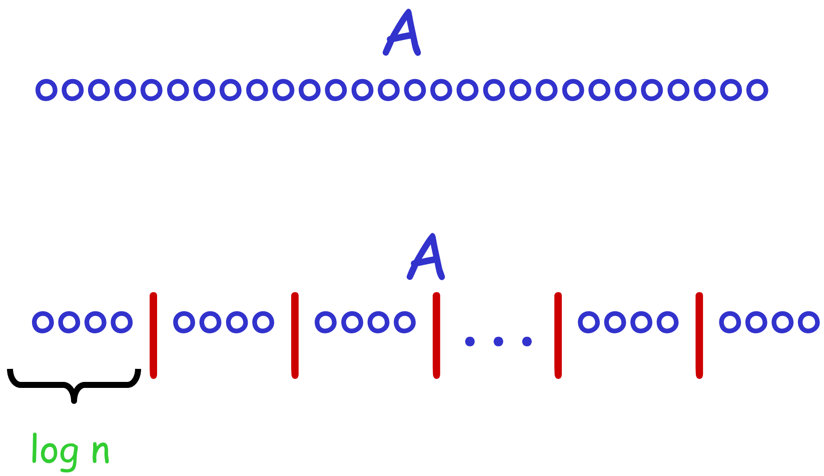
"3-op-structure"

case $n \leq 4$: trivial

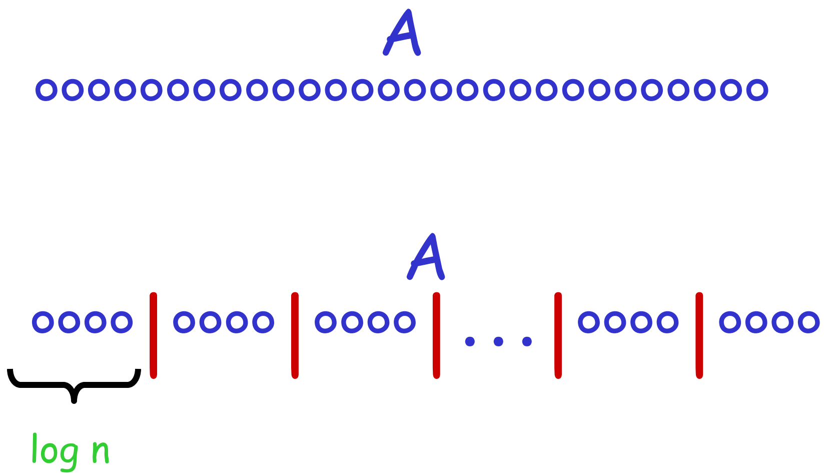
case $n \geq 5$: use recursive construction

A

partition A-sequence into
$n/\log n$ subsequences of length
$\leq \log n$ each

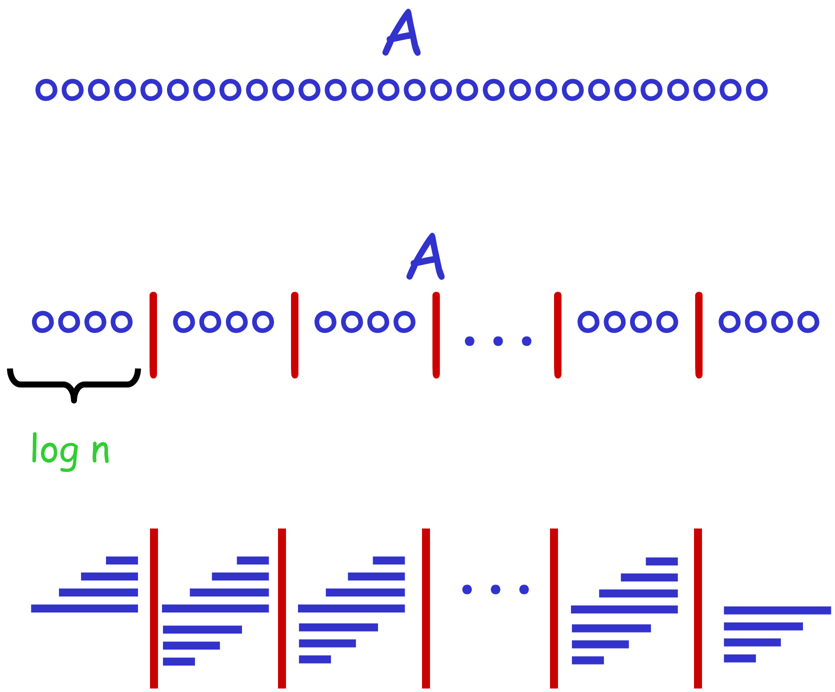


partition A-sequence into
n/log n subsequences of length
≤ log n each



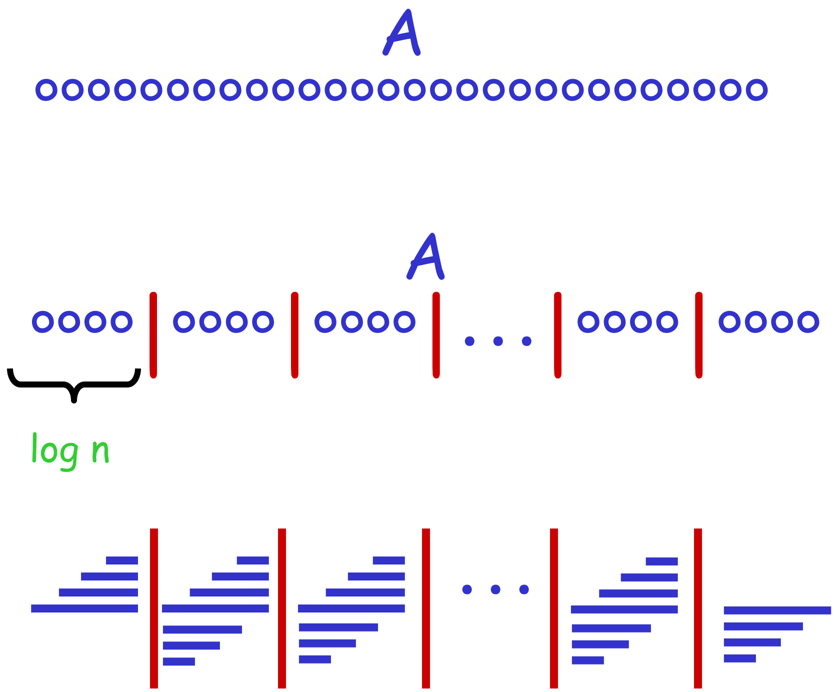
partition A-sequence into
$n/\log n$ subsequences of length
$\leq \log n$ each

store all prefix- and all suffix-
sums within each subsequence



partition A-sequence into $n/\log n$ subsequences of length $\leq \log n$ each

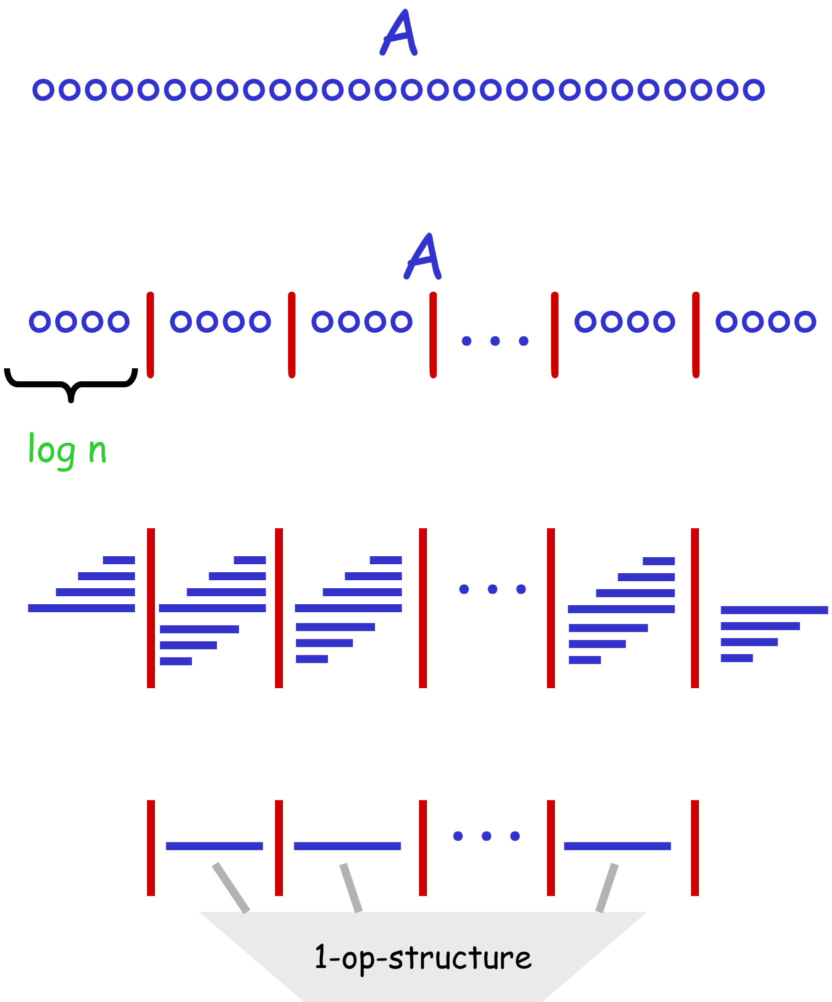
store all prefix- and all suffix-sums within each subsequence



partition *A*-sequence into n/log n subsequences of length ≤ log n each

store all prefix- and all suffix-sums within each subsequence

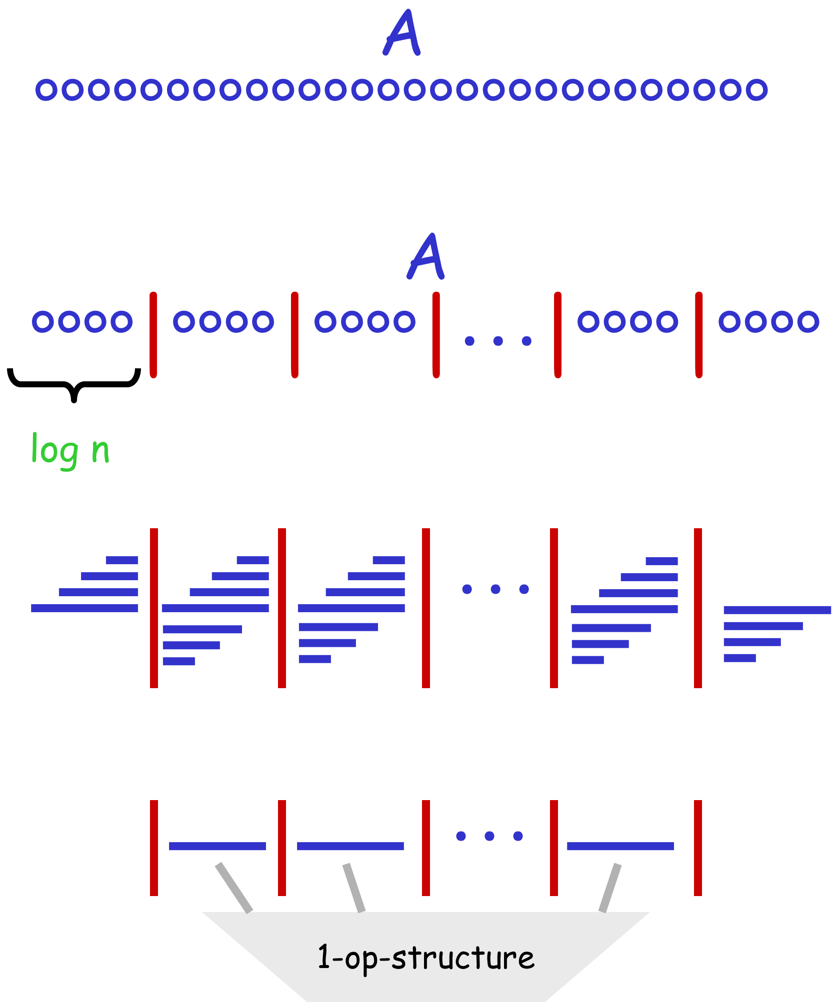
build a 1-op-structure for the n/log n subsequence-**sums**



partition A-sequence into
n/log n subsequences of length
≤ log n each

store all prefix- and all suffix-
sums within each subsequence

build a 1-op-structure for the
n/log n subsequence-**sums**

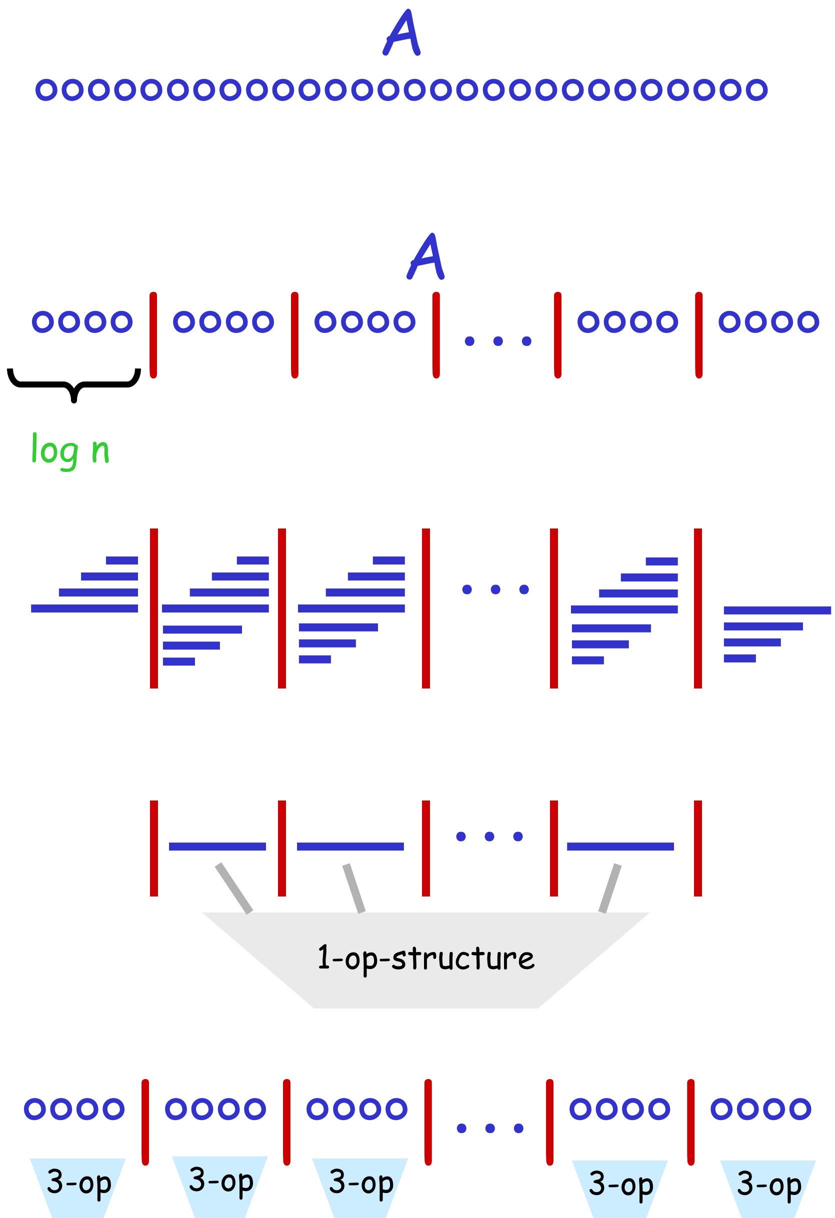


partition A-sequence into $n/\log n$ subsequences of length $\leq \log n$ each

store all prefix- and all suffix-sums within each subsequence

build a 1-op-structure for the $n/\log n$ subsequence-**sums**

recursively build a 3-op-structure for each of the $n/\log n$ subsequences

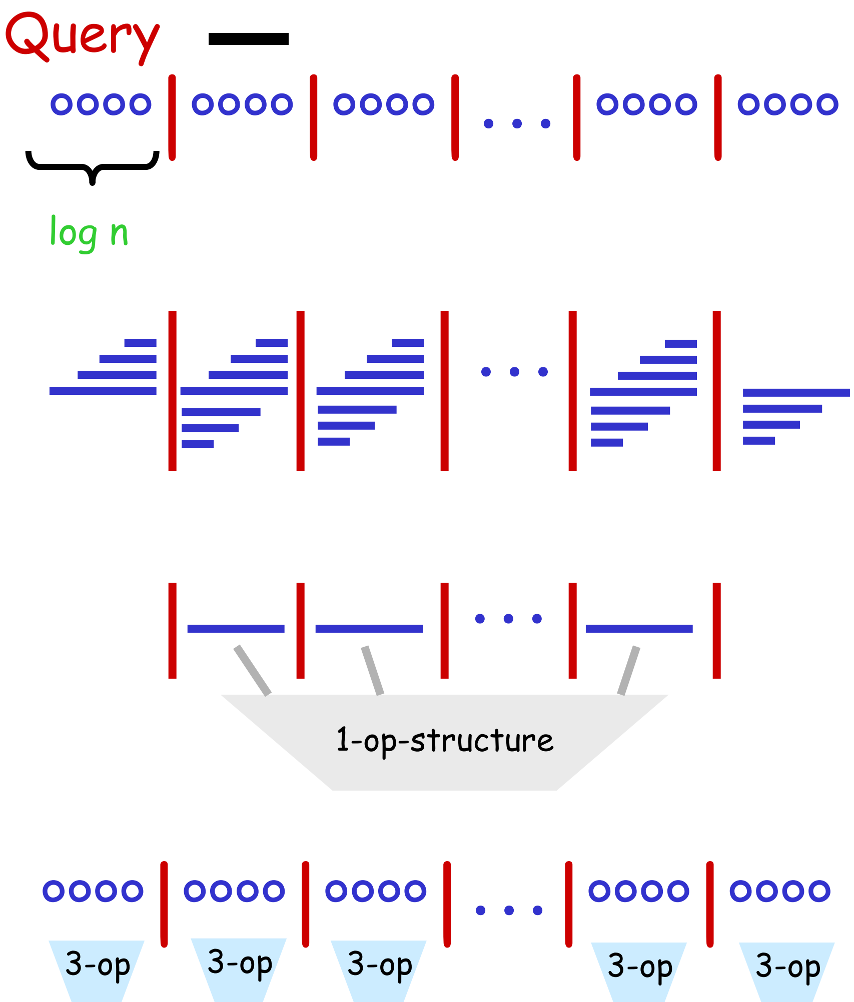


partition A-sequence into $n/\log n$ subsequences of length $\leq \log n$ each

store all prefix- and all suffix-sums within each subsequence

build a 1-op-structure for the $n/\log n$ subsequence-**sums**

recursively build a 3-op-structure for each of the $n/\log n$ subsequences



store all prefix- and all suffix-sums within each subsequence

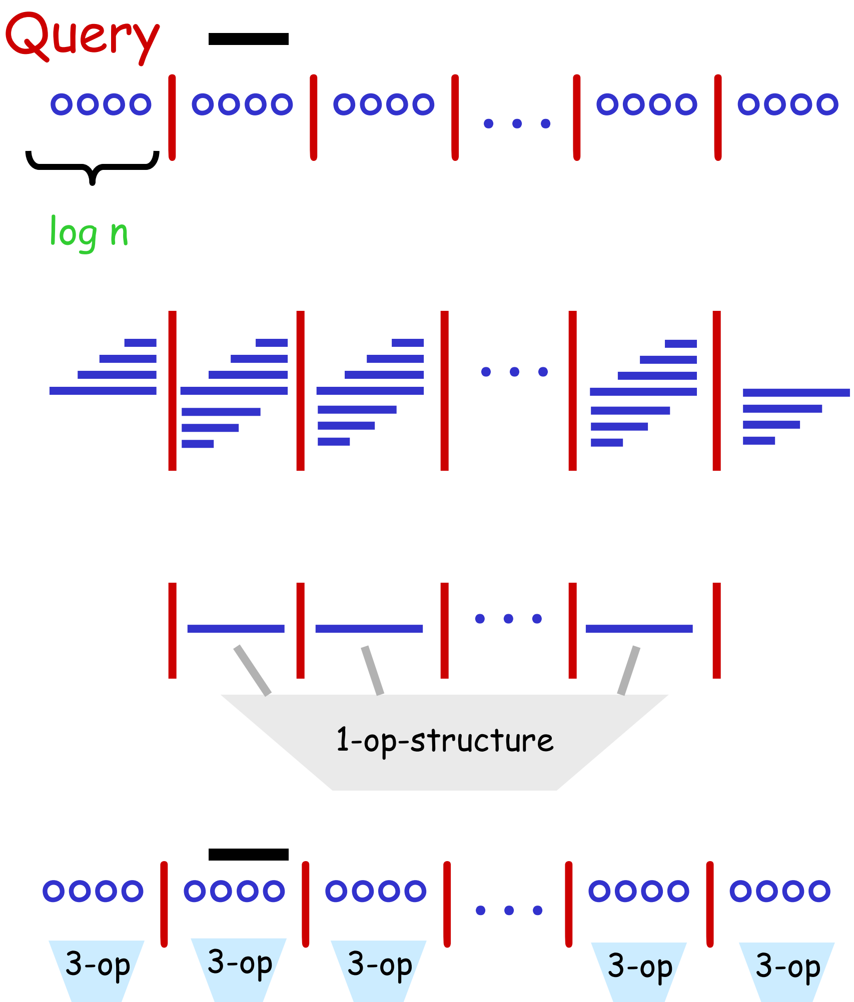
build a 1-op-structure for the n/log n subsequence-**sums**

recursively build a 3-op-structure for each of the n/log n subsequences

Query answering:

either use one of the recursive 3-op-structures

or return (suffix-sum)+(answer from 1-op-structure)+(prefix-sum)



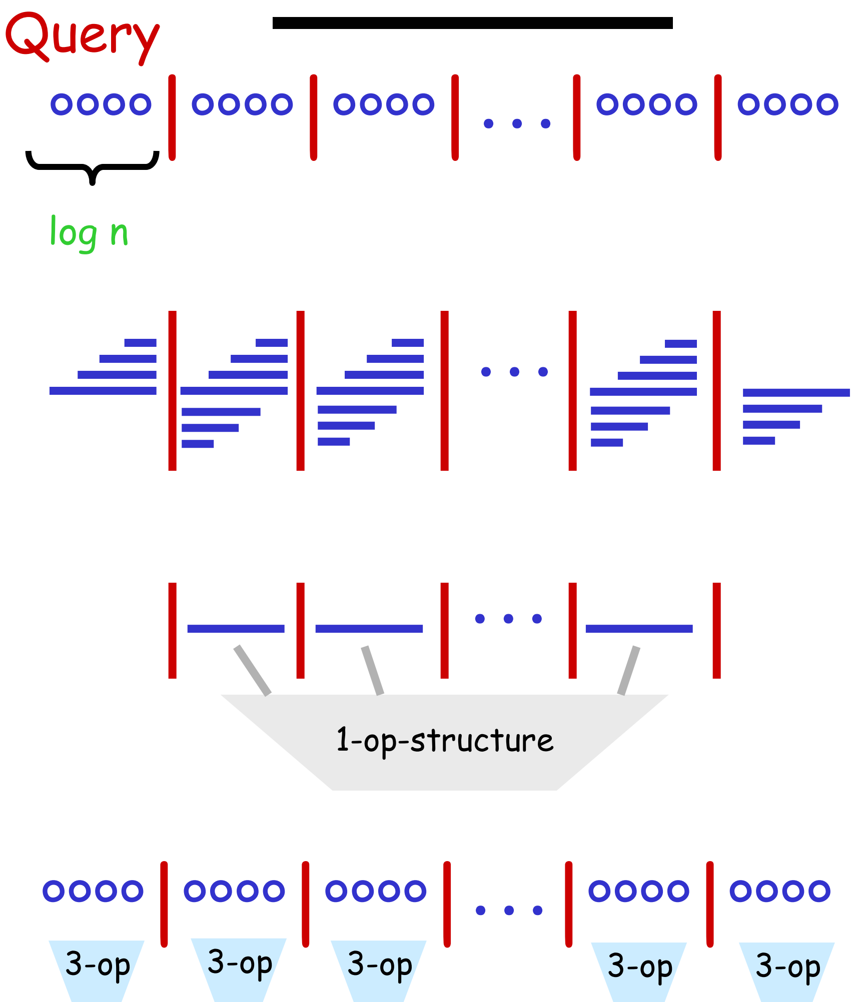
store all prefix- and all suffix-sums within each subsequence

build a 1-op-structure for the $n/\log n$ subsequence-**sums**

recursively build a 3-op-structure for each of the $n/\log n$ subsequences

Query answering:

either use one of the recursive 3-op-structures
or return (suffix-sum)+(answer from 1-op-structure)+(prefix-sum)



store all prefix- and all suffix-sums within each subsequence

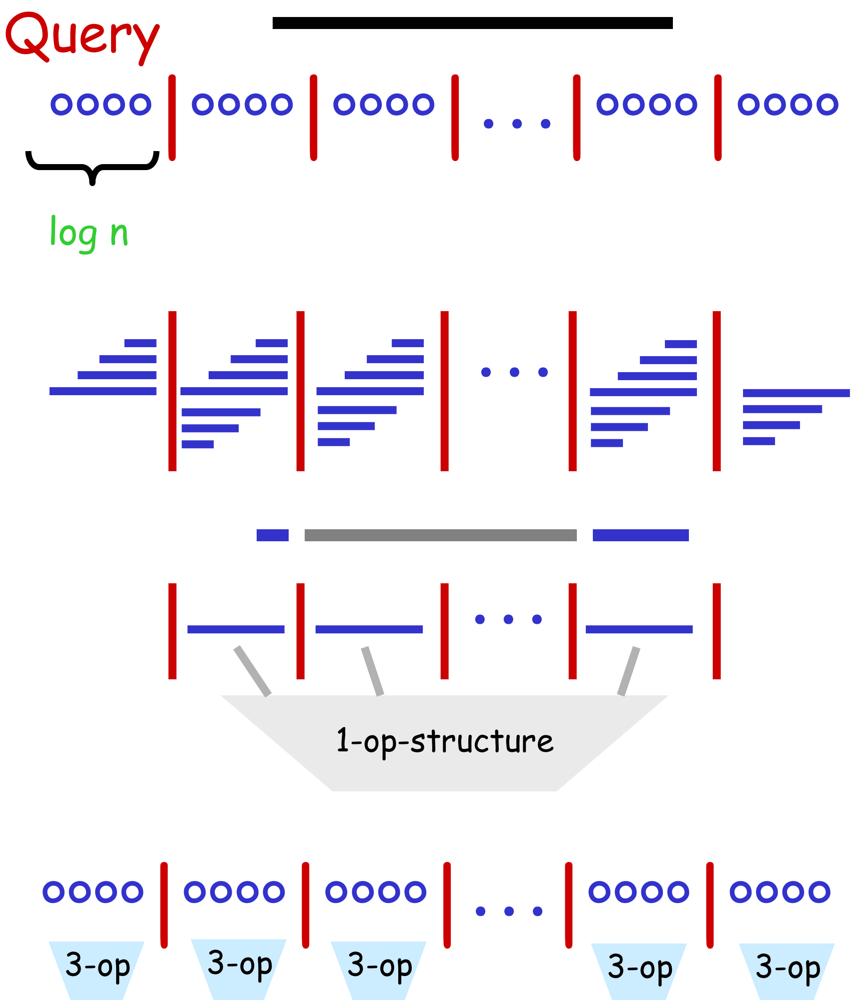
build a 1-op-structure for the $n/\log n$ subsequence-**sums**

recursively build a 3-op-structure for each of the $n/\log n$ subsequences

Query answering:

either use one of the recursive 3-op-structures

or return (suffix-sum)+(answer from 1-op-structure)+(prefix-sum)



store all prefix- and all suffix-sums within each subsequence

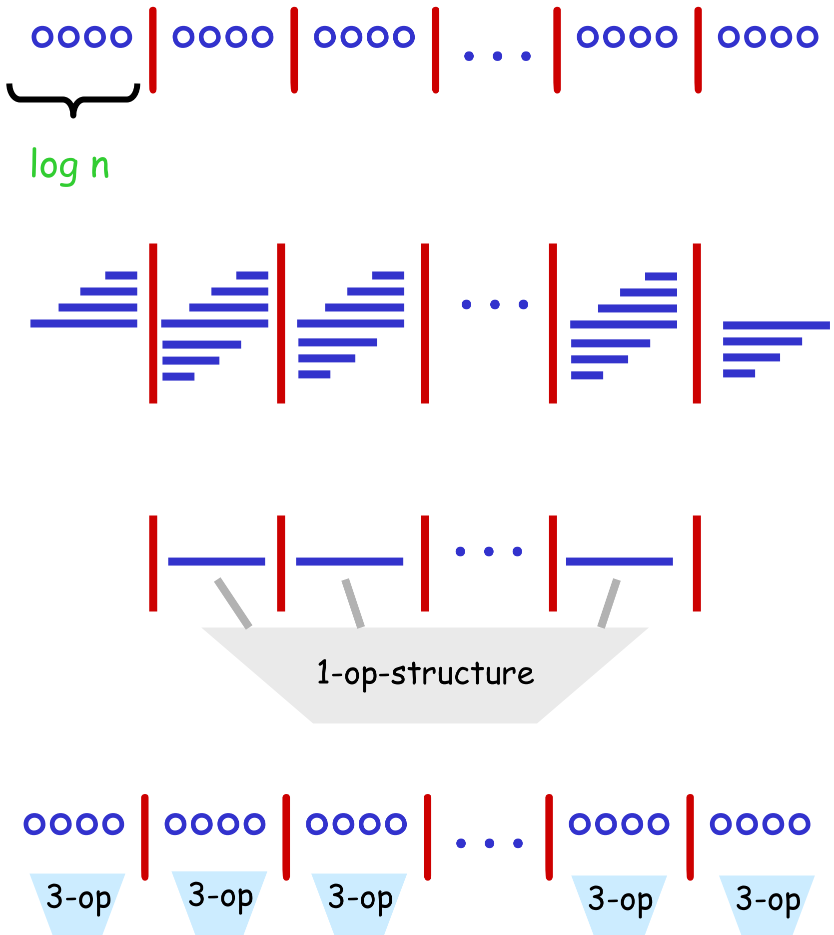
build a 1-op-structure for the n/log n subsequence-**sums**

recursively build a 3-op-structure for each of the n/log n subsequences

Query answering:

either use one of the recursive 3-op-structures

or return (suffix-sum)+(answer from 1-op-structure)+(prefix-sum)

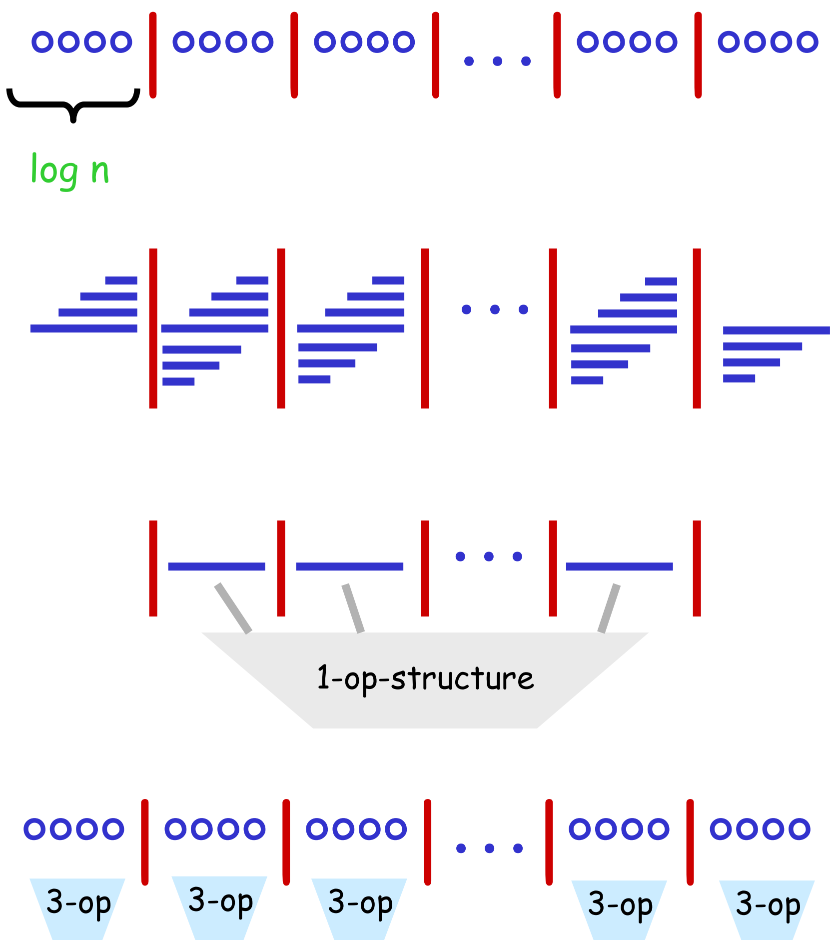


store all prefix- and all suffix-sums within each subsequence

build a 1-op-structure for the $n/\log n$ subsequence-**sums**

recursively build a 3-op-structure for each of the $n/\log n$ subsequences

$$S_3(n) \leq 2n + S_1\left(\frac{n}{\log n}\right) + \frac{n}{\log n} \cdot S_3(\log n)$$

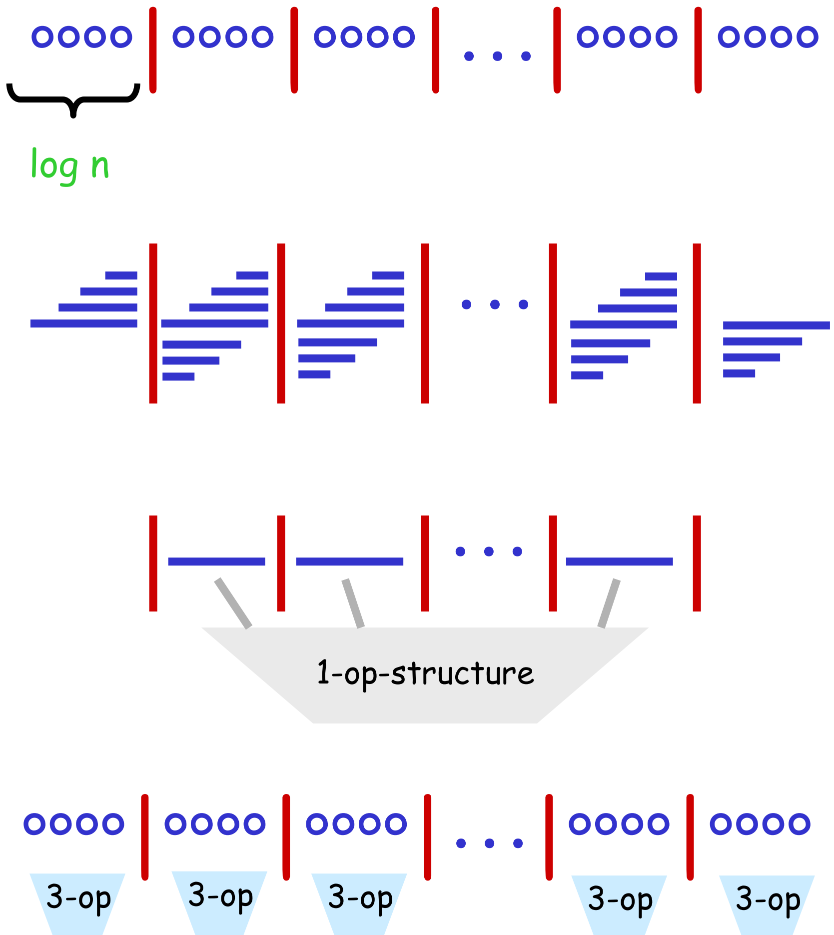


store all prefix- and all suffix-sums within each subsequence

build a 1-op-structure for the $n/\log n$ subsequence-**sums**

recursively build a 3-op-structure for each of the $n/\log n$ subsequences

$$S_3(n) \le 2n + \underbrace{S_1\left(\frac{n}{\log n}\right)}_{\le n} + \frac{n}{\log n} \cdot S_3(\log n)$$

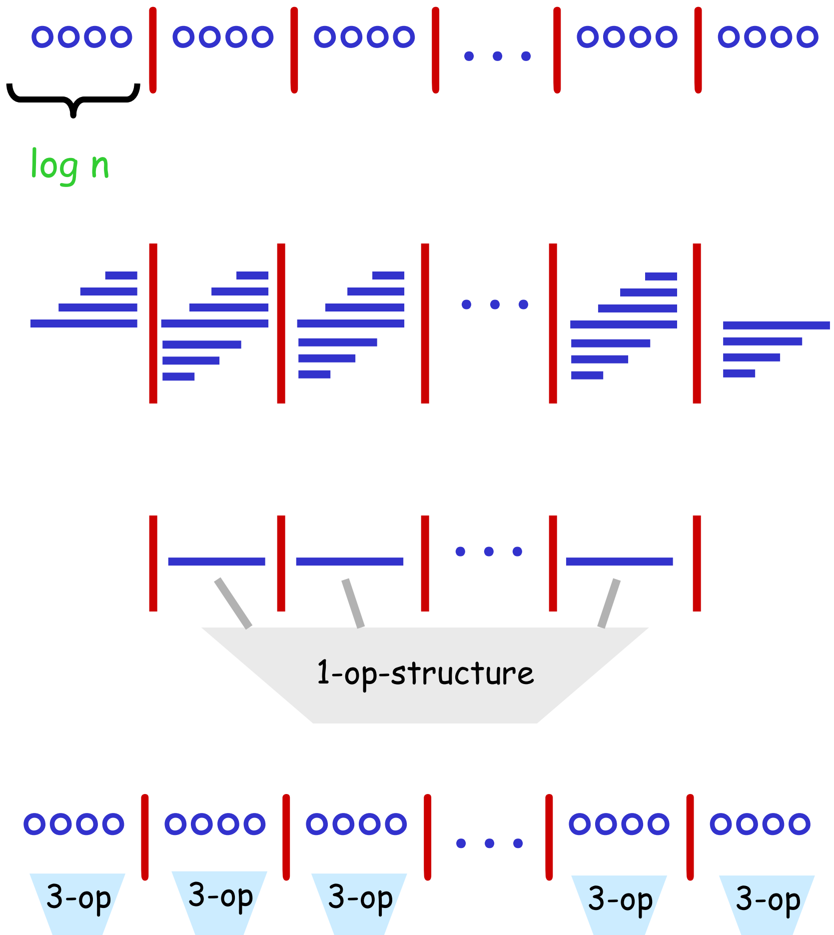


store all prefix- and all suffix-sums within each subsequence

build a 1-op-structure for the $n/\log n$ subsequence-**sums**

recursively build a 3-op-structure for each of the $n/\log n$ subsequences

$$S_3(n) \le 2n + \underbrace{S_1\left(\frac{n}{\log n}\right)}_{\le n} + \frac{n}{\log n} \cdot S_3(\log n) \le 3n + \frac{n}{\log n} \cdot S_3(\log n)$$



store all prefix- and all suffix-sums within each subsequence

build a 1-op-structure for the $n/\log n$ subsequence-**sums**

recursively build a 3-op-structure for each of the $n/\log n$ subsequences

$$S_3(n) \le 2n + \underbrace{S_1\left(\frac{n}{\log n}\right)}_{\le n} + \frac{n}{\log n} \cdot S_3(\log n) \le 3n + \frac{n}{\log n} \cdot S_3(\log n)$$

$$\Rightarrow \; S_3(n) \le 3n \log^* n$$

$S_5(n) = ?$ $S_7(n) = ?$ $S_9(n) = ?$

$S_{2k+1}(n) = ?$

$S_5(n) = ?$ $S_7(n) = ?$ $S_9(n) = ?$

$S_{2k+1}(n) = ?$

Assume: $S_{2k-1}(n) \leq (2k-1) \cdot n \cdot f(n)$

realized by (2k-1)-op-structure

$S_5(n) = ?$ $S_7(n) = ?$ $S_9(n) = ?$

$S_{2k+1}(n) = ?$

Assume: $S_{2k-1}(n) \leq (2k-1) \cdot n \cdot f(n)$

realized by (2k-1)-op-structure

Show: $S_{2k+1}(n) \leq (2k+1) \cdot n \cdot f^*(n)$

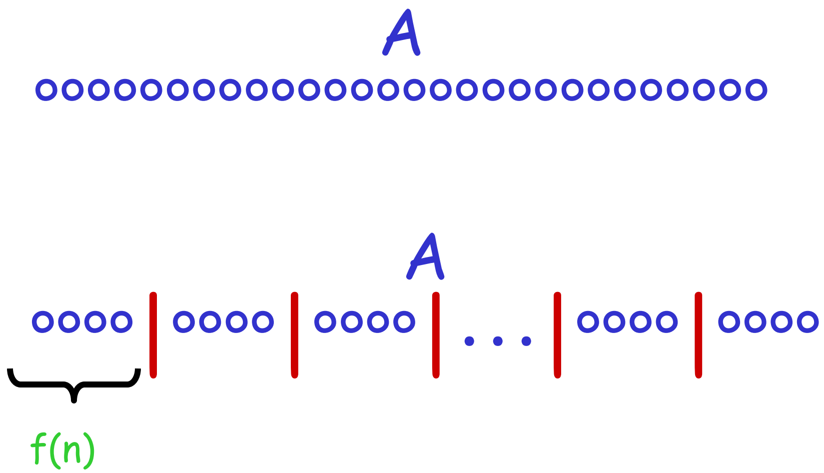
"(2k+1)-op-structure"

case $n \leq 2k+2$: trivial

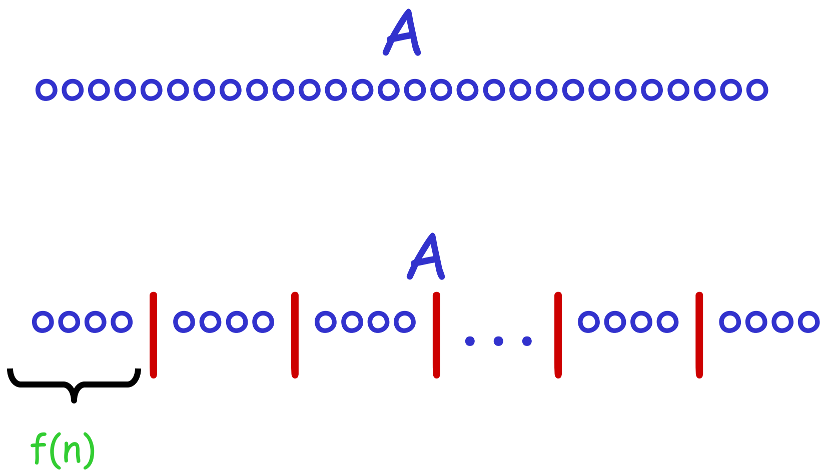
case $n \geq 2k+3$: use recursive construction


A

partition A-sequence into $n/f(n)$ subsequences of length $\leq f(n)$ each

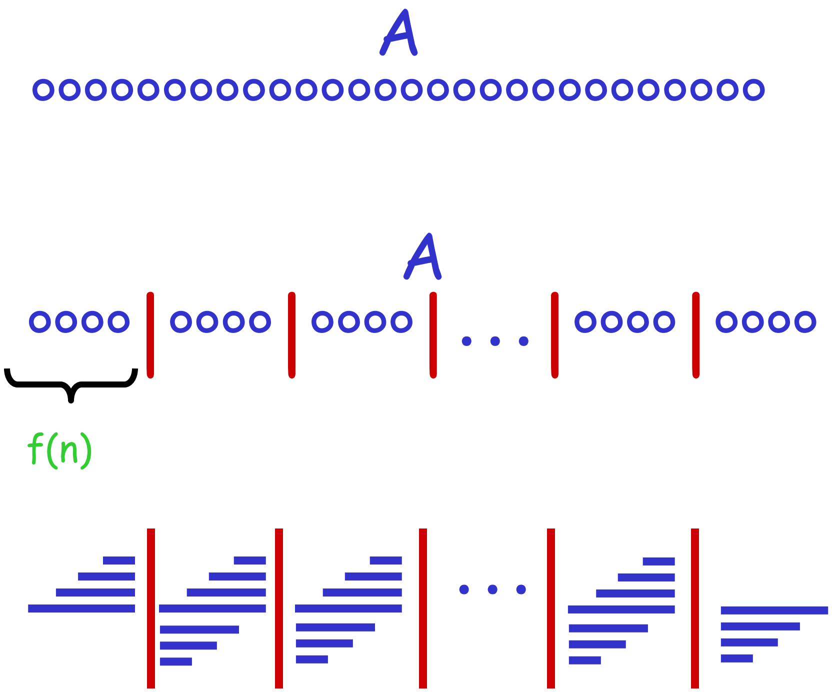


partition A-sequence into
$n/f(n)$ subsequences of length
$\leq f(n)$ each



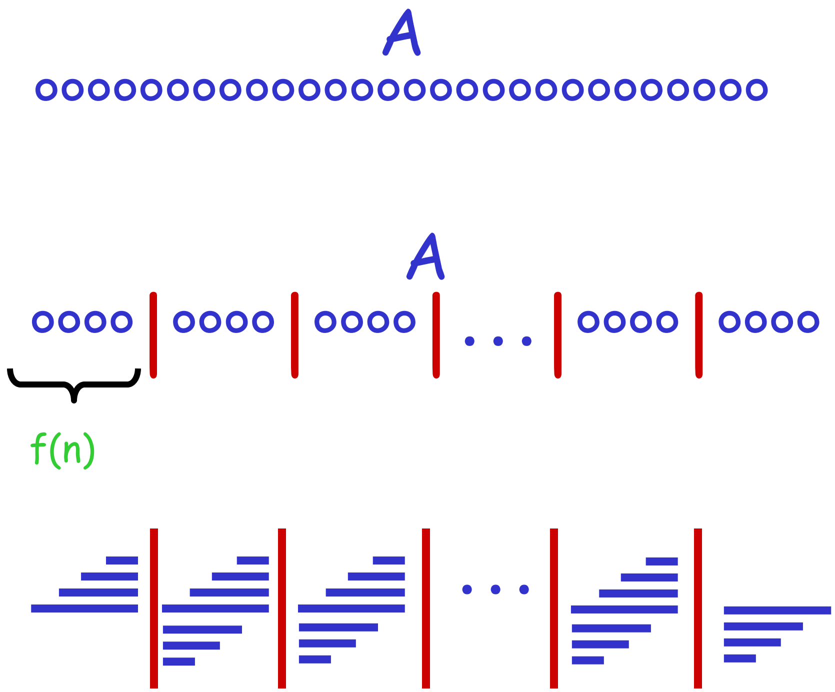
partition A-sequence into
n/f(n) subsequences of length
≤ f(n) each

store all prefix- and all suffix-
sums within each subsequence



partition A-sequence into $n/f(n)$ subsequences of length $\leq f(n)$ each

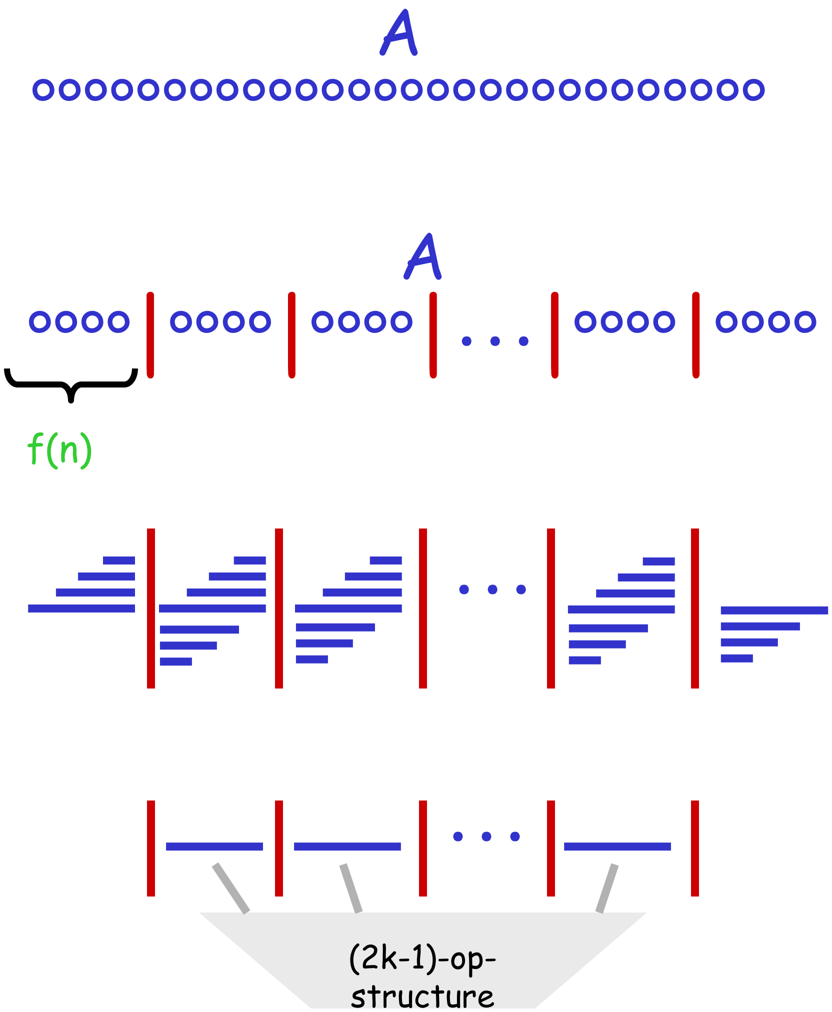
store all prefix- and all suffix-sums within each subsequence



partition A-sequence into n/f(n) subsequences of length $\leq$ f(n) each

store all prefix- and all suffix-sums within each subsequence

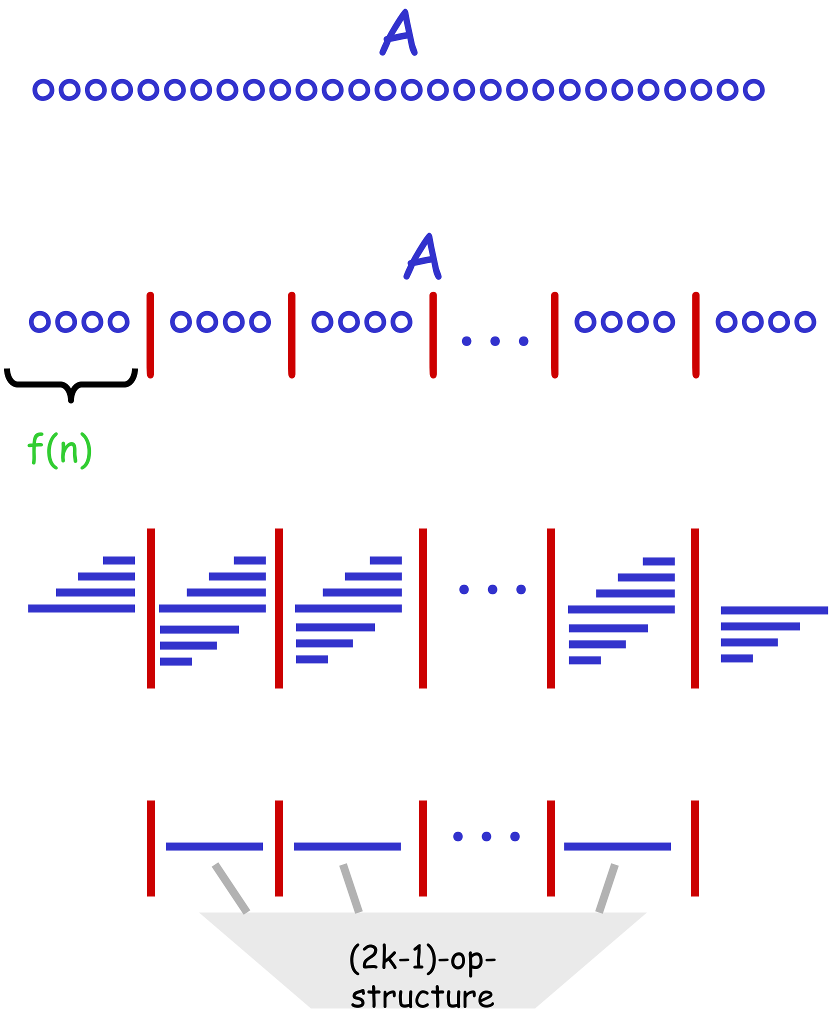
build a (2k-1)-op-structure for the n/f(n) subsequence-**sums**



partition A-sequence into n/f(n) subsequences of length ≤ f(n) each

store all prefix- and all suffix-sums within each subsequence

build a (2k-1)-op-structure for the n/f(n) subsequence-**sums**

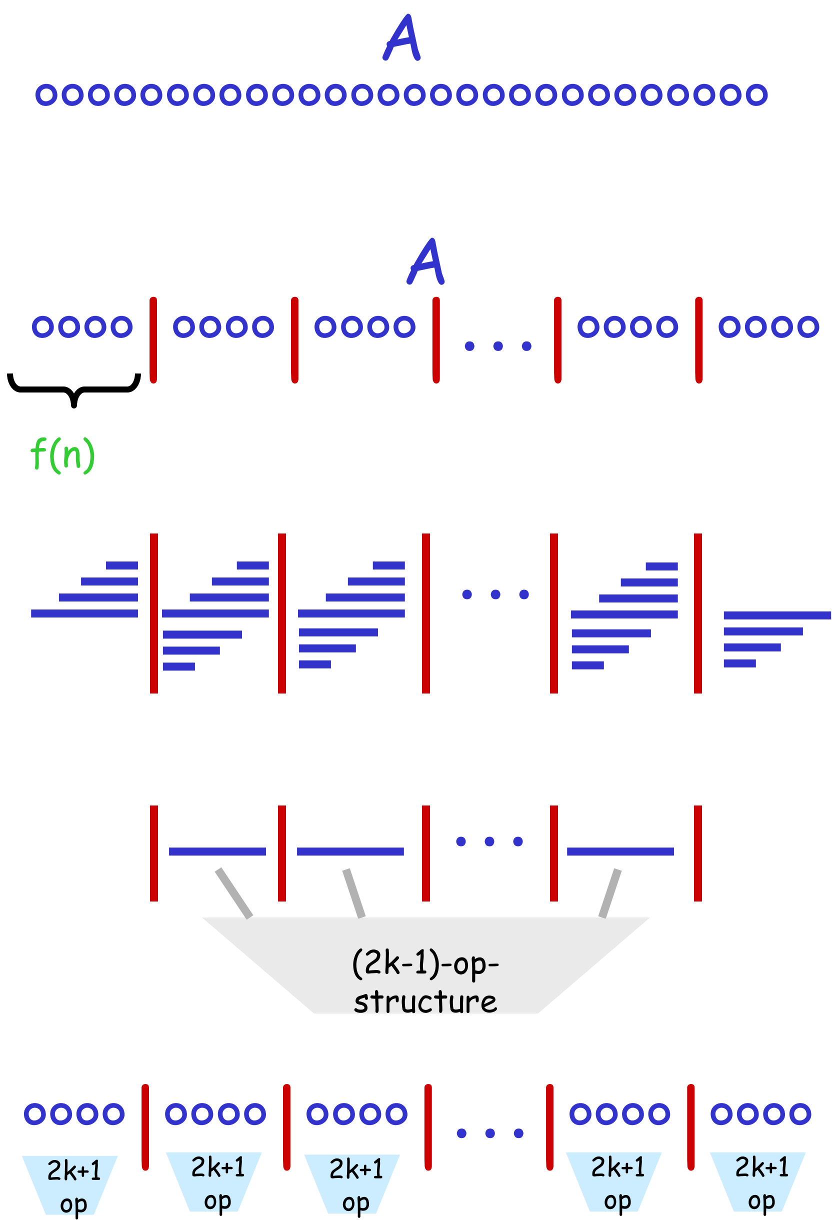


partition A-sequence into $n/f(n)$ subsequences of length $\leq f(n)$ each

store all prefix- and all suffix-sums within each subsequence

build a (2k-1)-op-structure for the $n/f(n)$ subsequence-**sums**

recursively build a (2k+1)-op-structure for each of the $n/f(n)$ subsequences

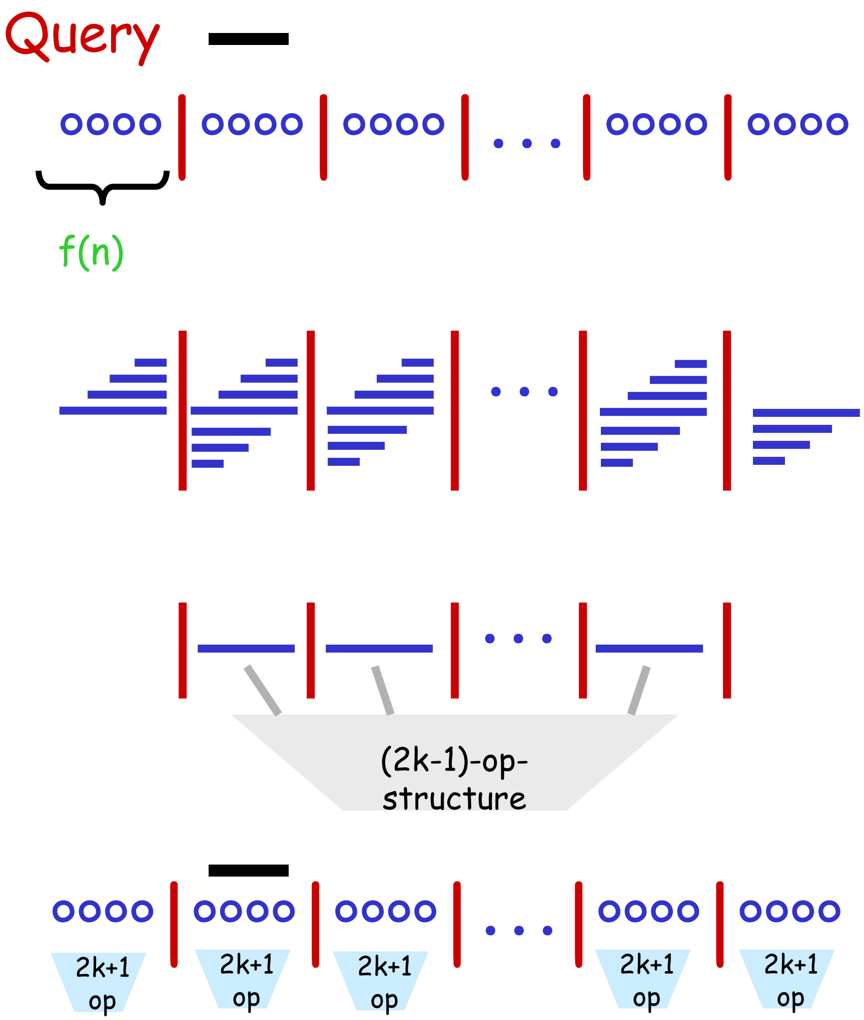


partition A-sequence into $n/f(n)$ subsequences of length $\leq f(n)$ each

store all prefix- and all suffix-sums within each subsequence

build a (2k-1)-op-structure for the $n/f(n)$ subsequence-**sums**

recursively build a (2k+1)-op-structure for each of the $n/f(n)$ subsequences



store all prefix- and all suffix-sums within each subsequence

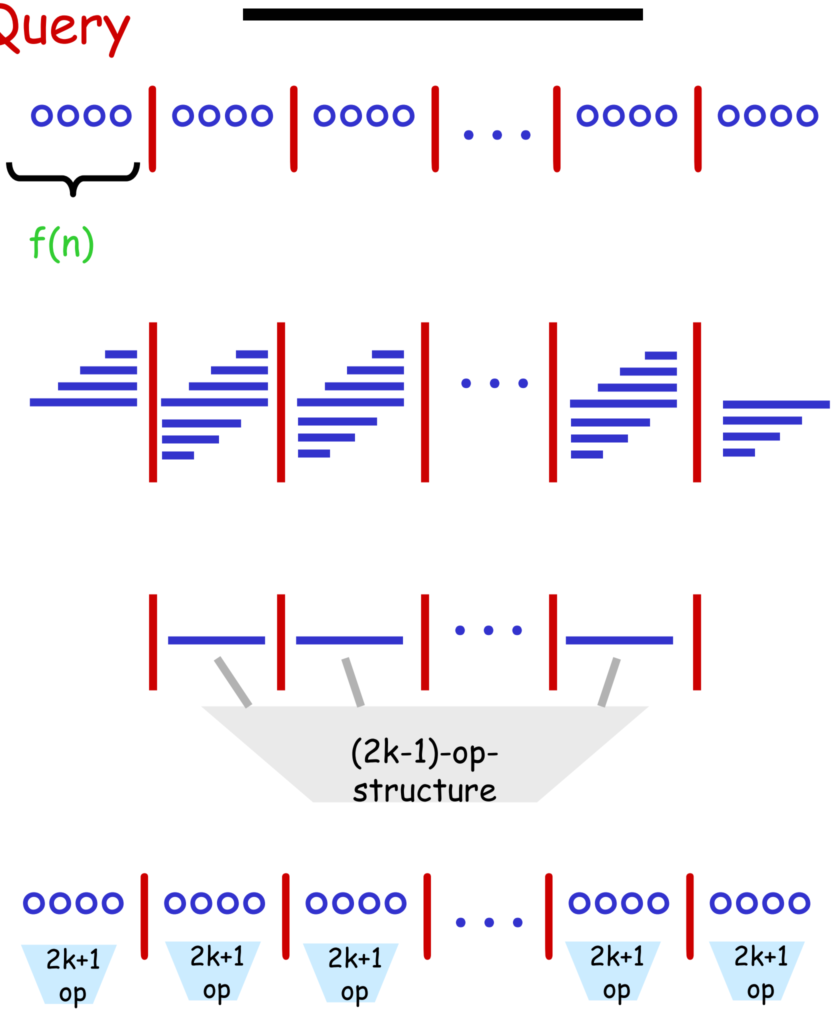
build a (2k-1)-op-structure for the n/f(n) subsequence-**sums**

recursively build a (2k+1)-op-structure for each of the n/f(n) subsequences

Query answering:

either use one of the recursive (2k+1)-op-structures
or return (suffix-sum)+(answer from (2k-1)-op-structure)+(prefix-sum)

Query



store all prefix- and all suffix-sums within each subsequence

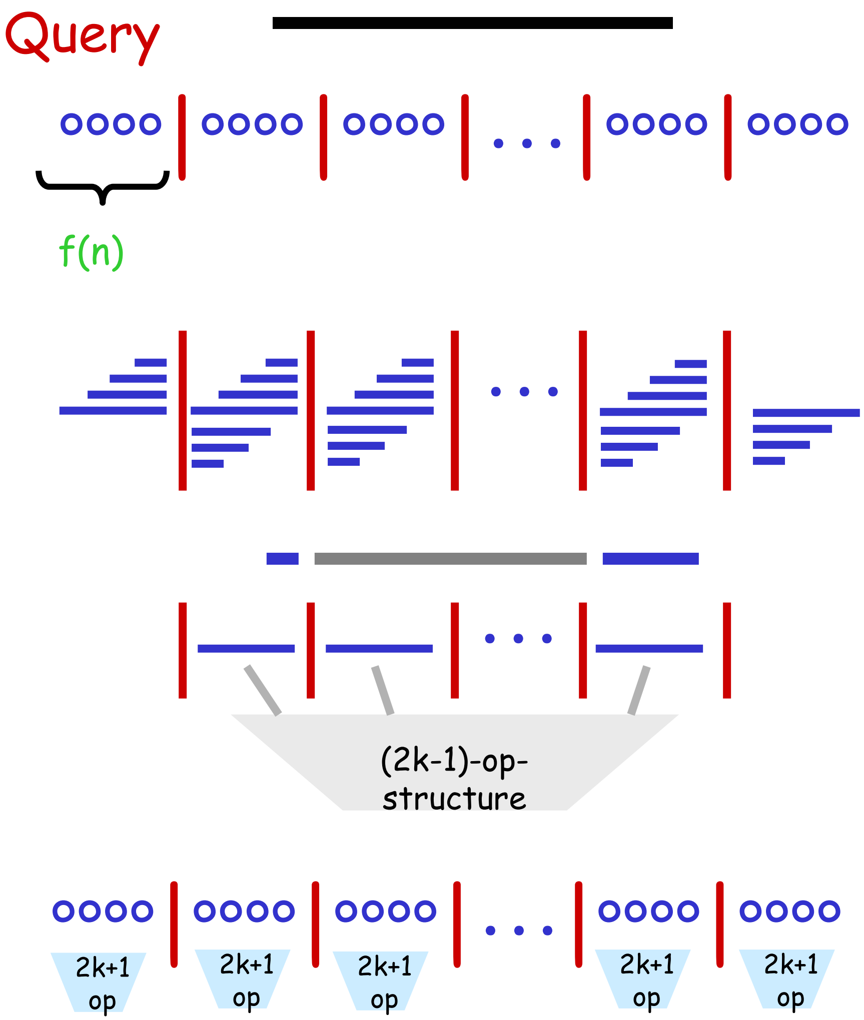
build a (2k-1)-op-structure for the n/f(n) subsequence-**sums**

recursively build a (2k+1)-op-structure for each of the n/f(n) subsequences

Query answering:

either use one of the recursive (2k+1)-op-structures

or return (suffix-sum)+(answer from (2k-1)-op-structure)+(prefix-sum)



store all prefix- and all suffix-sums within each subsequence

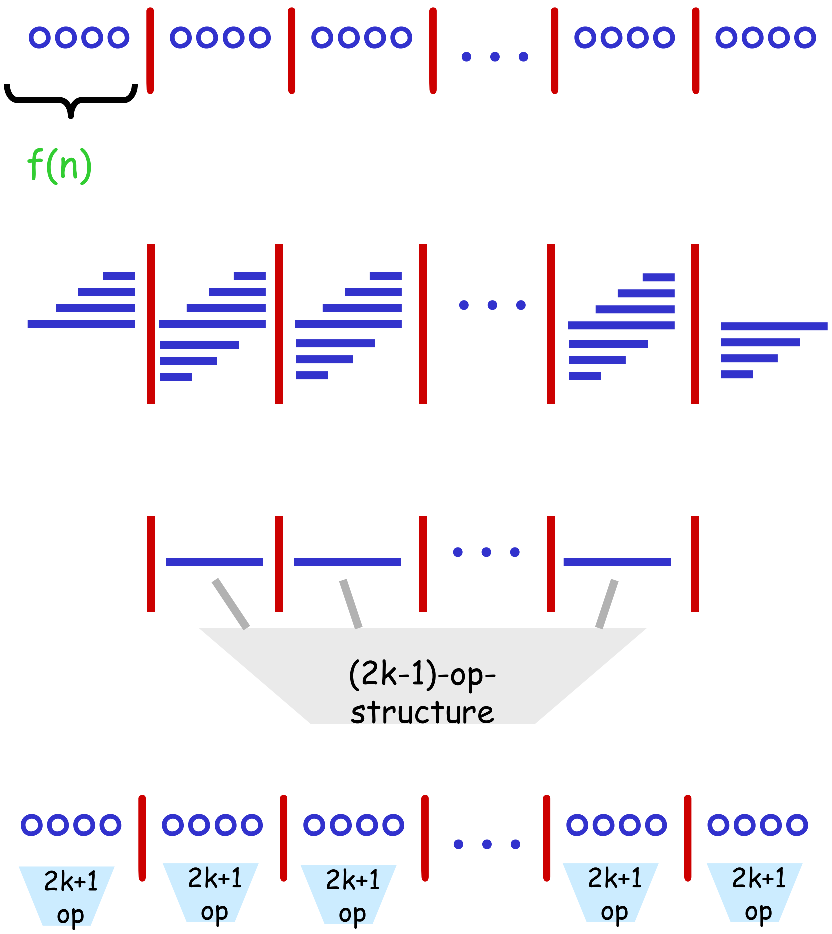
build a (2k-1)-op-structure for the n/f(n) subsequence-**sums**

recursively build a (2k+1)-op-structure for each of the n/f(n) subsequences

Query answering:

either use one of the recursive (2k+1)-op-structures

or return (suffix-sum)+(answer from (2k-1)-op-structure)+(prefix-sum)



store all prefix- and all suffix-sums within each subsequence

build a (2k-1)-op-structure for the $n/f(n)$ subsequence-**sums**

recursively build a (2k+1)-op-structure for each of the $n/f(n)$ subsequences

$$S_{2k+1}(n) \leq 2n + S_{2k-1}\left(\frac{n}{f(n)}\right) + \frac{n}{f(n)} \cdot S_{2k+1}(\,f(n)\,)$$

$$S_{2k+1}(n) \le 2n + \underbrace{S_{2k-1}\left(\frac{n}{f(n)}\right)}_{\le (2k-1)\frac{n}{f(n)} \cdot f\left(\frac{n}{f(n)}\right)} + \frac{n}{f(n)} \cdot S_{2k+1}(\, f(n) \,)$$

$$S_{2k+1}(n) \leq 2n + \underbrace{S_{2k-1}\left(\frac{n}{f(n)}\right)}_{\leq (2k-1)\frac{n}{\cancel{f(n)}} \cdot f\left(\cancel{\frac{n}{f(n)}}\right)} + \frac{n}{f(n)} \cdot S_{2k+1}(f(n))$$

$$S_{2k+1}(n) \leq 2n + S_{2k-1}\left(\frac{n}{f(n)}\right) + \frac{n}{f(n)} \cdot S_{2k+1}(\, f(n)\,)$$

$$\leq (2k-1)\, \frac{n}{f(n)} \cdot f\left(\frac{n}{f(n)}\right)$$

$$S_{2k+1}(n) \leq (2k+1) \cdot n + \frac{n}{f(n)} \cdot S_{2k+1}(\, f(n)\,)$$

$$S_{2k+1}(n) \leq 2n + \underbrace{S_{2k-1}\left(\frac{n}{f(n)}\right)}_{\leq (2k-1)\frac{n}{f(n)} \cdot f\left(\frac{n}{f(n)}\right)} + \frac{n}{f(n)} \cdot S_{2k+1}(\, f(n)\,)$$

$$S_{2k+1}(n) \leq (2k+1) \cdot n + \frac{n}{f(n)} \cdot S_{2k+1}(\, f(n)\,)$$

$$\Rightarrow \; S_{2k+1}(n) \leq (2k+1) n\, f^{*}(n)$$

$k=1$: $S_1(n) \le n \log n$

For all $k>1$: $S_{2k-1}(n) \le (2k-1)\cdot n\cdot f(n)$

$$\Rightarrow \quad S_{2k+1}(n) \le (2k+1)\cdot n\cdot f^*(n)$$

$k=1$: $S_1(n) \leq n \log n$

For all $k>1$: $S_{2k-1}(n) \leq (2k-1)\cdot n\cdot f(n)$

$\Rightarrow S_{2k+1}(n) \leq (2k+1)\cdot n\cdot f^*(n)$

For all $k\geq 1$: $S_{2k+1} \leq (2k+1)\cdot n\cdot \log^{\overbrace{**\ldots*}^{k \text{ times}}}(n)$

For all $k \geq 1$: $S_{2k+1} \leq (2k+1) \cdot n \cdot \log^{\overbrace{**\cdots*}^{k \text{ times}}}(n)$

For all $k \geq 1$: $S_{2k+1} \leq (2k+1) \cdot n \cdot \log^{\overbrace{**\cdots*}^{k \text{ times}}}(n)$

Define $\alpha(n) = \min\{\, k \mid \log^{\overbrace{**\cdots*}^{k \text{ times}}}(n) \leq 2 \,\}$

For all $k \geq 1$: $S_{2k+1} \leq (2k+1) \cdot n \cdot \log^{\overbrace{**\cdots*}^{k \text{ times}}}(n)$

Define $\alpha(n) = \min\{ k \mid \log^{\overbrace{**\cdots*}^{k \text{ times}}}(n) \leq 2 \}$

For $k = \alpha(n)$: $S_{2\alpha(n)+1} \leq (2\alpha(n)+1) \cdot n \cdot 2$

$= O(\alpha(n) \cdot n)$

For all $k \geq 1$: $S_{2k+1} \leq (2k+1) \cdot n \cdot \log^{\overbrace{**\cdots*}^{k \text{ times}}}(n)$

Define $\alpha(n) = \min\{\, k \mid \log^{\overbrace{**\cdots*}^{k \text{ times}}}(n) \leq 2 \,\}$

For $k=\alpha(n)$: $S_{2\alpha(n)+1} \leq (2\alpha(n)+1) \cdot n \cdot 2$

$= O(\alpha(n) \cdot n)$

For $O(\alpha(n))$ query cost, space $O(\alpha(n) \cdot n)$ suffices.

Exercise:

For $O(\alpha(n))$ query cost, space $O(n)$ suffices.

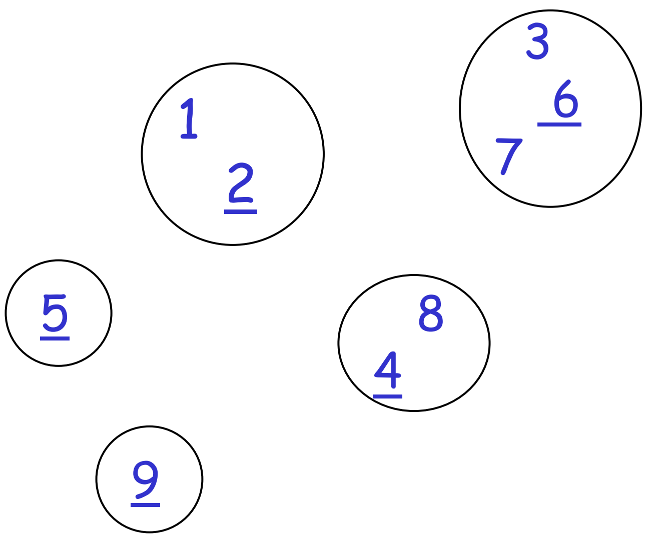
Yao; Chazelle, Rosenberg

Union Find with Path Compressions

Union Find with Path Compressions

Maintain partition of $S = \{1, 2, \cdots, n\}$

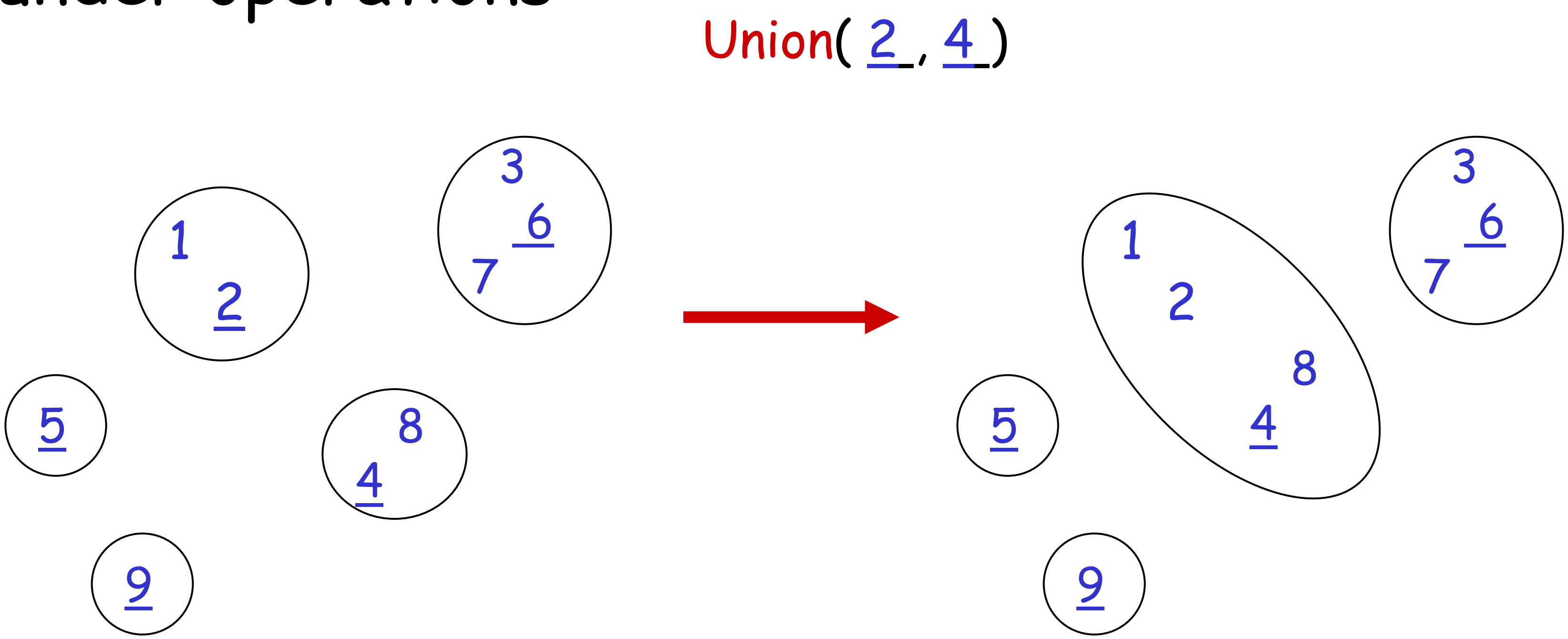
under operations



Union Find with Path Compressions

Maintain partition of $S = \{ 1,2,\cdots,n\}$

under operations

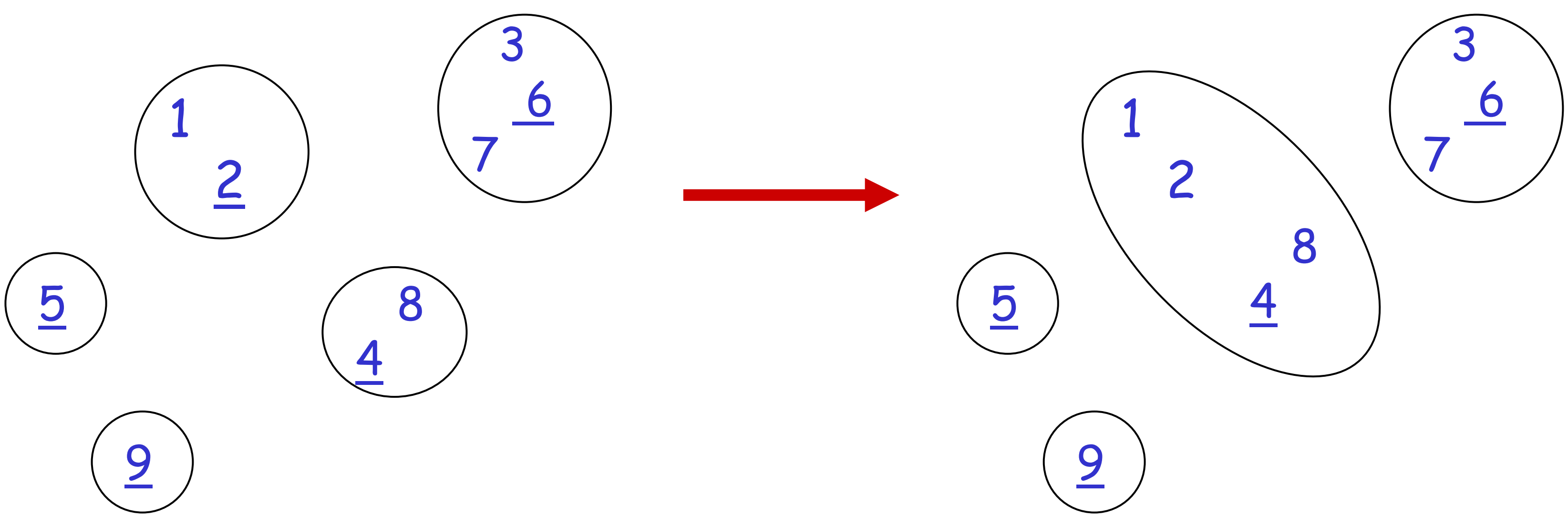


Union Find with Path Compressions

Maintain partition of $S = \{1, 2, \cdots, n\}$

under operations

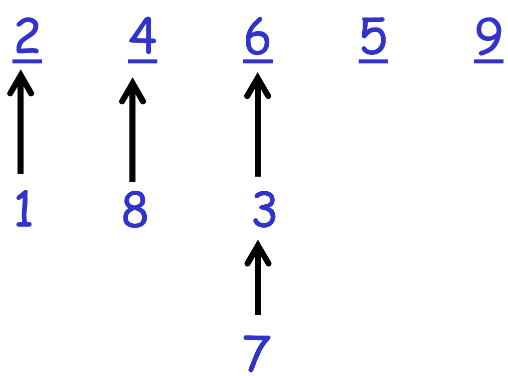
Union(2 , 4)



Find(3) = 6 (representative element)

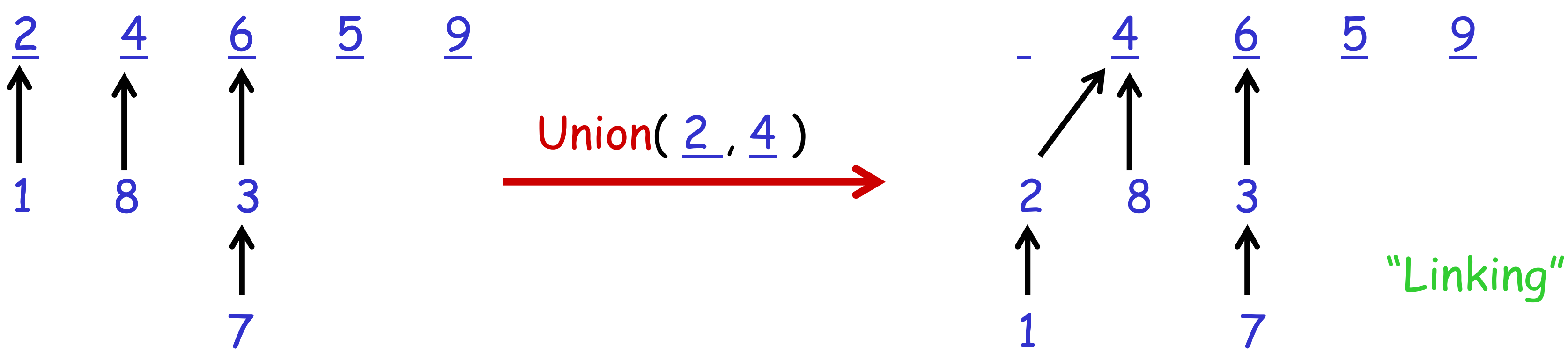
Impementation

* forest $\mathcal{F}$ of rooted trees with node set S
* one tree for each group in current partition
* root of tree is representative of the group



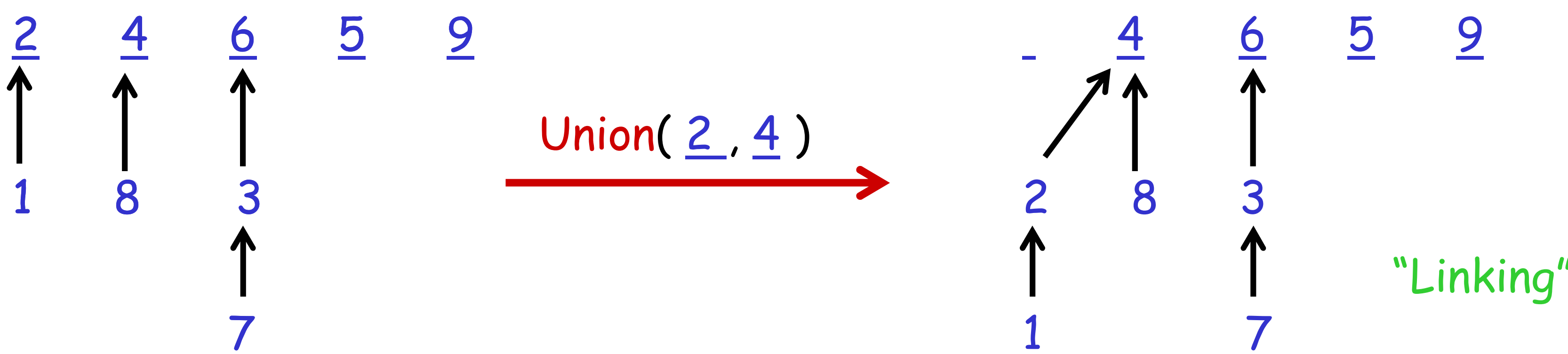
Impementation

* forest $\mathcal{F}$ of rooted trees with node set S
* one tree for each group in current partition
* root of tree is representative of the group



Impementation

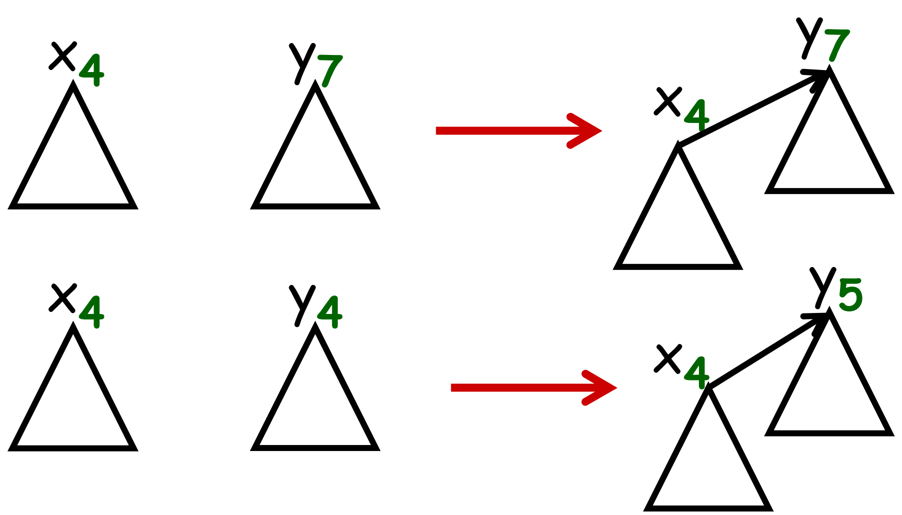
* forest $\mathcal{F}$ of rooted trees with node set S
* one tree for each group in current partition
* root of tree is representative of the group



Find(x) follow path from x to root "path follwoing"

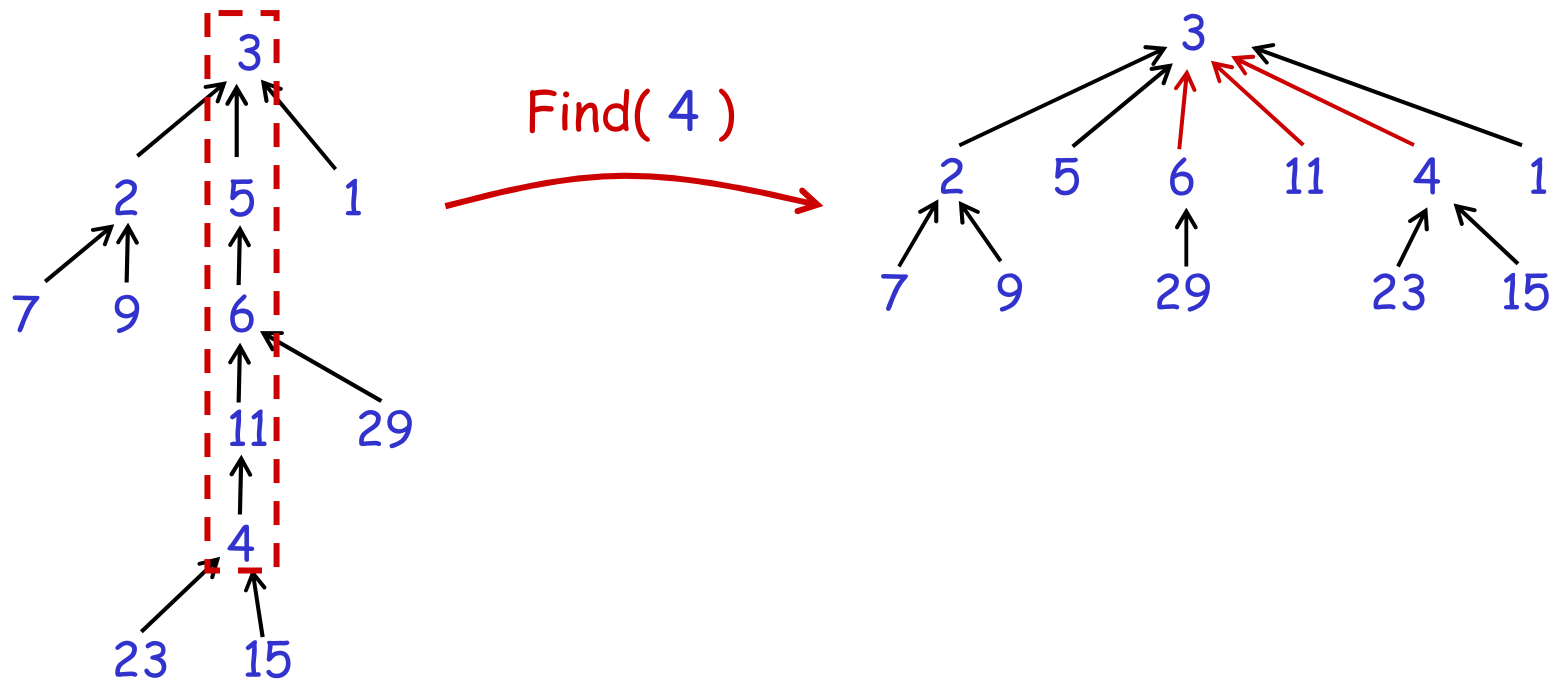
Heuristic 1: "linking by rank"

- each node x carries integer rk(x)
- initially rk(x) = 0
- as soon as x is NOT a root, rk(x) stays unchanged
- for Union(x , y) make node with smaller rank child of the other
  in case of tie, increment one of the ranks



Heuristic 2: Path compression

when performin a Find(x) operation make all nodes in the "findpath" children of the root



sequence of Union and Find operation

Explicit cost model:

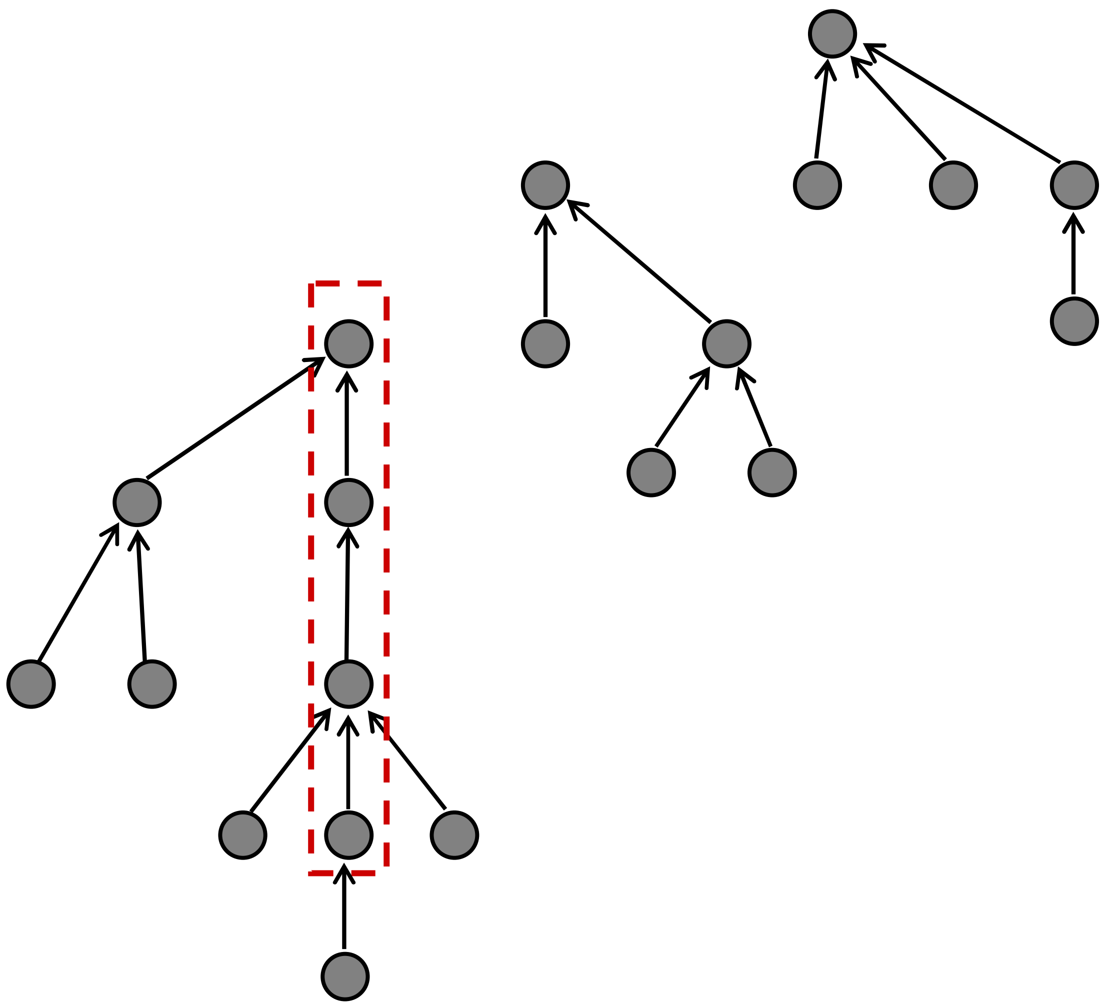
cost(op) = # times some node gets a new parent

Time for Union(x , y) = O(1) = O(cost(Union(x,y)))

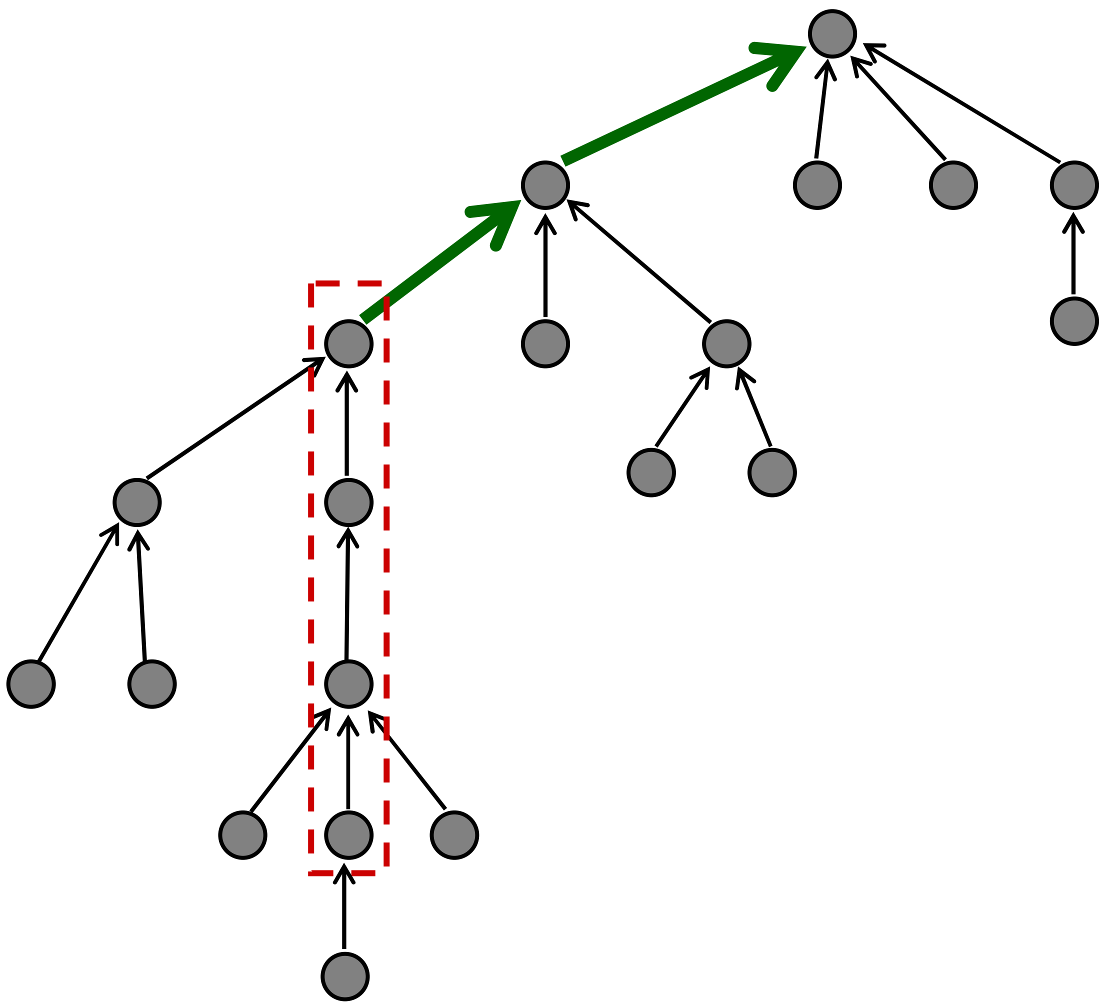
Time for Find(x) = O(# of nodes on findpath)

= O(2 + cost(Find(x)))

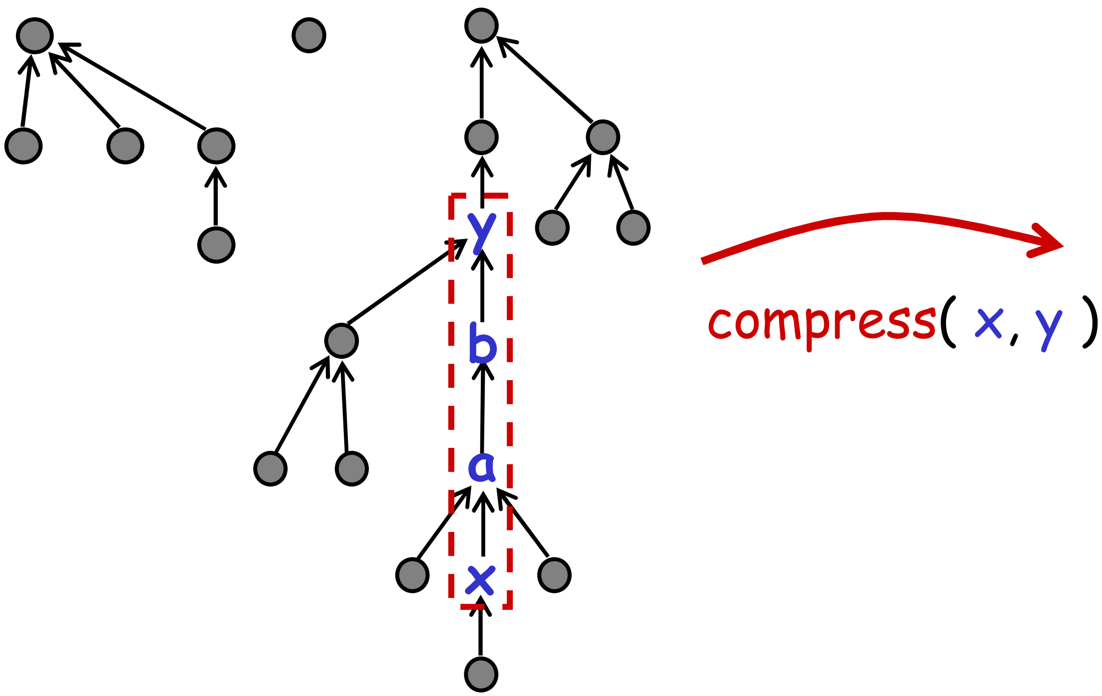
For analysis assume all Unions are performed first, but Find-paths are only followed (and compressed) to correct node.



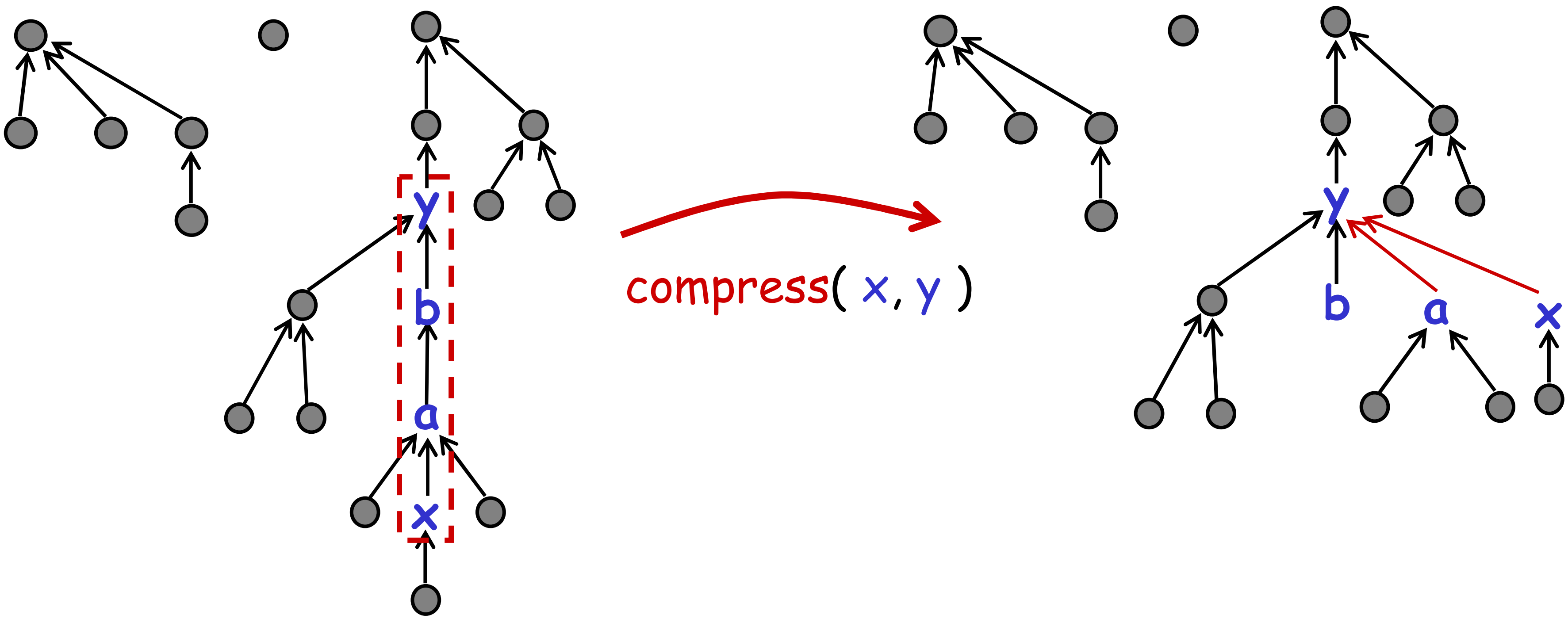
For analysis assume all Unions are performed first, but Find-paths are only followed (and compressed) to correct node.



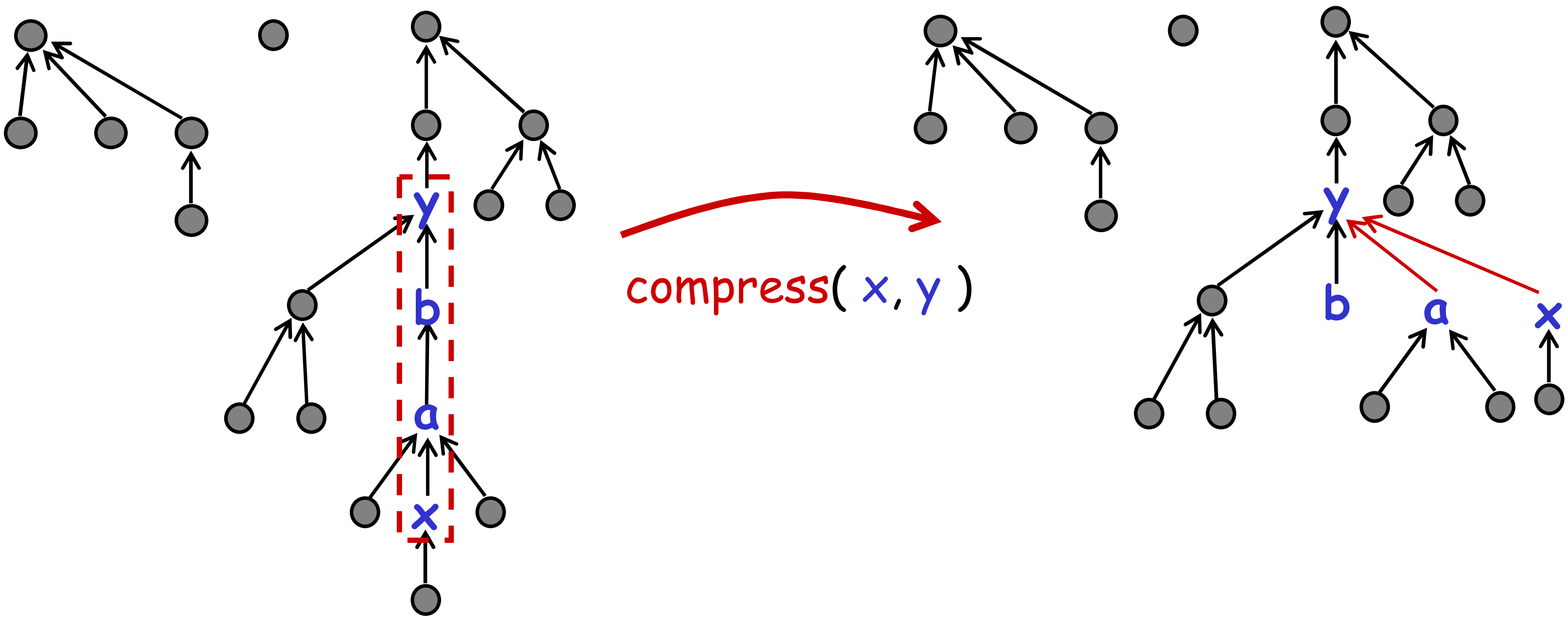
General path compression in forest $\mathcal{F}$



General path compression in forest $\mathcal{F}$



General path compression in forest $\mathcal{F}$



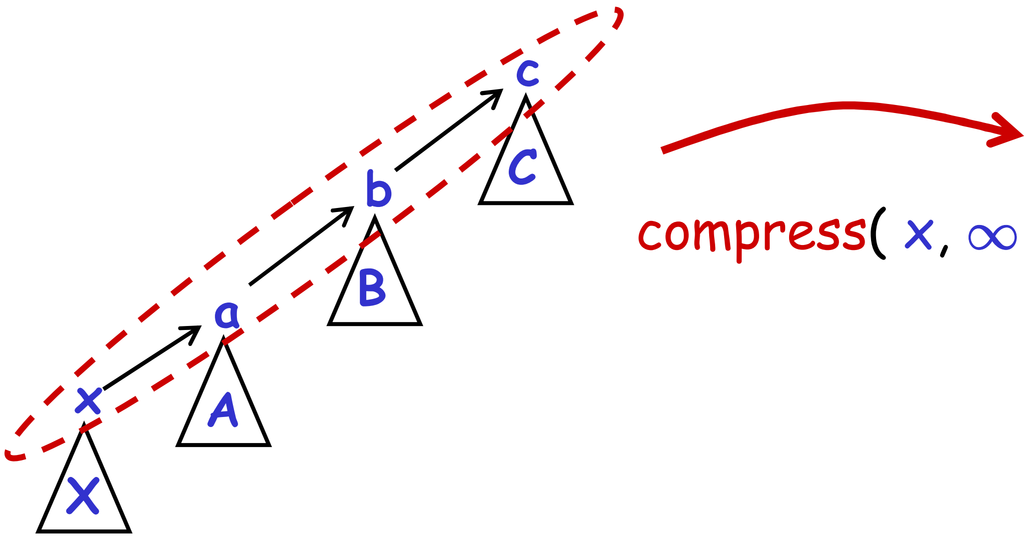
cost(compress(x,y)) = # of nodes that get a new parent

General path compression in forest $\mathcal{F}$

“rootpath compress“

General path compression in forest $\mathcal{F}$

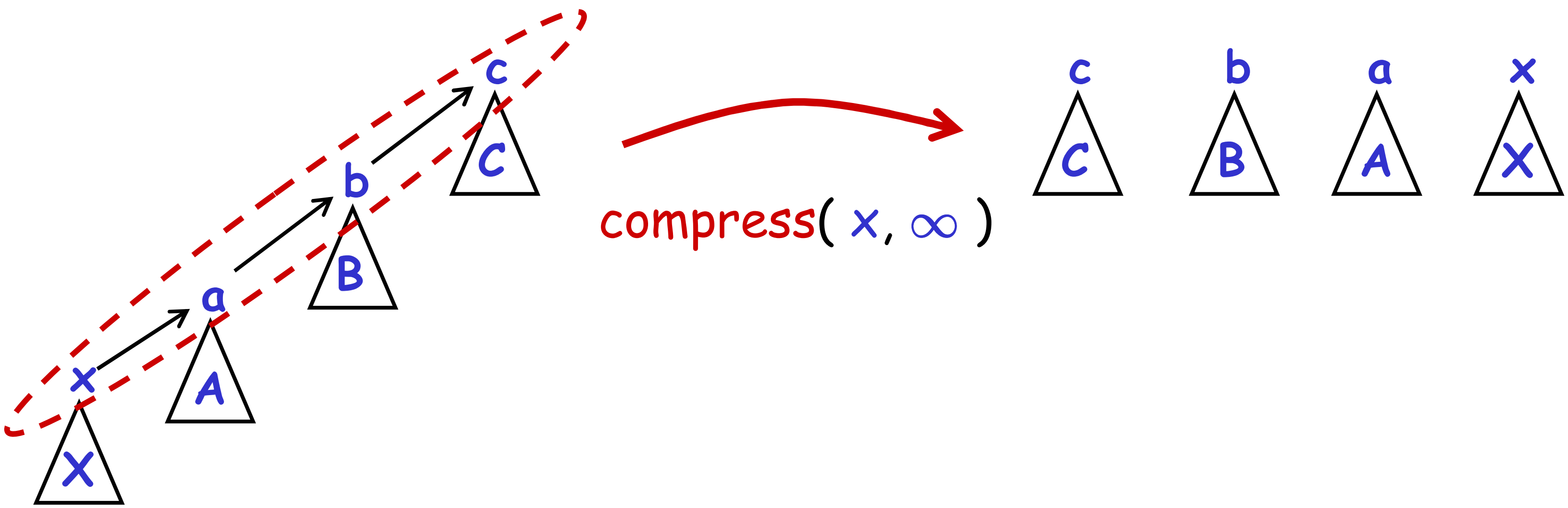
"rootpath compress"



compress(x, ∞)

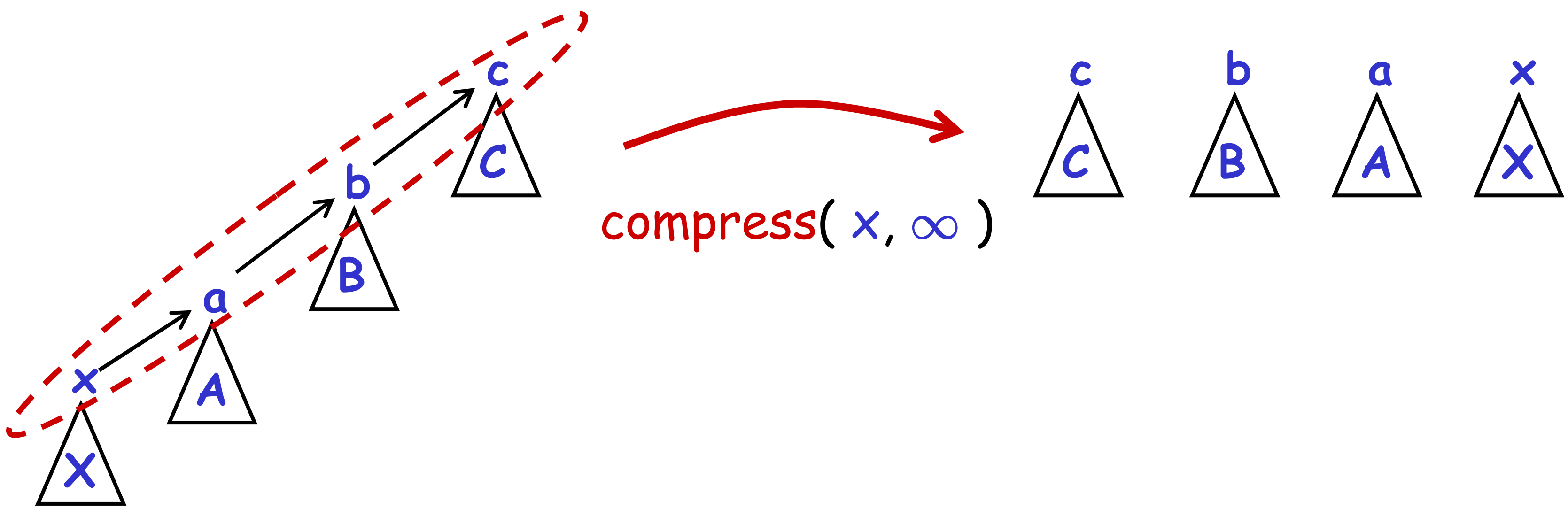
General path compression in forest $\mathcal{F}$

"rootpath compress"



General path compression in forest $\mathcal{F}$

"rootpath compress"



cost(compress(x, ∞)) = # of nodes that get a new parent

= 0

Problem formulation

$\mathcal{F}$ forest on node set X

C sequence of compress operations on $\mathcal{F}$

$|C|$ = # of true compress operations in C

(rootpath compresses excluded)

cost(C) = $\sum$(cost of individual operations)

Problem formulation

$\mathcal{F}$ forest on node set X

C sequence of compress operations on $\mathcal{F}$

$|C|$ = # of true compress operations in C

(rootpath compresses excluded)

cost(C) = $\sum$(cost of individual operations)

How large can cost(C) be at most,
in terms of |X| and |C| ?

Dissection of a forest $\mathcal{F}$ with node set X :

partition of X into "top part" X_t
and "bottom part" X_b

so that top part X_t is "upwards closed",

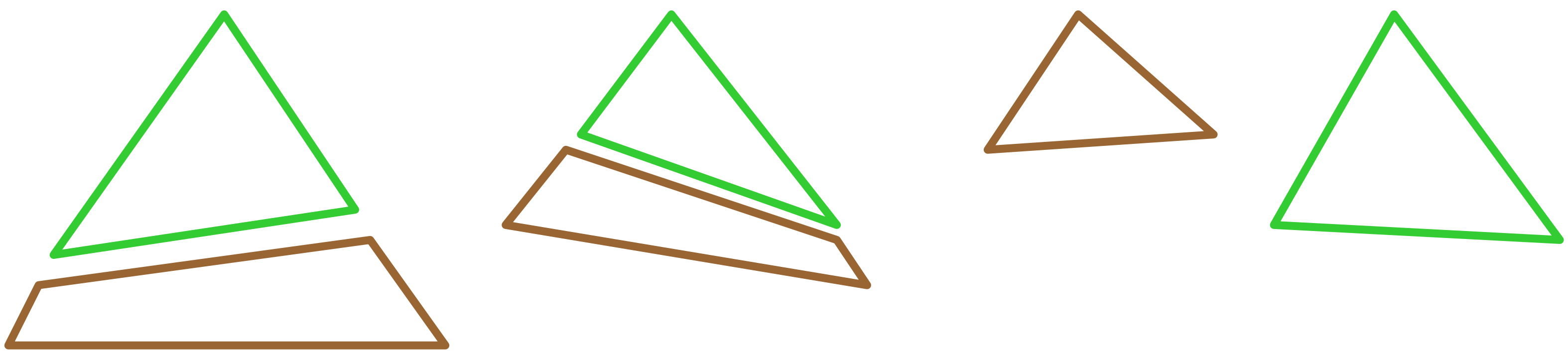
i.e. $x \in X_t \Rightarrow$ every ancestor of x is in X_t also

Dissection of a forest $\mathcal{F}$ with node set X :

partition of X into "top part" X_t
and "bottom part" X_b

so that top part X_t is "upwards closed",

i.e. $x \in X_t \Rightarrow$ every ancestor of x is in X_t also

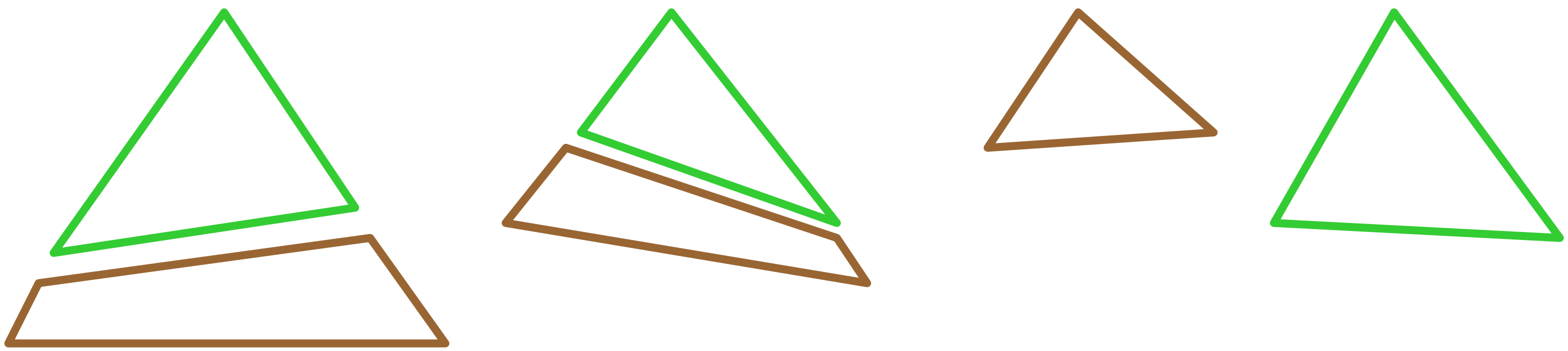


Dissection of a forest $\mathcal{F}$ with node set X :

partition of X into "top part" X_t
and "bottom part" X_b

so that top part X_t is "upwards closed",

i.e. $x \in X_t \Rightarrow$ every ancestor of x is in X_t also



Note: X_t, X_b dissection for $\mathcal{F}$
$\mathcal{F}'$ obtained from $\mathcal{F}$ by
sequence of path compressions
$\Rightarrow$ X_t, X_b is dissection for $\mathcal{F}'$

Main Lemma:

$\mathcal{C}$ ... sequence of operations on $\mathcal{F}$ with node set X

X_t, X_b dissection for $\mathcal{F}$ inducing subforests $\mathcal{F}_t$, $\mathcal{F}_b$

Main Lemma:

C ... sequence of operations on $\mathcal{F}$ with node set X

X_t, X_b dissection for $\mathcal{F}$ inducing subforests $\mathcal{F}_t$, $\mathcal{F}_b$

$\Rightarrow$ $\exists$ compression sequences
C_b for $\mathcal{F}_b$ and C_t for $\mathcal{F}_t$
with and

$$|C_b| + |C_t| \leq |C|$$

and

$$\text{cost}(C) \leq \text{cost}(C_b) + \text{cost}(C_t) + |X_b| + |C_t|$$

Proof: 1) How to get C_b and C_t from C:

Proof: 1) How to get C_b and C_t from C:

compression paths from C

case 1: 

into C_t

Proof: 1) How to get C_b and C_t from C:

compression paths from C

case 1:

into C_t

case 2:

into C_b

Proof: 1) How to get C_b and C_t from C:

compression paths from C

case 1: 

into C_t

case 2: 

into C_b

case 3: 

into C_t

into C_b

Proof:

$$|C_b| + |C_t| \leq |C|$$

compression paths from C

case 1: 

into C_t

case 2: 

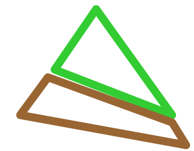
into C_b

case 3: 

into C_t

into C_b

$$\text{cost}(C) \leq \text{cost}(C_b) + \text{cost}(C_t) + |X_b| + |C_t|$$



cost(C)	
green node gets new green parent:	accounted by $\text{cost}(C_t)$
brown node gets new brown parent:	accounted by $\text{cost}(C_b)$
brown node gets new green parent: for the first time	accounted by $|X_b|$
brown node gets new green parent: again	accounted by $|C_t|$

$f(m,n)$ ... maximum cost of any compression sequence C with $|C|=m$ in an arbitrary forest with n nodes.

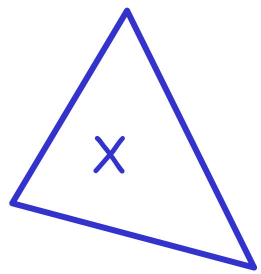
Claim: $f(m,n) \leq (m+n) \cdot \log_2 n$

Claim: $f(m,n) \leq (m+n) \cdot \log_2 n$

Claim: $f(m,n) \leq (m+n)\cdot\log_2 n$

Proof:

forest $\mathcal{F}$



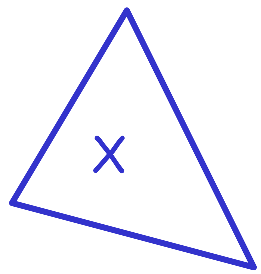
$|X|=n$

C compression sequence $|C|=m$

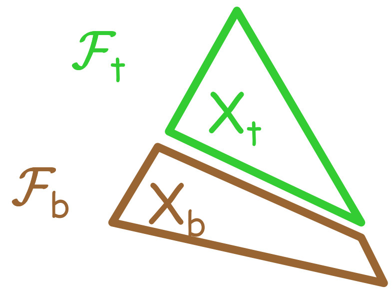
Claim: $f(m,n) \leq (m+n)\cdot \log_2 n$

Proof:

forest $\mathcal{F}$



$|X|=n$



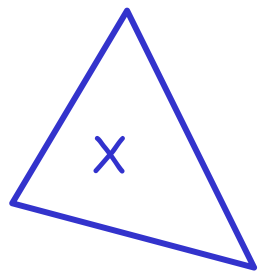
$|X_t|=|X_b|=n/2$

C compression sequence $|C|=m$

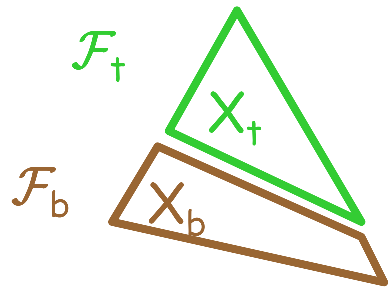
Claim: $f(m,n) \leq (m+n) \cdot \log_2 n$

Proof:

forest $\mathcal{F}$



$|X| = n$



$|X_t| = |X_b| = n/2$

C compression sequence $\quad |C| = m$

Main Lemma $\Rightarrow \exists\, C_t, C_b \quad |C_b| + |C_t| \leq |C|$

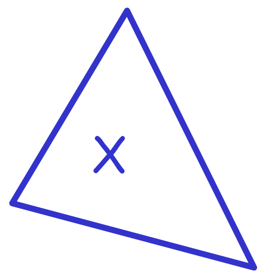
$$m_b + m_t \leq m$$

$$\text{cost}(C) \leq \text{cost}(C_b) + \text{cost}(C_t) + |X_b| + |C_t|$$

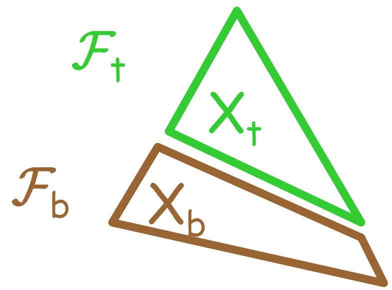
Claim: $f(m,n) \leq (m+n)\cdot\log_2 n$

Proof:

forest $\mathcal{F}$



$|X|=n$



$|X_t|=|X_b|=n/2$

C compression sequence $|C|=m$

Main Lemma $\Rightarrow$ $\exists\ C_t, C_b$ $|C_b|+|C_t| \leq |C|$

$m_b + m_t \leq m$

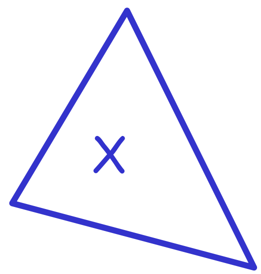
$$\text{cost}(C) \leq \text{cost}(C_b) + \text{cost}(C_t) + |X_b| + |C_t|$$

Induction: $\leq (m_b+n/2)\log n/2 + (m_t+n/2)\log n/2 + n/2 + m_t$

Claim: $f(m,n) \le (m+n)\cdot\log_2 n$

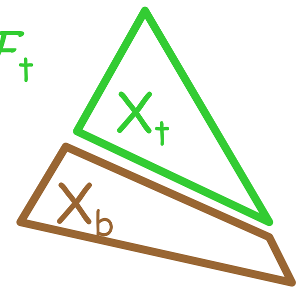
Proof:

forest $\mathcal{F}$



$|X|=n$

$\mathcal{F}_t$ $\mathcal{F}_b$



$|X_t|=|X_b|=n/2$

$\mathcal{C}$ compression sequence $|\mathcal{C}|=m$

Main Lemma $\Rightarrow$ $\exists\ \mathcal{C}_t, \mathcal{C}_b$ $|\mathcal{C}_b|+|\mathcal{C}_t| \le |\mathcal{C}|$

$m_b + m_t \le m$

$$\mathrm{cost}(\mathcal{C}) \le \mathrm{cost}(\mathcal{C}_b) + \mathrm{cost}(\mathcal{C}_t) + |X_b| + |\mathcal{C}_t|$$

Induction: $\le (m_b+n/2)\log n/2 + (m_t+n/2)\log n/2 + n/2 + m_t$

$$\le (m+n)\cdot\log_2 n$$

Corollary:

Any sequence of m Union, Find operations in a universe of n elements that uses arbitrary linking and path compression takes time at most

$$O(\ (m+n)\cdot \log n)$$

Corollary:

Any sequence of m Union, Find operations in a universe of n elements that uses arbitrary linking and path compression takes time at most

$$O(\,(m+n)\cdot\log n)$$

By choosing a dissection that is "unbalanced" in relation to m/n one can prove a better bound of

$$O(\,(m+n)\cdot\log_{\lceil m/n\rceil+1} n)$$

Path compression and union by rank

Path compression and union by rank

Def: $\mathcal{F}$ forest, x node in $\mathcal{F}$

$r(x)$ = height of subtree rooted at x

($r(\text{leaf}) = 0$)

$\mathcal{F}$ is a **rank forest**, if

for every node x

for every i with $0 \leq i < r(x)$,

there is a child y_i of x with $r(y_i)=i$.

Path compression and union by rank

Def: $\mathcal{F}$ forest, x node in $\mathcal{F}$

$r(x)$ = height of subtree rooted at x

($r(\text{leaf}) = 0$)

$\mathcal{F}$ is a **rank forest**, if

for every node x

for every i with $0 \leq i < r(x)$,

there is a child y_i of x with $r(y_i) = i$.

Note: Union by rank produces rank forests !

Path compression and union by rank

Def: $\mathcal{F}$ forest, x node in $\mathcal{F}$

$r(x)$ = height of subtree rooted at x

($r(\text{leaf}) = 0$)

$\mathcal{F}$ is a **rank forest**, if

for every node x

for every i with $0 \leq i < r(x)$,

there is a child y_i of x with $r(y_i) = i$.

Note: Union by rank produces rank forests !

Lemma: $r(x)=r \Rightarrow x$ is root of subtree with at least 2^r nodes.

Inheritance Lemma:

$\mathcal{F}$ rank forest with maximum rank r and node set X

$s \in \mathbb{N}$: $X_{>s} = \{ x \in X \mid r(x) > s \}$ $\mathcal{F}_{>s}$

$X_{\leq s} = \{ x \in X \mid r(x) \leq s \}$ $\mathcal{F}_{\leq s}$

induced forests

Inheritance Lemma:

$\mathcal{F}$ rank forest with maximum rank r and node set X

$s \in \mathbb{N}$: $X_{>s} = \{ x \in X \mid r(x) > s \}$ $\mathcal{F}_{>s}$ induced forests

$X_{\leq s} = \{ x \in X \mid r(x) \leq s \}$ $\mathcal{F}_{\leq s}$

i) $X_{\leq s}$, $X_{>s}$ is a dissection for $\mathcal{F}$

ii) $\mathcal{F}_{\leq s}$ is a rank forest with maximum rank $\leq s$

iii) $\mathcal{F}_{>s}$ is a rank forest with maximum rank $\leq r-s-1$

iv) $|X_{>s}| \leq |X| / 2^{s+1}$

Inheritance Lemma:

$\mathcal{F}$ rank forest with maximum rank r and node set X

$s \in \mathbb{N}$: $X_{>s} = \{ x \in X \mid r(x) > s \}$ $\mathcal{F}_{>s}$ induced forests

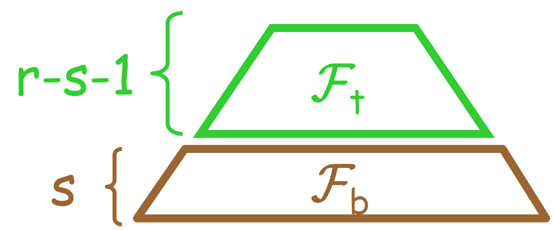
$X_{\leq s} = \{ x \in X \mid r(x) \leq s \}$ $\mathcal{F}_{\leq s}$

i) $X_{\leq s}$, $X_{>s}$ is a dissection for $\mathcal{F}$

ii) $\mathcal{F}_{\leq s}$ is a rank fores with maximum rank $\leq s$

iii) $\mathcal{F}_{>s}$ is a rank fores with maximum rank $\leq r-s-1$

iv) $|X_{>s}| \leq |X| / 2^{s+1}$



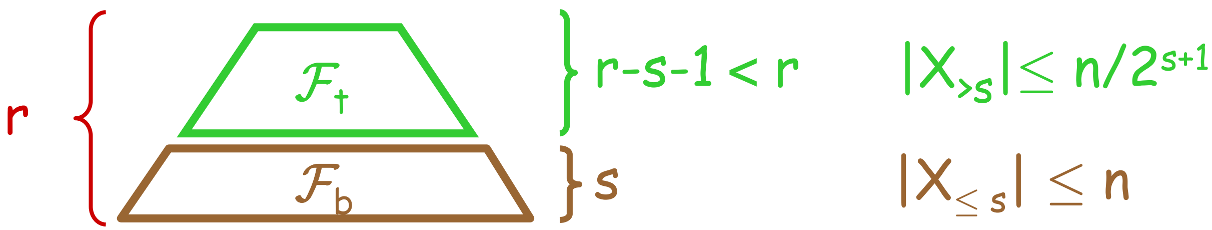
$f(m,n,r)$ = maximum cost of any compression sequence C, with $|C|=m$, in rank forest $\mathcal{F}$ with n nodes and maximum rank r.

$f(m,n,r)$ = maximum cost of any compression sequence C, with $|C|=m$, in rank forest $\mathcal{F}$ with n nodes and maximum rank r.

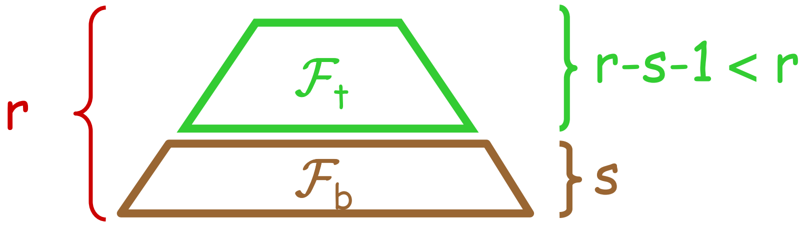
Trivial bounds:

$$f(m,n,r) \leq (r-1)\cdot n$$

$$f(m,n,r) \leq (r-1)\cdot m$$



$f(M,N,R) \leq N \cdot R$

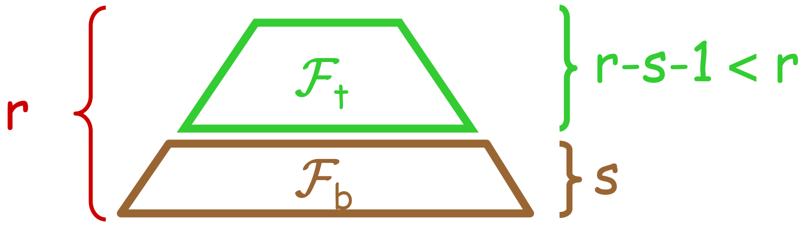


$|X_{>s}| \leq n/2^{s+1}$

$|X_{\leq s}| \leq n$

$$f(M,N,R) \leq N \cdot R$$

$$\text{cost}(C) \leq \text{cost}(C_t) + \text{cost}(C_b) + |X_b| + |C_t|$$

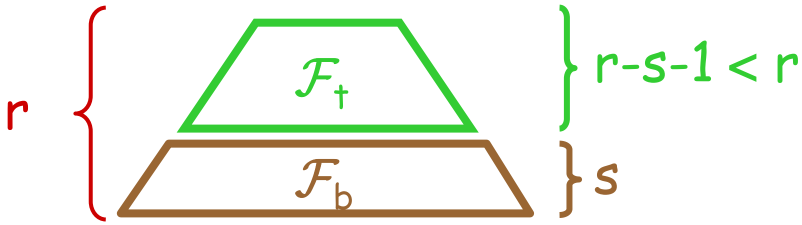


$|X_{>s}| \leq n/2^{s+1}$

$|X_{\leq s}| \leq n$

$f(M,N,R) \leq N \cdot R$

$$\text{cost}(\,C\,) \leq \underbrace{\text{cost}(\,C_t\,)}_{\leq (n/2^{s+1})\cdot r} + \text{cost}(\,C_b\,) + \underbrace{|X_b|}_{\leq n} + |C_t|$$



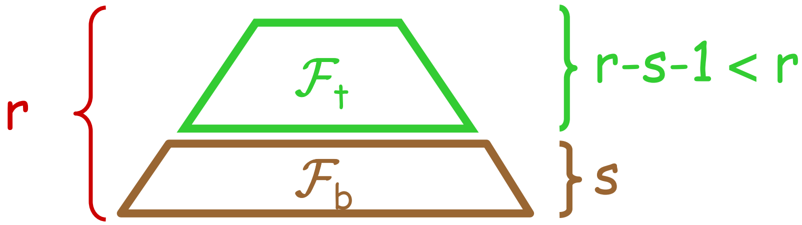
$|X_{>s}| \leq n/2^{s+1}$

$|X_{\leq s}| \leq n$

$$f(M,N,R) \leq N \cdot R$$

$$\mathrm{cost}(C) \leq \underbrace{\mathrm{cost}(C_t)}_{\leq (n/2^{s+1})\cdot r} + \mathrm{cost}(C_b) + \underbrace{|X_b|}_{\leq n} + |C_t|$$

$s = \log r$ $\qquad \leq n$



$|X_{>s}| \leq n/2^{s+1}$

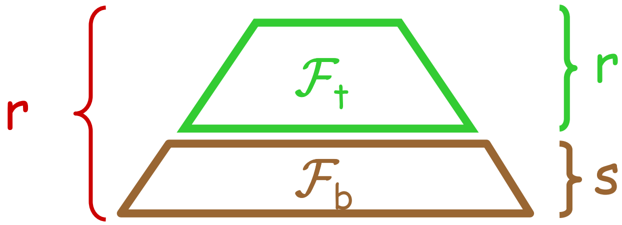
$|X_{\leq s}| \leq n$

$f(M,N,R) \leq N \cdot R$

$$\mathrm{cost}(\mathcal{C}) \leq \underbrace{\mathrm{cost}(\mathcal{C}_t)}_{\leq (n/2^{s+1}) \cdot r} + \mathrm{cost}(\mathcal{C}_b) + \underbrace{|X_b|}_{\leq n} + |\mathcal{C}_t|$$

$s = \log r$ $\quad\quad \leq n$

$$\mathrm{cost}(\mathcal{C}) \leq 2n + \mathrm{cost}(\mathcal{C}_b) + |\mathcal{C}_t|$$



$|X_{>s}| \leq n/2^{s+1}$

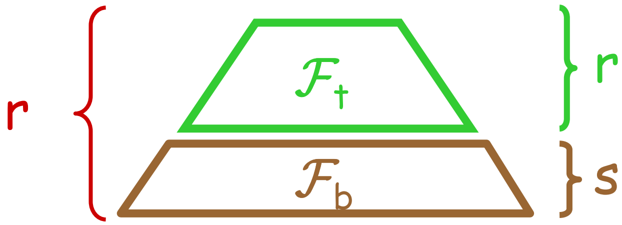
$|X_{\leq s}| \leq n$

$$f(M,N,R) \leq N \cdot R$$

$$\text{cost}(\,C\,) \leq \underbrace{\text{cost}(\,C_t\,)}_{\leq (n/2^{s+1})\cdot r} + \text{cost}(\,C_b\,) + \underbrace{|X_b|}_{\leq n} + |C_t|$$

$s = \log r$ $\qquad \leq n$

$$\text{cost}(\,C\,) \leq 2n + \text{cost}(\,C_b\,) + |C_t| \;\Big|\; - \overbrace{(|C_b| + |C_t|)}^{=|C|}$$



$|X_{>s}| \leq n/2^{s+1}$

$|X_{\leq s}| \leq n$

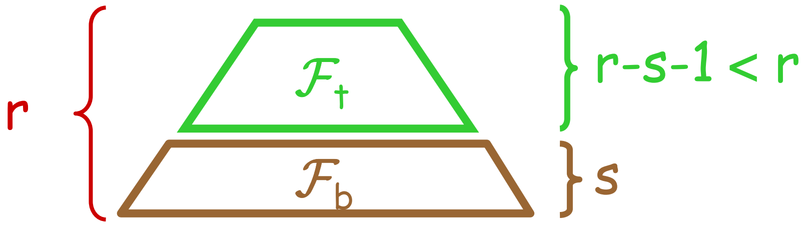
$f(M,N,R) \leq N \cdot R$

$$\mathrm{cost}(\,C\,) \leq \underbrace{\mathrm{cost}(\,C_t\,)}_{\leq (n/2^{s+1})\cdot r \;\leq\; n} + \mathrm{cost}(\,C_b\,) + \underbrace{|X_b|}_{\leq n} + |C_t|$$

$s = \log r$

$$\mathrm{cost}(\,C\,) \leq 2n + \mathrm{cost}(\,C_b\,) + |C_t| \;\Big|\; - \overbrace{(|C_b| + |C_t|)}^{=|C|}$$

$$\mathrm{cost}(\,C\,) - |C| \leq 2n + (\,\mathrm{cost}(\,C_b\,) - |C_b|\,)$$



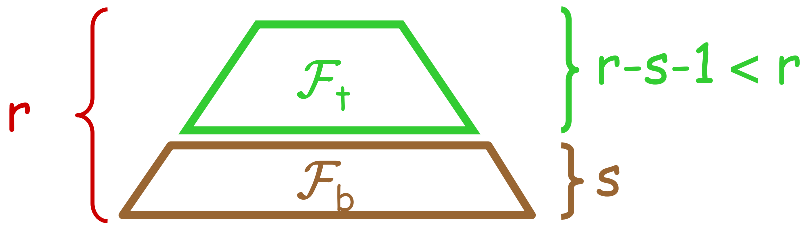
$|X_{>s}| \leq n/2^{s+1}$

$|X_{\leq s}| \leq n$

$$f(M,N,R) \leq N \cdot R$$

$$s = \log r$$

$$\mathrm{cost}(\mathcal{C}) - |\mathcal{C}| \leq 2n + (\mathrm{cost}(\mathcal{C}_b) - |\mathcal{C}_b|)$$



$|X_{>s}| \leq n/2^{s+1}$

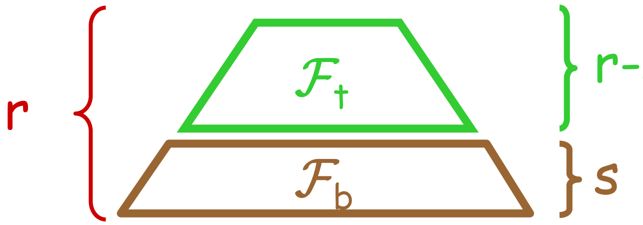
$|X_{\leq s}| \leq n$

$$f(M,N,R) \leq N \cdot R$$

$$s = \log r$$

$$\mathrm{cost}(C) - |C| \leq 2n + (\mathrm{cost}(C_b) - |C_b|)$$

$$(f(m,n,r) - m) \leq 2n + (f(m_b, n, \log r) - m_b)$$



$|X_{>s}| \leq n/2^{s+1}$

$|X_{\leq s}| \leq n$

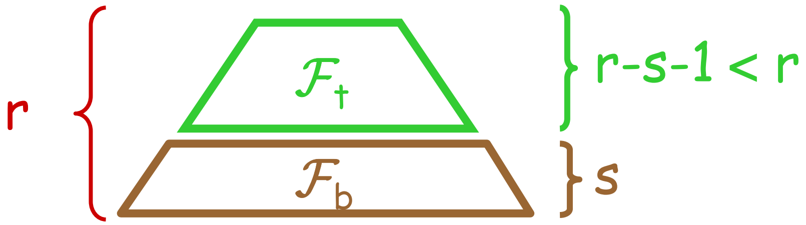
$$f(M,N,R) \leq N \cdot R$$

$$s = \log r$$

$$\text{cost}(\mathcal{C}) - |\mathcal{C}| \leq 2n + (\text{cost}(\mathcal{C}_b) - |\mathcal{C}_b|)$$

$$(f(m,n,r) - m) \leq 2n + (f(m_b,n,\log r) - m_b)$$

$$(f(m,n,r) - m) \leq 2n \cdot \log^* r$$



$|X_{>s}| \leq n/2^{s+1}$

$|X_{\leq s}| \leq n$

$f(M,N,R) \leq N \cdot R$

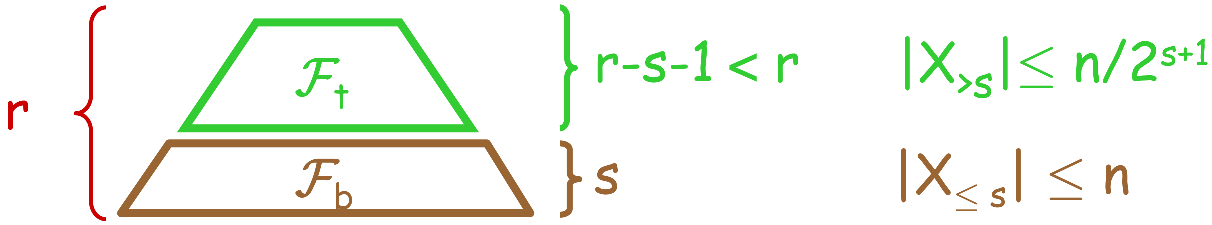
$s = \log r$

$$\text{cost}(\mathcal{C}) - |\mathcal{C}| \leq 2n + (\text{cost}(\mathcal{C}_b) - |\mathcal{C}_b|)$$

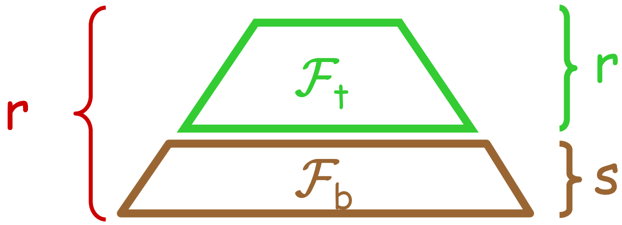
$$(f(m,n,r) - m) \leq 2n + (f(m_b, n, \log r) - m_b)$$

$$(f(m,n,r) - m) \leq 2n \cdot \log^* r$$

$$\boxed{f(m,n,r) \leq m + 2n \cdot \log^* r}$$



$$f(M,N,R) \leq M + 2N \cdot \log^* R$$

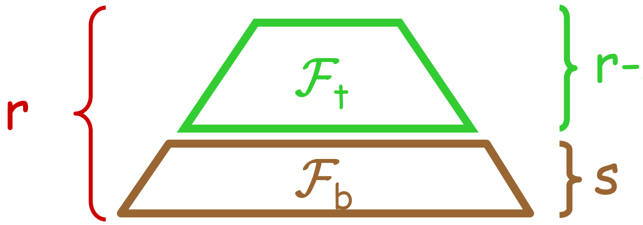


$|X_{>s}| \leq n/2^{s+1}$

$|X_{\leq s}| \leq n$

$$f(M,N,R) \leq M + 2N \cdot \log^* R$$

$$\text{cost}(\,C\,) \leq \text{cost}(\,C_t\,) + \text{cost}(\,C_b\,) + |X_b| + |C_t|$$

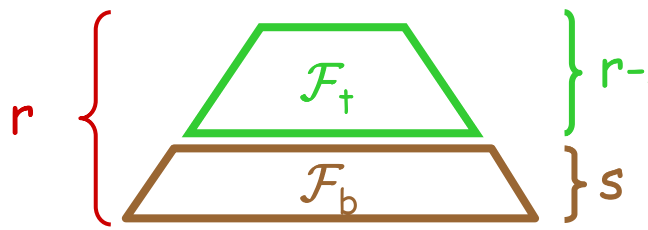


$$|X_{>s}| \leq n/2^{s+1}$$

$$|X_{\leq s}| \leq n$$

$$f(M,N,R) \leq M + 2N \cdot \log^* R$$

$$\text{cost}(\mathcal{C}) \leq \underbrace{\text{cost}(\mathcal{C}_t)}_{\leq |\mathcal{C}_t| + 2(n/2^{s+1}) \cdot \log^* r} + \text{cost}(\mathcal{C}_b) + \underbrace{|X_b|}_{\leq n} + |\mathcal{C}_t|$$



$|X_{>s}| \leq n/2^{s+1}$

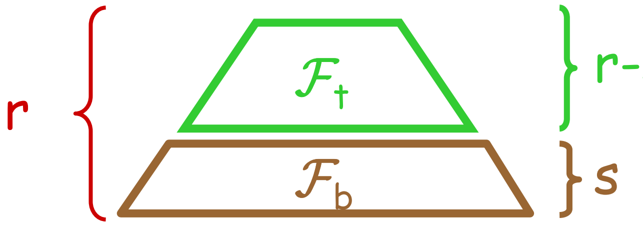
$|X_{\leq s}| \leq n$

$$f(M,N,R) \leq M + 2N \cdot \log^* R$$

$$\mathrm{cost}(\,\mathcal{C}\,) \leq \underbrace{\mathrm{cost}(\,\mathcal{C}_t\,)}_{\leq |\mathcal{C}_t| + 2(n/2^{s+1}) \cdot \log^* r} + \mathrm{cost}(\,\mathcal{C}_b\,) + \underbrace{|X_b|}_{\leq n} + |\mathcal{C}_t|$$

$s = \log \log^* r$

$$\leq |\mathcal{C}_t| + n$$



$|X_{>s}| \leq n/2^{s+1}$

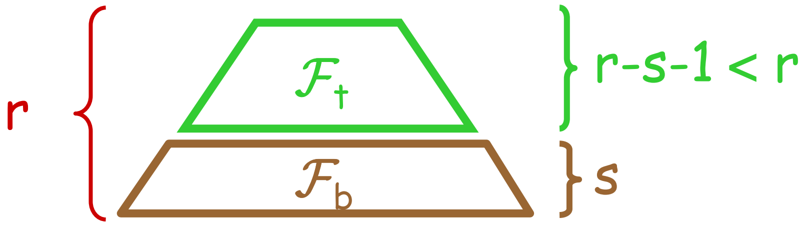
$|X_{\leq s}| \leq n$

$$f(M,N,R) \leq M + 2N \cdot \log^* R$$

$$\text{cost}(C) \leq \underbrace{\text{cost}(C_t)}_{\leq |C_t| + 2(n/2^{s+1}) \cdot \log^* r} + \text{cost}(C_b) + \underbrace{|X_b|}_{\leq n} + |C_t|$$

$s = \log \log^* r$ $\qquad \leq |C_t| + n$

$$\text{cost}(C) \leq 2n + \text{cost}(C_b) + 2|C_t|$$



$$|X_{>s}| \leq n/2^{s+1}$$

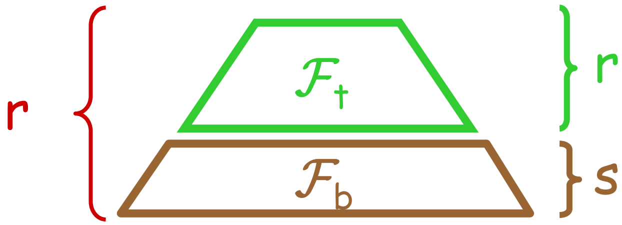
$$|X_{\leq s}| \leq n$$

$$f(M,N,R) \leq M + 2N \cdot \log^* R$$

$$\text{cost}(\,C\,) \leq \underbrace{\text{cost}(\,C_t\,)}_{\leq |C_t| + 2(n/2^{s+1}) \cdot \log^* r} + \text{cost}(\,C_b\,) + \underbrace{|X_b|}_{\leq n} + |C_t|$$

$$s = \log \log^* r \qquad \leq |C_t| + n$$

$$\text{cost}(\,C\,) \leq 2n + \text{cost}(\,C_b\,) + 2|C_t| \;\Big|\; - 2\overbrace{(|C_b| + |C_t|)}^{=|C|}$$



$|X_{>s}| \le n/2^{s+1}$

$|X_{\le s}| \le n$

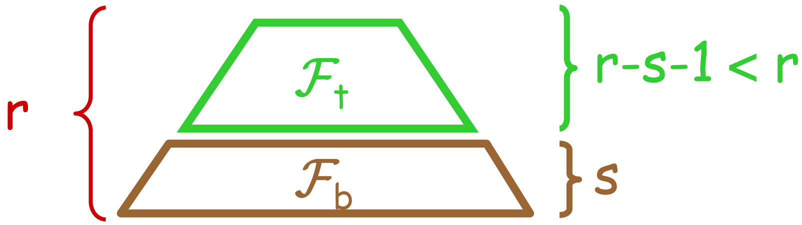
$$f(M,N,R) \le M + 2N\cdot\log^* R$$

$$\mathrm{cost}(C) \le \underbrace{\mathrm{cost}(C_t)}_{\le |C_t| + 2(n/2^{s+1})\cdot\log^* r} + \mathrm{cost}(C_b) + \underbrace{|X_b|}_{\le n} + |C_t|$$

$s = \log\log^* r$ $\quad \le |C_t| + n$

$$\mathrm{cost}(C) \le 2n + \mathrm{cost}(C_b) + 2|C_t| - 2\overbrace{(|C_b| + |C_t|)}^{=|C|}$$

$$\mathrm{cost}(C) - 2|C| \le 2n + (\mathrm{cost}(C_b) - 2|C_b|)$$



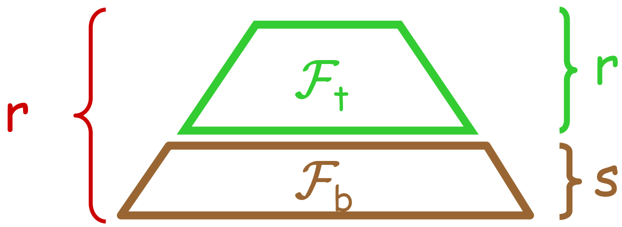
$|X_{>s}| \leq n/2^{s+1}$

$|X_{\leq s}| \leq n$

$$f(M,N,R) \leq M + 2N \cdot \log^* R$$

$$s = \log \log^* r$$

$$\text{cost}(\,C\,) - 2|C| \;\leq\; 2n + (\,\text{cost}(\,C_b\,) - 2|C_b|\,)$$



$|X_{>s}| \leq n/2^{s+1}$

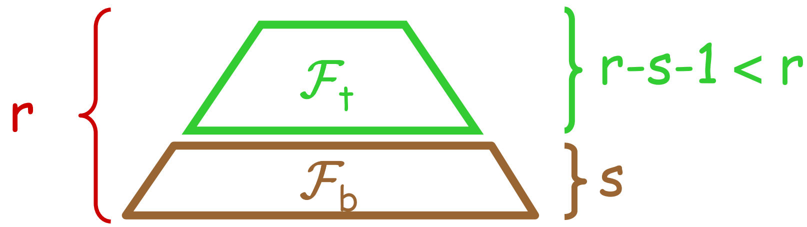
$|X_{\leq s}| \leq n$

$$f(M,N,R) \leq M + 2N \cdot \log^* R$$

$$s = \log \log^* r$$

$$\text{cost}(\mathcal{C}) - 2|\mathcal{C}| \leq 2n + (\text{cost}(\mathcal{C}_b) - 2|\mathcal{C}_b|)$$

$$(f(m,n,r) - 2m) \leq 2n + (f(m_b,n, \log \log^* r) - 2m_b)$$



$|X_{>s}| \leq n/2^{s+1}$

$|X_{\leq s}| \leq n$

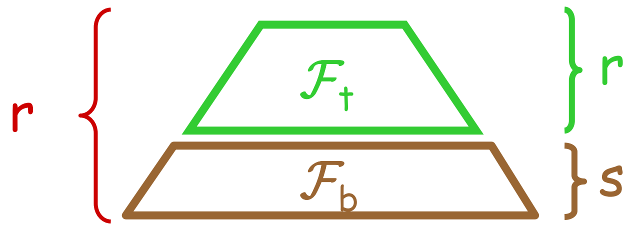
$$f(M,N,R) \leq M + 2N \cdot \log^* R$$

$$s = \log \log^* r$$

$$\text{cost}(C) - 2|C| \leq 2n + (\text{cost}(C_b) - 2|C_b|)$$

$$(f(m,n,r) - 2m) \leq 2n + (f(m_b, n, \log \log^* r) - 2m_b)$$

$$(f(m,n,r) - 2m) \leq 2n \cdot (\log \log^*)^*(r)$$



$|X_{>s}| \leq n/2^{s+1}$

$|X_{\leq s}| \leq n$

$$f(M,N,R) \leq M + 2N \cdot \log^* R$$

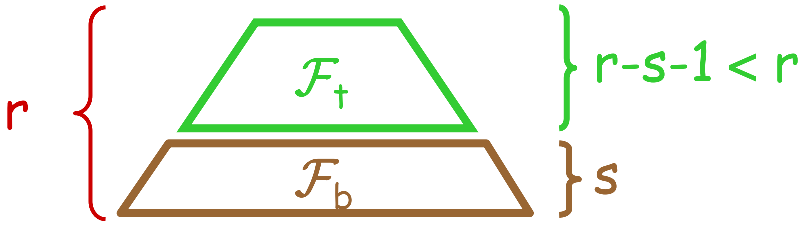
$$s = \log \log^* r$$

$$\text{cost}(\mathcal{C}) - 2|\mathcal{C}| \leq 2n + (\text{cost}(\mathcal{C}_b) - 2|\mathcal{C}_b|)$$

$$(f(m,n,r) - 2m) \leq 2n + (f(m_b,n, \log \log^* r) - 2m_b)$$

$$(f(m,n,r) - 2m) \leq 2n \cdot (\log \log^*)^*(r)$$

$$\boxed{f(m,n,r) \leq 2m + 2n \cdot (\log \log^*)^*(r)}$$



$|X_{>s}| \leq n/2^{s+1}$

$|X_{\leq s}| \leq n$

$$f(M,N,R) \leq k \cdot M + 2N \cdot g(R)$$

$$s = \log g(r)$$

$$\mathrm{cost}(\mathcal{C}) - (k+1) \cdot |\mathcal{C}| \leq 2n + (\mathrm{cost}(\mathcal{C}_b) - (k+1) \cdot |\mathcal{C}_b|)$$

$$(f(m,n,r) - (k+1) \cdot m) \leq 2n + (f(m_b, n, \log g(r)) - (k+1) \cdot m_b)$$

$$(f(m,n,r) - (k+1) \cdot m) \leq 2n \cdot (\log \circ g)^*(r)$$

$$\boxed{f(m,n,r) \leq (k+1) \cdot m + 2n \cdot (\log \circ g)^*(r)}$$

Def.: $g : \mathbb{N} \to \mathbb{N}$ "nice"

$$g^{\diamond}(r) = \begin{cases} 0 & \text{if } r \leq 1 \\ 1 + g^{\diamond}(\lceil \log_2 g(r) \rceil) & \text{if } n > 1 \end{cases}$$

Note: $g^{\diamond} = (\lceil \log_2 \rceil \circ g)^*$

Shifting Lemma:

Assume $k \geq 0$, $g: \mathbb{N} \rightarrow \mathbb{N}$, "nice", non-decreasing, $g(r) < r$ for $r > 0$.

Shifting Lemma:

Assume $k \geq 0$, $g: \mathbb{N} \rightarrow \mathbb{N}$, "nice", non-decreasing, $g(r) < r$ for $r > 0$.

If

$$f(m,n,r) \leq k \cdot m + 2 \cdot n \cdot g(r) \qquad \text{for all } m,n,r$$

Shifting Lemma:

Assume $k \geq 0$, $g:\mathbb{N}\to\mathbb{N}$, "nice", non-decreasing, $g(r)<r$ for $r>0$.

If

$$f(m,n,r) \leq k\cdot m + 2\cdot n\cdot g(r) \qquad \text{for all } m,n,r$$

then also

$$f(m,n,r) \leq (k+1)\cdot m + 2\cdot n\cdot g^{\diamond}(r) \qquad \text{for all } m,n,r$$

Def: $J_0(r) = \lceil (r-1)/2 \rceil$

$J_k(r) = J^{\diamond}_{k-1}(r)$ for $k>0$

Def: $J_0(r) = \lceil (r-1)/2 \rceil$

$J_k(r) = J_{k-1}^{\diamond}(r)$ for $k>0$

Lemma: For all $k \in \mathbb{N}$

$$f(m,n,r) \leq km + 2nJ_k(r)$$

Def: $J_0(r) = \lceil (r-1)/2 \rceil$

$J_k(r) = J_{k-1}^{\diamond}(r)$ for $k>0$

Lemma: For all $k \in \mathbb{N}$

$$f(m,n,r) \leq km + 2nJ_k(r)$$

Def: $\alpha(m,n) = \min\{\, k \in \mathbb{N} \mid J_k(\lfloor \log_2 n \rfloor) \leq 1+m/n \,\}$

Note: $r \leq \lfloor \log_2 n \rfloor$ always

Def: $J_0(r) = \lceil (r-1)/2 \rceil$

$J_k(r) = J_{k-1}^{\diamond}(r)$ for $k>0$

Lemma: For all $k \in \mathbb{N}$

$$f(m,n,r) \leq km + 2nJ_k(r)$$

Def: $\alpha(m,n) = \min\{\, k \in \mathbb{N} \mid J_k(\lfloor \log_2 n \rfloor) \leq 1 + m/n \,\}$

$$\alpha(m,n) = \min\{\, k \in \mathbb{N} \mid J_0^{\overbrace{\diamond\diamond\cdots\diamond}^{k \text{ times}}}(\lfloor \log_2 n \rfloor) \leq 1 + m/n \,\}$$

Def: $J_0(r) = \lceil (r-1)/2 \rceil$

$J_k(r) = J_{k-1}^{\diamond}(r)$ for $k>0$

Lemma: For all $k \in \mathbb{N}$

$$f(m,n,r) \le km + 2nJ_k(r)$$

Def: $\alpha(m,n) = \min\{ k \in \mathbb{N} \mid J_k(\lfloor \log_2 n \rfloor) \le 1 + m/n \}$

$$\alpha(m,n) = \min\{ k \in \mathbb{N} \mid J_0^{\overbrace{\diamond\diamond\cdots\diamond}^{k \text{ times}}}(\lfloor \log_2 n \rfloor) \le 1 + m/n \}$$

Corollary: $f(m,n,r) \le (\ \alpha(m,n) + 2\ )m + 2n$

Corollary:

Any sequence of m Union, Find operations in a universe of n elements that uses linking by rank and path compression takes time at most

$$O(m \cdot \alpha(m,n) + n)$$

Hopcroft – Ullman, Tarjan, van Leeuwen, Kozen, Harfst-Reingold;
Sharir

For $r \leq 65$: $J_1(r) \leq 2$

$J_2(r) \leq 1$

$f(m,n,r) \leq \min\{ m+4n , 2m+2n \}$ for $n<2^{66}$

For $r \leq 65$: $J_1(r) \leq 2$

$J_2(r) \leq 1$

$f(m,n,r) \leq \min\{ m+4n , 2m+2n \}$ for $n<2^{66}$

Actually:

$f(m,n,r) \leq m+2.1n$ for $n<2^{66}$

$f(m,n,r) \leq 2m+n$ for $n<2^{2^{24615}}$

Similar proof for $O(m\cdot\alpha(m,n) + n)$ bound also works for

linking by weight and path compression

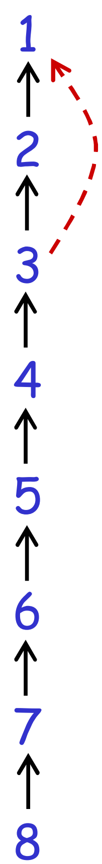
linking by rank and generalized path compaction

Heuristic 2': Path compaction

when performin a Find(x) operation make "all" nodes in the "findpath" child of some node further up.

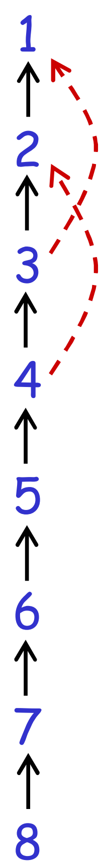
Heuristic 2': Path compaction

when performin a Find(x) operation make "all" nodes in the "findpath" child of some node further up.



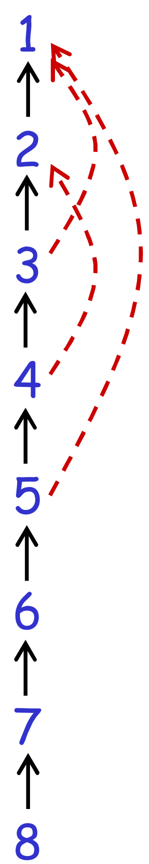
Heuristic 2': Path compaction

when performin a Find(x) operation make "all" nodes in the "findpath" child of some node further up.



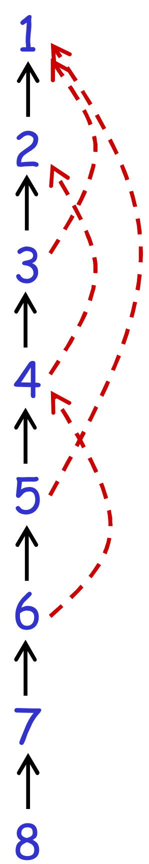
Heuristic 2': Path compaction

when performin a Find(x) operation make "all" nodes in the "findpath" child of some node further up.



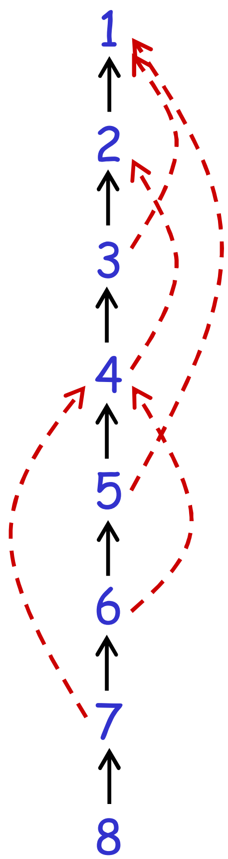
Heuristic 2': Path compaction

when performin a Find(x) operation make "all" nodes in the "findpath" child of some node further up.



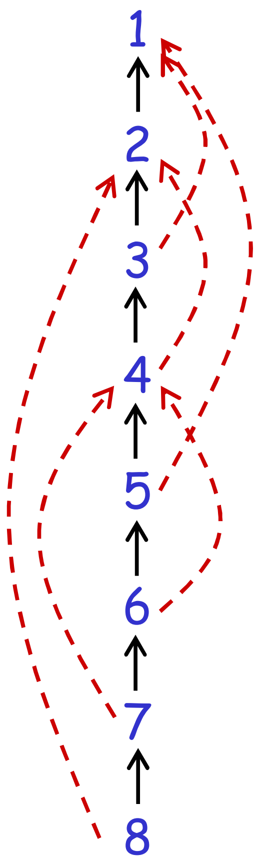
Heuristic 2': Path compaction

when performin a Find(x) operation make "all" nodes in the "findpath" child of some node further up.



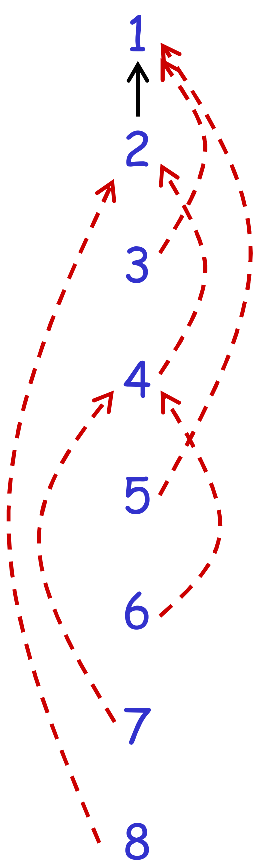
Heuristic 2': Path compaction

when performin a Find(x) operation make "all" nodes in the "findpath" child of some node further up.



Heuristic 2': Path compaction

when performin a Find(x) operation make "all" nodes in the "findpath" child of some node further up.



Heuristic 2': Path compaction

when performin a Find(x) operation make "all" nodes in the "findpath" child of some node further up.