Understanding the Inverse Ackermann Function

Raimund Seidel

Universität des Saarlandes
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 \( \alpha(m, n) \) exist; for example, \( \log_2 n \) is sometimes replaced by \( n \), and the floor function is sometimes replaced by a ceiling.
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:
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;

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
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 $n>1$.)
Recurrence: \[ X(n) \leq \begin{cases} 
0 & \text{if } n \leq 1 \\
\alpha \cdot n + \frac{n}{f(n)} \cdot X(f(n)) & \text{if } n > 1 
\end{cases} \]
Recurrence:

\[ X(n) \leq \begin{cases} 
  0 & \text{if } n \leq 1 \\
  a \cdot n + \frac{n}{f(n)} \cdot X(f(n)) & \text{if } n > 1 
\end{cases} \]

Solution:

\[ X(n) \leq a \cdot n \cdot f^*(n) \]
\[ f^*(n) = \begin{cases} 
0 & \text{if } n \leq 1 \\
1 + f^*(f(n)) & \text{if } n > 1 
\end{cases} \]
\[ f^*(n) = \begin{cases} 
0 & \text{if } n \leq 1 \\
1 + f^*(f(n)) & \text{if } n > 1 
\end{cases} \]

\[ f^*(n) = \min \left\{ k \mid f(f(\ldots f(n)\ldots)) \leq 1 \right\} \]

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

\[ f^*(n) = \min \{ k \mid f(f(\ldots f(n)\ldots)) \leq 1 \} \]

**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 $f^*$:

<table>
<thead>
<tr>
<th>$f(n)$</th>
<th>$f^*(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n-1$</td>
<td>$n-1$</td>
</tr>
<tr>
<td>$n-2$</td>
<td>$n/2$</td>
</tr>
<tr>
<td>$n-c$</td>
<td>$n/c$</td>
</tr>
<tr>
<td>$n/2$</td>
<td>$\log_2 n$</td>
</tr>
<tr>
<td>$n/c$</td>
<td>$\log_c n$</td>
</tr>
<tr>
<td>$\sqrt{n}$</td>
<td>$\log \log n$</td>
</tr>
<tr>
<td>$\log n$</td>
<td>$\log^* n$</td>
</tr>
</tbody>
</table>
Partial sum problem in the semi-group setting

Data: \( A_1, A_2, \ldots, A_n \in \text{“Semigroup” } (G,+) \)

Query: \( i,j \) \hspace{1cm} Answer: \( A_i + A_{i+1} + \cdots + A_j \)

“partial sum”
Partial sum problem in the semi-group setting

**Data:** \( A_1, A_2, \ldots, A_n \in \text{“Semigroup”} \ (G,+ ) \)

**Query:** \( i,j \) \hspace{1cm} **Answer:** \( A_i + A_{i+1} + \ldots + A_j \) \hspace{1cm} “partial sum”

**Goal:** Store “few” values of \( G \) so that each query can be answered with little \textit{cost}
Partial sum problem in the semi-group setting

**Data:** \( A_1, A_2, \ldots, A_n \in \text{“Semigroup” (G,+)} \)

**Query:** \( i,j \) \hspace{1cm} **Answer:** \( A_i+A_{i+1}+\ldots+A_j \)

**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, \ldots, A_n \in \text{"Semigroup" } (G,+))

Query: \( i,j \)  

Answer: \( A_i + A_{i+1} + \ldots + A_j \)  

"partial sum"

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

\[ S_k(n) = \# \text{ of values to be stored so that every query can be answered using at most } k \text{ "+" operations.} \]
Partial sum problem in the semi-group setting

**Data:** \( A_1, A_2, \ldots, A_n \in \text{“Semigroup” } (G,+\)\)

**Query:** \( i, j \) \hspace{1cm} **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

\[
S_k(n) = \# \text{ of values to be stored so that every query can be answered using at most } k \text{ “+” operations.}
\]

\[
S_0(n) = \binom{n+1}{2}
\]
Example semi-groups \((G,+)\):

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

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

\((d \times d \text{ matrices, mult})\)
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 $$(\text{suffix-sum}) + (\text{prefix-sum})$$
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 $A''$

$$S_1(n) \leq n + \frac{n}{(n/2)} S_1\left(\frac{n}{2}\right)$$
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 $A''$

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

\[ \Rightarrow \quad S_1(n) \leq n \cdot (n/2)^* = n \log_2 n \]
$S_3(n) = ?$
$S_3(n) = ?$

“3-op-structure”

\[
\text{case } n \leq 4 : \text{ trivial}
\]

\[
\text{case } n \geq 5 : \text{ use recursive construction}
\]
A

--------------------------------------------------
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 $\leq \log n$ each
partition $A$-sequence into $\frac{n}{\log n}$ 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 $\leq \log n$ each

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

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

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 \( \frac{n}{\log n} \) subsequence-sums

recursively build a 3-op-structure for each of the \( \frac{n}{\log n} \) subsequences

**Query answering:**

either use one of the recursive 3-op-structures
or return \((\text{suffix-sum}) + (\text{answer from 1-op-structure}) + (\text{prefix-sum})\)
Query answering:
- either use one of the recursive 3-op-structures
- or return \((\text{suffix-sum})+(\text{answer from 1-op-structure})+(\text{prefix-sum})\)
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

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

\[\leq n\]
store all prefix- and all suffix-sums within each subsequence

build a 1-op-structure for the \( \frac{n}{\log n} \) subsequence-sums

recursively build a 3-op-structure for each of the \( \frac{n}{\log n} \) subsequences

\[
S_3(n) \leq 2n + S_1\left(\frac{n}{\log n}\right) + \frac{n}{\log n} \cdot S_3\left(\log n\right) \leq 3n + \frac{n}{\log n} \cdot S_3\left(\log n\right)
\]

\[
\leq n
\]
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

\[
S_3(n) \leq 2n + S_1\left(\frac{n}{\log n}\right) + \frac{n}{\log n} \cdot S_3(\log n) \leq 3n + \frac{n}{\log n} \cdot S_3(\log n)
\]

\[\Rightarrow S_3(n) \leq 3n \log^* n\]
$S_5(n) = ? \quad S_7(n) = ? \quad S_9(n) = ?$

$S_{2k+1}(n) = ?$
$S_5(n) = ? \quad S_7(n) = ? \quad S_9(n) = ?$

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

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

realized by (2k-1)-op-structure
$S_5(n) = ? \quad S_7(n) = ? \quad S_9(n) = ?$

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

**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
partition $A$-sequence into $\frac{n}{f(n)}$ 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 $\leq f(n)$ 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
partition $A$-sequence into \( \frac{n}{f(n)} \) 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 $\frac{n}{f(n)}$ 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
partition $A$-sequence into $\frac{n}{f(n)}$ 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 $\frac{n}{f(n)}$ subsequence-sums

recursively build a $(2k+1)$-op-structure for each of the $\frac{n}{f(n)}$ 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
Query answering:

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

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 \((\text{suffix-sum})+\) (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
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

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

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

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

\[ \leq (2k-1) \frac{n}{f(n)} \cdot f( \frac{n}{f(n)} ) \]

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

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

\[ S_{2k+1}(n) \leq (2k+1)n + \frac{n}{f(n)} \cdot S_{2k+1}(f(n)) \]

\[ \Rightarrow S_{2k+1}(n) \leq (2k+1)n f^*(n) \]
\( k=1 : \quad S_1(n) \leq n \log n \)

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

\[ \Rightarrow S_{2k+1}(n) \leq (2k+1) \cdot n \cdot f^*(n) \]
$k=1: \quad S_1(n) \leq n \log n$

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

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

For all $k \geq 1: \quad S_{2k+1} \leq (2k+1) \cdot n \cdot \log^{**\ldots\ast}(n)$
For all \( k \geq 1 \): \[ S_{2k+1} \leq (2k+1) \cdot n \cdot \log^{\underbrace{**\ldots\*}}(n) \]
For all $k \geq 1$ : $S_{2k+1} \leq \underbrace{(2k+1) \cdot n \cdot \log \cdots \log}_{k \text{ times}}(n)$

Define $\alpha(n) = \min\{ k \mid \underbrace{\log \cdots \log}_{k \text{ times}}(n) \leq 2 \}$
For all $k \geq 1$:

$$S_{2k+1} \leq (2k+1) \cdot n \cdot \log^{\underbrace{\cdots}}(n)$$

Define $\alpha(n) = \min\{ k \mid \log^{\underbrace{\cdots}}(n) \leq 2 \}$

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 \( k \geq 1 \): \( S_{2k+1} \leq (2k+1) \cdot n \cdot \log^{k \text{ times}}(n) \)

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

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 $S = \{1, 2, \ldots, n\}$ under operations
Union Find with Path Compressions

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

$\text{Union}(2, 4)$
Union Find with Path Compressions

Maintain partition of \( S = \{ 1, 2, \ldots, n \} \)
under operations

\[
\text{Union}(2, 4)
\]

\[
\text{Find}(3) = 6 \quad \text{(representative element)}
\]
Implementation

* 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
Implementation

* 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

```
1 8 3
↑
7
```
```
2 4 6 5 9
↑ ↑ ↑
2 8 3
↑
1 7
```

Union(2, 4)

```
1 8 3
↑
7
```
```
2 4 6 5 9
↑ ↑ ↑
2 8 3
↑
1 7
```

“Linking”
Implementation

* 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

```
1  8  3
7
```

```
2  4  6  5  9
```

```
1  8  3
7
```

```
2  4  6  5  9
```

```
2  4  6  5  9
```

```
Find(x)  follow path from x to root
```

```
Union(2, 4)
```

```
“Linking”
```

```
“path following”
```

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 $\text{Union}(x, y)$ 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:

\[
\text{cost}(\text{op}) = \# \text{ times some node gets a new parent}
\]

Time for **Union**(x, y) = \(O(1) = O(\text{cost}(\text{Union}(x,y)))\)

Time for **Find**(x) = \(O(\# \text{ of nodes on findpath})\)
  = \(O(2 + \text{cost}(\text{Find}(x)))\)
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 $\mathcal{F}$

compress($x, y$)
General path compression in forest $\mathcal{F}$
General path compression in forest $\mathcal{F}$

$\text{cost}(\text{compress}(x, y)) = \# \text{ of nodes that get a new parent}$
General path compression in forest $\mathcal{F}$

“rootpath compress”
General path compression in forest $\mathcal{F}$

“rootpath compress”

compress($x, \infty$)
General path compression in forest $\mathcal{F}$

“rootpath compress”

$\xrightarrow{\text{compress}(x, \infty)}$
General path compression in forest $\mathcal{F}$

“rootpath compress”

$\text{compress}(x, \infty)$

$\text{cost}(\text{compress}(x, \infty)) = \# \text{ of nodes that get a new parent}$

$= 0$
Problem formulation

\( F \) forest on node set \( X \)

\( C \) sequence of compress operations on \( F \)

\(|C| = \# \) of true compress operations in \( C \)

(rootpath compresses excluded)

\( \text{cost}(C) = \sum(\text{cost of individual operations}) \)
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)

\[ \text{cost}( C ) = \sum( \text{cost of individual operations} ) \]

How large can \( \text{cost}( C ) \) 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:

$C$ ... 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 \( X \nabla X_t, X_b \) dissection for \( \mathcal{F} \) inducing subforests \( \mathcal{F}_t, \mathcal{F}_b \)

\[ \Rightarrow \exists \text{ compression sequences} \]
\( C_b \) 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 $C_+$ from $C$: 
Proof: 1) How to get $C_b$ and $C_t$ from $C$:

compression paths from $C$:

\[
\text{case 1:} \quad \begin{array}{c}
  Y \\
  \quad \downarrow
  X
\end{array} \quad \begin{array}{c}
  Y \\
  \quad \downarrow
  X
\end{array} \quad \text{into } C_t
\]
Proof: 1) How to get $C_b$ and $C_+$ from $C$:

compression paths from $C$

case 1: \[ \text{\textbullet} \] \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ X \quad \text{\textbullet} \]

\[ Y \quad \text{\textbullet} \]
Proof:  1) How to get $C_b$ and $C_t$ from $C$:

compression paths from $C$

\[
\begin{array}{c}
\text{case 1:} \\
\text{into } C_t
\end{array}
\]

\[
\begin{array}{c}
\text{case 2:} \\
\text{into } C_b
\end{array}
\]

\[
\begin{array}{c}
\text{case 3:} \\
\text{into } C_b
\end{array}
\]
Proof:

\[ |C_b| + |C_t| \leq |C| \]

compression paths from \( C \)

- **case 1:**
  - \( Y \)
  - \( X \)

- **case 2:**
  - \( Y \)
  - \( X \)
  - into \( C_t \)

- **case 3:**
  - \( Y \)
  - \( X' \)
  - \( X \)
  - into \( C_b \)
\[ \text{cost}(C) \leq \text{cost}(C_b) + \text{cost}(C_t) + |X_b| + |C_t| \]

green node gets new green parent:

brown node gets new brown parent:

brown node gets new green parent: for the first time

brown node gets new green parent: again

accounted by \( \text{cost}(C_t) \)

accounted by \( \text{cost}(C_b) \)

accounted by \( |X_b| \)

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 \( \mathcal{F} \)

- \( |X| = n \)
- \( |X_+| = |X_b| = n/2 \)

Compression sequence \( C \)

- \( |C| = m \)
Claim: \[ f(m, n) \leq (m+n) \cdot \log_2 n \]

Proof:

forest \( \mathcal{F} \)

\[ |X| = n \]

\( \mathcal{F}_t \)

\[ |X_t| = |X_b| = n/2 \]

\( \mathcal{F}_b \)

\( \mathcal{C} \) compression sequence \[ |\mathcal{C}| = m \]

Main Lemma \[ \Rightarrow \exists \; \mathcal{C}_t, \; \mathcal{C}_b \quad |\mathcal{C}_b| + |\mathcal{C}_t| \leq |\mathcal{C}| \]

\[ m_b + m_t \leq m \]

\[ \text{cost}(\mathcal{C}) \leq \text{cost}(\mathcal{C}_b) + \text{cost}(\mathcal{C}_t) + |X_b| + |C_t| \]
Claim: \( f(m,n) \leq (m+n) \cdot \log_2 n \)

Proof:

forest \( \mathcal{F} \)

\[ |X| = n \]

\[ X \]

\[ \mathcal{F}_t \]

\[ |X_t| = |X_b| = n/2 \]

\[ \mathcal{F}_b \]

\[ X_b \]

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

Proof:

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

\[ m_b + m_t \leq m \]

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

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

\[ \leq (m+n) \cdot \log_2 n \]
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:** \( F \) forest, \( x \) node in \( F \)
\[
r(x) = \text{height of subtree rooted at } x
\]
( \( r(\text{leaf}) = 0 \) )

\( F \) 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:** \( F \) forest, \( x \) node in \( F \)
\[
r(x) = \text{height of subtree rooted at } x
\]
\[
(\quad r(\text{leaf}) = 0 \quad)
\]

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

$r(x) = \text{height of subtree rooted at } x$

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

$\mathcal{F}$ 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:

\[ \mathcal{F} \text{ rank forest with maximum rank } r \text{ and node set } X \]

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

\[ X_{\leq s} = \{ x \in X \mid r(x) \leq s \} \quad \mathcal{F}_{\leq s} \]
Inheritance Lemma:

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

$s \in \mathbb{N}$:

$X_{> s} = \{ x \in X \mid r(x) > s \}$  \hspace{1cm} $\mathcal{F}_{> s}$ induced forests

$X_{\leq s} = \{ x \in X \mid r(x) \leq s \}$  \hspace{1cm} $\mathcal{F}_{\leq s}$

\begin{enumerate}
  \item $X_{\leq s}, X_{> s}$ is a dissection for $\mathcal{F}$
  \item $\mathcal{F}_{\leq s}$ is a rank forest with maximum rank $\leq s$
  \item $\mathcal{F}_{> s}$ is a rank forest with maximum rank $\leq r-s-1$
  \item $|X_{> s}| \leq |X| / 2^{s+1}$
\end{enumerate}
Inheritance Lemma:

\[ \mathcal{F} \text{ rank forest with maximum rank } r \text{ and node set } X \]

\[ s \in \mathbb{N}: \quad X_{\geq s} = \{ x \in X \mid r(x) \geq s \} \quad \mathcal{F}_{\geq s} \quad \text{induced forests} \]

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

i) \( X_{\leq s}, X_{\geq s} \) 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) \( |X_{> s}| \leq |X| / 2^{s+1} \)
\[ f(m,n,r) = \text{maximum cost of any compression sequence } C, \text{ with } |C|=m, \text{ in rank forest } F \text{ with } n \text{ nodes and maximum rank } r. \]
$$f(m,n,r) = \text{maximum cost of any compression sequence } C, \text{ with } |C|=m, \text{ in rank forest } \mathcal{F} \text{ with } n \text{ nodes and maximum rank } r.$$ 

Trivial bounds:

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

$$f(m,n,r) \leq (r-1) \cdot m$$
\[
\begin{align*}
\mathcal{F}_r^+ & \quad \{ \quad r-s-1 < r \quad \}
\mathcal{F}_b^-
\quad \{ \quad s \quad \}
\end{align*}
\]

\[
f(M,N,R) \leq N \cdot R
\]
\[ f(M,N,R) \leq N \cdot R \]
\[ \cost(C) \leq \cost(C_t) + \cost(C_b) + |X_b| + |C_t| \]
$$f(M,N,R) \leq N \cdot R$$

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

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

$$|X_{\leq s}| \leq n$$
\[ f(M, N, R) \leq N \cdot R \]

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

\[ s = \log r \leq n \]
\[
f(M, N, R) \leq N \cdot R
\]

\[
\text{cost}(C) \leq \text{cost}(C_t) + \text{cost}(C_b) + |X_b| + |C_t| \leq (n/2^{s+1}) \cdot r \leq n
\]

\[
s = \log r \leq n
\]

\[
\text{cost}(C) \leq 2n + \text{cost}(C_b) + |C_t|
\]
\[ f(M,N,R) \leq N \cdot R \]

\[
\text{cost}(C) \leq \underbrace{\text{cost}(C_+)}_{\leq (n/2^{s+1}) \cdot r} + \underbrace{\text{cost}(C_b)}_{\leq n} + |X_b| + |C_+| = |C|
\]

\[
\text{cost}(C) \leq 2n + \underbrace{\text{cost}(C_b)}_{\text{cost}(C_b) + |C_+|} + \underbrace{|C_+|}_{|C_b| + |C_+|} - (|C_b| + |C_+|)
\]
\[ f(M, N, R) \leq N \cdot R \]

\[
\text{cost}(C) \leq \underbrace{\text{cost}(C_+)}_{\leq (n/2^{s+1}) \cdot r} + \underbrace{\text{cost}(C_b) + |X_b| + |C_+|}_{\leq n} \\
\]

\[ s = \log r \leq n \]

\[
\text{cost}(C) \leq 2n + \text{cost}(C_b) + |C_+| - (|C_b| + |C_+|) \\
\]

\[
\text{cost}(C) - |C| \leq 2n + (\text{cost}(C_b) - |C_b|) \\
\]
\[
f(M, N, R) \leq N \cdot R
\]

\[s = \log r\]

\[
\text{cost}(C) - |C| \leq 2n + (\text{cost}(C_b) - |C_b|)
\]
\[
f(M,N,R) \leq N \cdot R
\]

\[s = \log r\]

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

\[
(f(m,n,r) - m) \leq 2n + (f(m_b,n,\log r) - m_b)
\]
\( f(M,N,R) \leq N \cdot R \)

\( s = \log r \)

\[
\begin{align*}
\text{cost}( C ) - |C| & \leq 2n + ( \text{cost}( C_b ) - |C_b| ) \\
( f(m,n,r) - m ) & \leq 2n + ( f(m_b,n,\log r) - m_b ) \\
( f(m,n,r) - m ) & \leq 2n \cdot \log^* r
\end{align*}
\]
\[
\begin{align*}
\{ & r \} \quad \{ & r-s-1 < r \quad |X_{>s}| \leq n/2^{s+1} \\
& s \} \quad |X_{\leq s}| \leq n
\end{align*}
\]

\[f(M,N,R) \leq N \cdot R\]

\[s = \log r\]

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

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

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

\[
f(m,n,r) \leq m + 2n \cdot \log^* r
\]
\[ f(M,N,R) \leq M + 2N \cdot \log^* R \]
\[ f(M,N,R) \leq M + 2N \cdot \log^* R \]

\[ \text{cost}(C) \leq \text{cost}(C_t) + \text{cost}(C_b) + |X_b| + |C_t| \]
\[
f(M, N, R) \leq M + 2N \cdot \log^* R
\]

\[
\text{cost}( C ) \leq \underbrace{\text{cost}( C_t )}_{\text{cost}( C_{\mathcal{F}_b} ) + |X_b| + |C_t|} + \underbrace{\text{cost}( C_b )}_{|C_t| + 2(n/2^{s+1}) \cdot \log^* r} \leq n
\]
\[ f(M,N,R) \leq M + 2N \cdot \log^* R \]

\[
\text{cost}(C) \leq \underbrace{\text{cost}(C_t)} + \underbrace{\text{cost}(C_b)} + |X_b| + |C_t|
\]

\[
\leq |C_t| + 2(n/2^{s+1}) \cdot \log^* r \leq n
\]

\[
s = \log \log^* r \leq |C_t| + n
\]
\[ f(M,N,R) \leq M + 2N \cdot \log^* R \]

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

\[ \leq |C_t| + 2(n/2^{s+1}) \cdot \log^* r \leq n \]

\[ s = \log \log^* r \leq |C_t| + n \]

\[ \text{cost}(C) \leq 2n + \text{cost}(C_b) + 2|C_t| \]
\[
f(M,N,R) \leq M + 2N \cdot \log^* R
\]

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

\[
s = \log \log^* r \leq |C_t| + n
\]

\[
\text{cost}(C) \leq 2n + \text{cost}(C_b) + 2|C_t| - 2(|C_b| + |C_t|)
\]
\[ f(M, N, R) \leq M + 2N \cdot \log^* R \]

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

\[ s = \log \log^* r \leq |C_+| + n = |C| \]

\[
\text{cost}(C) \leq 2n + \text{cost}(C_b) + 2|C_+| - 2(|C_b| + |C_+|) \\
\]

\[
\text{cost}(C) - 2|C| \leq 2n + (\text{cost}(C_b) - 2|C_b|)
\]
\[
f(M,N,R) \leq M + 2N \cdot \log^* R
\]

\[
s = \log \log^* r
\]

\[
\text{cost}(C) - 2|C| \leq 2n + (\text{cost}(C_b) - 2|C_b|)
\]
\[ f(M,N,R) \leq M + 2N \cdot \log^* R \]

\[ s = \log \log^* r \]

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

\[ (f(m,n,r) - 2m) \leq 2n + (f(m_b,n, \log \log^* r) - 2m_b) \]
\[ r \left\{ \begin{array}{l} \mathcal{F}_+ \\ \mathcal{F}_b \end{array} \right\} \begin{array}{l} r-s-1 < r \\ s \end{array} \] 

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

\[ |X_{\leq s}| \leq n \]

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

\[ s = \log \log^* r \]

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

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

\[ (f(m,n,r) - 2m) \leq 2n \cdot (\log \log^*)^*(r) \]
\[ f(M,N,R) \leq M + 2N \cdot \log^* R \]

\[ s = \log \log^* r \]

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

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

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

\[ f(m,n,r) \leq 2m + 2n \cdot (\log \log^*)^*(r) \]
\[
\begin{aligned}
f(M,N,R) &\leq k \cdot M + 2N \cdot g(R) \\
\end{aligned}
\]

\[
\begin{aligned}
s &= \log g(r) \\
\text{cost}(C) - (k+1) \cdot |C| &\leq 2n + (\text{cost}(C_b) - (k+1) \cdot |C_b|) \\
(f(m,n,r) - (k+1) \cdot m) &\leq 2n + (f(m_b,n, \log g(r)) - (k+1) \cdot m_b) \\
(f(m,n,r) - (k+1) \cdot m) &\leq 2n \cdot (\log \circ g)^*(r) \\
f(m,n,r) &\leq (k+1) \cdot m + 2n \cdot (\log \circ g)^*(r)
\end{aligned}
\]
Def.: \( g : \mathbb{N} \to \mathbb{N} \quad \text{“nice”} \)

\[
g^\circ(r) = \begin{cases} 
0 & \text{if } r \leq 1 \\
1 + g^\circ(\lfloor \log_2 g(r) \rfloor) & \text{if } n > 1
\end{cases}
\]

Note: \( g^\circ = (\lfloor \log_2 \rfloor \circ g)^* \)
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
$$f(m,n,r) \leq k \cdot m + 2 \cdot n \cdot g(r)$$
for all $m,n,r$.
**Shifting Lemma:**
Assume $k \geq 0$, $g: \mathbb{N} \rightarrow \mathbb{N}$, “nice”, non-decreasing, $g(r) < r$ for $r > 0$.

If
$$f(m, n, r) \leq k \cdot m + 2 \cdot n \cdot g(r)$$
for all $m, n, r$

then also
$$f(m, n, r) \leq (k+1) \cdot m + 2 \cdot n \cdot g(r)$$
for all $m, n, r$
Def: $J_0(r) = \lceil (r-1)/2 \rceil$

$J_k(r) = J_{k-1}(r) \text{ for } k>0$
Def: \[ J_0(r) = \lceil (r-1)/2 \rceil \]
\[ J_k(r) = J_{k-1}(r) \quad \text{for } k>0 \]

Lemma: For all \( k \in \mathbb{N} \)
\[ f(m,n,r) \leq km + 2nJ_k(r) \]
Def: \[ J_0(r) = \lceil (r-1)/2 \rceil \]
\[ J_k(r) = J_{k-1}^\circ(r) \quad \text{for } k > 0 \]

Lemma: For all \( k \in \mathbb{N} \)
\[ f(m,n,r) \leq km + 2nJ_k(r) \]

Def: \[ \alpha(m,n) = \min\{ k \in \mathbb{N} \mid J_k(\lceil \log_2 n \rceil) \leq 1 + m/n \} \]

Note: \( r \leq \lceil \log_2 n \rceil \) always
Def: \[ J_0(r) = \lceil (r-1)/2 \rceil \]
\[ J_k(r) = J_{k-1}(r) \quad \text{for } k > 0 \]

**Lemma:** For all \( k \in \mathbb{N} \)
\[ f(m,n,r) \leq km + 2nJ_k(r) \]

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

\[ \alpha(m,n) = \min \{ k \in \mathbb{N} \mid J_0^{\otimes \cdots \otimes} (\lfloor \log_2 n \rfloor) \leq 1 + m/n \} \]
Def: \( J_0(r) = \lceil (r-1)/2 \rceil \)
\[
J_k(r) = J_{k-1}(r) \quad \text{for } k > 0
\]

Lemma: For all \( k \in \mathbb{N} \)
\[
f(m,n,r) \leq km + 2nJ_k(r)
\]

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

\[
\alpha(m,n) = \min\{ k \in \mathbb{N} | J_0 \underbrace{\circ \cdots \circ}_{k \text{ times}} (\lfloor \log_2 n \rfloor) \leq 1 + m/n \}
\]

Corollary: \( f(m,n,r) \leq (\alpha(m,n) + 2)m + 2n \)
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 $r \leq 65$: 
\[ J_1(r) \leq 2 \]
\[ J_2(r) \leq 1 \]

\[ f(m,n,r) \leq \min\{ m+4n, 2m+2n \} \text{ for } n<2^{66} \]
For $r \leq 65$: \[ J_1(r) \leq 2 \\
J_2(r) \leq 1 \]

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

Actually:
\[ f(m,n,r) \leq m+2.1n \quad \text{for } n<2^{66} \]
\[ f(m,n,r) \leq 2m+n \quad \text{for } n<2^{2^{24615}} \]
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 performing a \texttt{Find} ( x ) operation make "all" nodes in the "findpath" child of some node further up.
Heuristic 2': **Path compaction**

when performing a *Find*(*x*) operation make "all" nodes in the "findpath" child of some node further up.
Heuristic 2': Path compaction

when performing a $\textbf{Find}(x)$ operation make “all” nodes in the “findpath” child of some node further up.
Heuristic 2': Path compaction

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

1
2
3
4
5
6
7
8
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 performing a $\textbf{Find}(x)$ operation make "all" nodes in the "findpath" child of some node further up.
Heuristic 2': Path compaction

when performing a \texttt{Find}( x ) operation make “all” nodes in the “findpath” child of some node further up.
Heuristic 2': Path compaction

when performin a \textbf{Find}(x) operation make “all” nodes in the “findpath” child of some node further up.
Heuristic 2': **Path compaction**

when performing a $\text{Find}( \times )$ operation make "all" nodes in the "findpath" child of some node further up.