# Inverse Ackermann Function

Ian Chen

# Outline

# Section 1

## The Function

# Motivation

This function is meant to capture different orders of growth.

1. linear
2. logarithmic
3. iterated logarithmic

# Tarjan's Definition

**Definition (Tarjan '75)**

$$A(i,j) = \begin{cases} 2j & \textbf{for } i = 0 \land j \geq 1 \\ 1 & \textbf{for } i \geq 1 \land j = 0 \\ A(i-1, A(i,j-1)) & \textbf{otherwise} \end{cases}$$

**Definition (Tarjan '75)**

$$\alpha(i,x) = \min \{\, j \mid A(i,j) \geq x \,\}$$

# Definition

**Definition**

$$\alpha(i,j) = \begin{cases} 0 & \textbf{for } j = 1 \\ \lfloor j/2 \rfloor & \textbf{for } i = 1 \wedge j > 1 \\ 1 + \alpha(i, \alpha(i-1, j)) & \textbf{otherwise} \end{cases}$$

**Definition**

$$\beta(i,j) = \begin{cases} 0 & \textbf{for } j = 1 \\ \lfloor \sqrt{j} \rfloor & \textbf{for } i = 1 \wedge j > 1 \\ 1 + \beta(i, \beta(i-1, j)) & \textbf{otherwise} \end{cases}$$

# A sequence of functions

- $\alpha(1, n) = \lfloor n/2 \rfloor$
- $\alpha(2, n) = \lg n$
- $\alpha(3, n) = \lg^* n$

- $\beta(1, n) = \lfloor \sqrt{n} \rfloor$
- $\beta(2, n) = \lg \lg n$

Section 2

# RMQ Preprocessing

# A Short but Cute Result

Suppose we are given an array of $n$ integers. We want to be able to answer *range minimum queries* in few steps.

$$\lambda(2k, n) = \alpha(k, n) \qquad \lambda(2k+1, n) = \beta(k, n)$$

**Theorem (Alon '87)**

We can answer range minimum queries in $k$ steps using $\mathcal{O}(nk\lambda(k, n))$ preprocessing space.

# Divide and Conquer

Let $T_k(n)$ be the time to preprocess an array of $n$ elements for $k$-step queries.

$$T_2(n) = 2n \lg n$$
$$T_3(n) = 3n \lg \lg n$$
$$T_k(n) = \frac{n}{\lambda(k-2,n)} T_k(\lambda(k-2,n)) + T_{k-2}(n/\lambda(k-2,n)) + 2n$$

$$T_k(n) = \mathcal{O}(nk\lambda(k,n))$$

## Additional Results

- The analysis is tight.
- The best linear time preprocessing takes $\alpha(n)$ steps.
- Tree queries, linear queries.

Section 3

# Disjoint Sets

## Union-Find

Given a set of elements, label them 1 through $n$. We want to partition these elements into buckets, supporting the operations

- Union
- Find

# Implementation

$\underline{\textsc{Find}(x)}$:
    $y \leftarrow x$
    while $y \neq parent(y)$
        $y \leftarrow parent(y)$
    $\textsc{Compress}(x, y)$
    return $x$

$\underline{\textsc{Compress}(x, y)}$:
    if $x \neq y$
        $\textsc{Compress}(parent(x), y)$
        $parent(x) \leftarrow parent(y)$

# Implementation

$\underline{\textsc{UnionLeader}(x, y)}$:
    if $rank(x) > rank(y)$
        $leader(y) \leftarrow x$
    else
        $leader(x) \leftarrow y$
        if $rank(x) = rank(y)$
            $rank(x) \leftarrow rank(x) + 1$

$\underline{\textsc{Union}(x, y)}$:
    $\bar{x} \leftarrow \textsc{Find}(x)$
    $\bar{y} \leftarrow \textsc{Find}(y)$
    $\textsc{UnionLeader}(\bar{x}, \bar{y})$

# Analysis

1. leader ranks only increase
2. $\mathit{parent}(x)$ has lower rank than $x$
3. $\mathit{size}(\overline{x})$ is at least $2^{\mathit{rank}(\overline{x})}$
4. For any rank $r$, there are at most $n/2^r$ elements of rank $r$

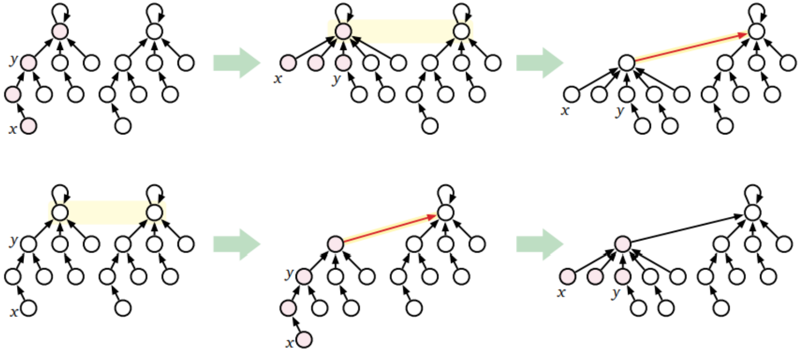
# Analysis

1. Amortized analysis
2. Number of pointer operations
3. Find is dominated by Compress
4. can make all calls to UnionLeader before Compress
5. can Shatter the tree into two forests

## Compress and Shatter

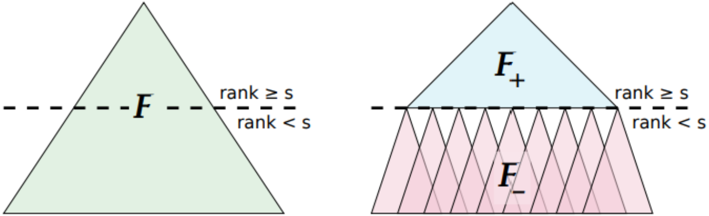
```
COMPRESS(x, y, F):
    if rank(x) ≥ s
        COMPRESS(x, y, F+)
    else if rank(y) < s
        COMPRESS(x, y, F−)
    else
        z ← x
        while rank(z) < s
            z' ← parent(z)
            parent(z) ← z
            z ← z'
        parent(z) ← z
        COMPRESS(parent(z), y, F+)
```

# Analysis



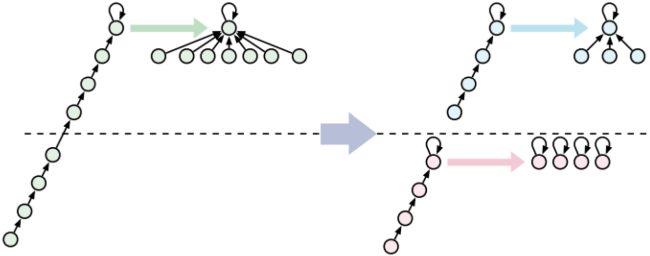

Top row: A COMPRESS followed by a UNION. Bottom row: The same operations in the opposite order.

# Analysis




Splitting the forest $F$ (in this case, a single tree) into sub-forests $F_+$ and $F_-$ at rank $s$.

# Analysis



# Analysis

Let $T(m, n, r)$ be the number of pointer operations that $m$ Compress operation takes, on a tree with $n$ nodes and maximum rank $r$.

**Theorem**

$$T(m, n, r) \leq nr$$

# Analysis

Consider the sequences of $m_+$ and $m_-$ Compress calls on $F_+$ and $F_-$.

$$T(m, n, r) \leq T(m_+, n_+, r) + T(m_-, n_-, r) + m_+ + n$$

# Analysis

Let $s = \lg r$.

Since $n_+ < n/2^s$, we have that

$$T(m_+, n_+, r) \leq rn_+ \leq rn/2^s = n$$
$$T(m, n, r) \leq T(m_-, n_-, \lg r) + m_+ + 2n$$

Letting $T'(m, n, r) = T(m, n, r) - m$,

$$T'(m, n, r) \leq T'(m, n, \lg r) + 2n$$

**Theorem**

$$T(m, n, r) \leq m + 2n \lg^* r$$

# Analysis

Let $s = \lg^* r$.

Since $n_+ < n/2^s$, we have that

$$T(m_+, n_+, r) \leq m_+ + 2n_+ \lg^* r \leq m_+ + 2n \frac{\lg^* r}{2^{\lg^* r}} \leq m + 2n$$

Letting $T'(m,n,r) = T(m,n,r) - 2m$,

$$T'(m,n,r) \leq T'(m,n,\lg^* r) + 3n$$

**Theorem**

$$T(m,n,r) \leq 2m + 3n \lg^{**} r$$

# Analysis

**Theorem**

$$T(m,n,r) \leq cm + (c+1)n \lg^{*^c} r$$

$$T(m,n,r) \leq cm + (c+1)n\alpha(c,r)$$

**Theorem**

$$T(m,n,r) \leq m\alpha(n)$$

Questions?

# Brainteaser (Erickson)

Consider the following game. I choose a positive integer n and keep it secret; your goal is to discover this integer. We play the game in rounds. In each round, you write a list of at most n integers on the blackboard. If you write more than n numbers in a single round, you lose. If n is one of the numbers you wrote, you win the game; otherwise, I announce which of the numbers you wrote is smaller or larger than n, and we proceed to the next round.

Describe a strategy that wins in $\mathcal{O}(\alpha(n))$ rounds.

*WAGA WAGA*

— Sariel Har-Peled (2024)

*All problems in computer science can be solved by another level of indirection.*

— David Wheeler (2014)

# Bibliography I


- Tar75, Tarjan, *Efficiency of a good but not linear set union algorithm*
- Alon87, Alon & Schieber, *Optimal Preprocessing for Answering On-Line Product Queries*
- Jeffe, Erickson, *Data Structures for Disjoint Sets*