# Analysis of Fibonacci Heaps

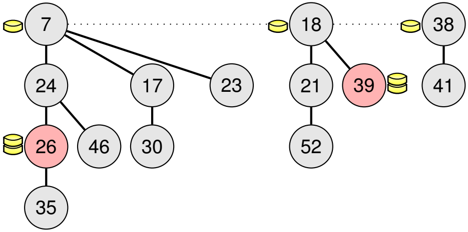
He Sun

University of BRISTOL

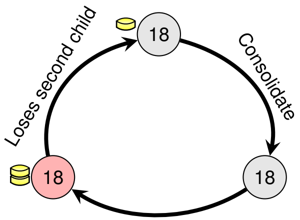
## Amortized Analysis via Potential Method

- INSERT: actual $\mathcal{O}(1)$ amortized $\mathcal{O}(1)$ ✓
- EXTRACT-MIN: actual $\mathcal{O}(\text{trees}(H) + d(n))$ amortized $\mathcal{O}(d(n))$ ?
- DECREASE-KEY: actual $\mathcal{O}(\text{\# cuts}) \leq \mathcal{O}(\text{marks}(H))$ amortized $\mathcal{O}(1)$ ?

$$\Phi(H) = \text{trees}(H) + 2 \cdot \text{marks}(H)$$




Lifecycle of a node



## Amortized Analysis of DECREASE-KEY

**Actual Cost**

- DECREASE-KEY: $\mathcal{O}(x+1)$, where $x$ is the number of cuts.

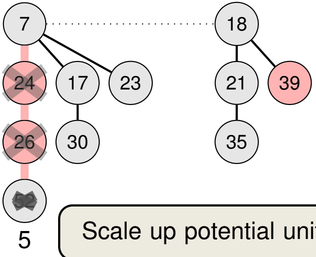
$$\Phi(H) = \mathrm{trees}(H) + 2 \cdot \mathrm{marks}(H)$$

First Coin $\rightsquigarrow$ pays cut
Second Coin $\rightsquigarrow$ increase of $\mathrm{trees}(H)$

**Change in Potential**

- $\mathrm{trees}(H') = \mathrm{trees}(H) + x$
- $\mathrm{marks}(H') \leq \mathrm{marks}(H) - x + 2$

$\Rightarrow \Delta\Phi \leq x + 2 \cdot (-x + 2) = 4 - x.$




Scale up potential units

**Amortized Cost**

$$\widetilde{c}_i = c_i + \Delta\Phi \leq \mathcal{O}(x+1) + 4 - x = \mathcal{O}(1)$$

# Amortized Analysis of EXTRACT-MIN

Actual Cost

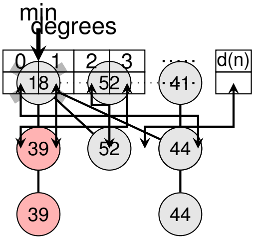
- EXTRACT-MIN: $\mathcal{O}(\text{trees}(H) + d(n))$

$$\Phi(H) = \text{trees}(H) + 2 \cdot \text{marks}(H)$$

Change in Potential

- $\text{marks}(H')$ ? $\leq \text{marks}(H)$
- $\text{trees}(H') \leq d(n) + 1$

$\Rightarrow \Delta\Phi \leq d(n) + 1 - \text{trees}(H)$




Amortized Cost

$$\widetilde{c}_i = c_i + \Delta\Phi \leq \mathcal{O}(\text{trees(H)} + d(n)) + d(n) + 1 - \text{trees(H)} = \mathcal{O}(d(n))$$

How to bound $d(n)$?

# Fibonacci Numbers

For $k = 2, 3, \ldots$, the $k$th Fibonacci number is defined by

$$F_k = F_{k-1} + F_{k-2}.$$

In particular, $F_0 = 0$ and $F_1 = 1$.

We can write

$$F_i = \frac{\phi^i - \hat{\phi}^i}{\sqrt{5}},$$

where

$$\phi = \frac{1+\sqrt{5}}{2} \approx 1.618, \qquad \hat{\phi} = \frac{1-\sqrt{5}}{2} \approx -0.618.$$

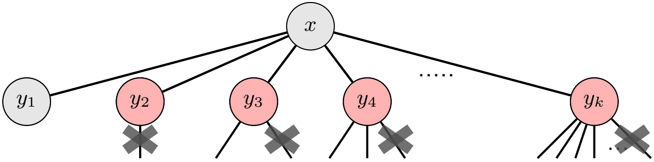
# Lower Bounding Degrees of Children

We will prove a stronger statement:
Any tree with degree $k$ contains at least $\varphi^k$ nodes.

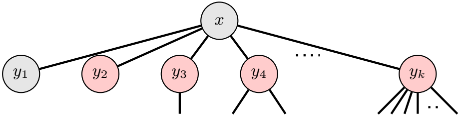
$$d(n) \le \log_\varphi n$$

- Consider any node $x$ of degree $k$ (not necessarily a root) at the final state
- Let $y_1, y_2, \ldots, y_k$ be the children in the order of attachment and $d_1, d_2, \ldots, d_k$ be their degrees

$\Rightarrow$ $\boxed{\forall 1 \le i \le k: \quad d_i \ge i - 2}$



# From Degrees to Minimum Subtree Sizes




$$\forall 1 \le i \le k: \quad d_i \ge i - 2$$

**Definition**

Let $N(k)$ be the minimum possible number of nodes of a subtree rooted at a node of degree $k$.

By induction, we have that

$$N(k) \ge 2 + \sum_{i=2}^{k} N(i-2).$$

**Homework**

$N(k) \ge F(k+2)$, where $F(k)$ is the $k$th Fibonacci number.

# Exponential Growth of Fibonacci Numbers

**Lemma 19.4**

For all integers $k \geq 0$, the $(k+2)$nd Fib. number satisfies $F(k+2) \geq \varphi^k$, where $\varphi = (1+\sqrt{5})/2 = 1.61803\ldots$.

$\varphi^2 = \varphi + 1$

Fibonacci Numbers grow at least exponentially fast in $k$.

Proof by induction on $k$:

- Base $k = 0$: $F(2) = 1$ and $\varphi^0 = 1$ ✓
- Base $k = 1$: $F(3) = 2$ and $\varphi^1 \approx 1.619 < 2$ ✓
- Inductive Step $(k \geq 2)$:

$$\begin{aligned}
F(k+2) &= F(k+1) + F(k) \\
&\geq \varphi^{k-1} + \varphi^{k-2} && \text{(by the inductive hypothesis)} \\
&= \varphi^{k-2} \cdot (\varphi + 1) \\
&= \varphi^{k-2} \cdot \varphi^2 && (\varphi^2 = \varphi + 1) \\
&= \varphi^k && \square
\end{aligned}$$

## Putting the Pieces Together

**Amortized Analysis**

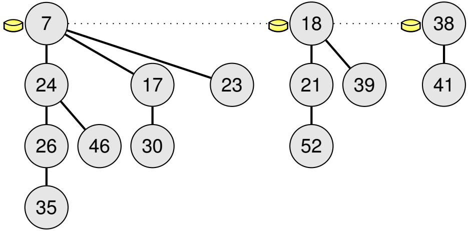
- INSERT: amortized cost $\mathcal{O}(1)$
- EXTRACT-MIN amortized cost $\cancel{\mathcal{O}(d(n))}$ $\mathcal{O}(\log n)$
- DECREASE-KEY amortized cost $\mathcal{O}(1)$

$$
\begin{aligned}
& n \geq N(k) \geq F(k+2) \geq \varphi^k \\
\Rightarrow \quad & \log_\varphi n \geq k
\end{aligned}
$$

## What if we don't have marked nodes?

- INSERT: actual $\mathcal{O}(1)$ amortized $\mathcal{O}(1)$
- EXTRACT-MIN: actual $\mathcal{O}(\text{trees}(H) + d(n))$ amortized $\mathcal{O}(d(n)) \neq \mathcal{O}(\log n)$
- DECREASE-KEY: actual $\mathcal{O}(1)$ amortized $\mathcal{O}(1)$

$$\Phi(H) = \text{trees}(H)$$



# Summary

| Operation | Linked list | Binary heap | Binomial heap | Fibon. heap |
|---|---|---|---|---|
| MAKE-HEAP | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ | $\mathcal{O}(1)$ |
| INSERT | $\mathcal{O}(1)$ | $\mathcal{O}(\log n)$ | $\mathcal{O}(\log n)$ | $\mathcal{O}(1)$ |
| MINIMUM | $\mathcal{O}(n)$ | $\mathcal{O}(1)$ | $\mathcal{O}(\log n)$ | $\mathcal{O}(1)$ |
| EXTRACT-MIN | $\mathcal{O}(n)$ | $\mathcal{O}(\log n)$ | $\mathcal{O}(\log n)$ | $\mathcal{O}(\log n)$ |
| UNION | $\mathcal{O}(n)$ | $\mathcal{O}(n)$ | $\mathcal{O}(\log n)$ | $\mathcal{O}(1)$ |
| DECREASE-KEY | $\mathcal{O}(1)$ | $\mathcal{O}(\log n)$ | | |
| DELETE | $\mathcal{O}(1)$ | $\mathcal{O}(\log n)$ | | |

Can we perform EXTRACT-MIN in $o(\log n)$?

If this was possible, then there would be a sorting algorithm with runtime $o(n \log n)$!

DELE

EXTRACT-MIN = MIN + DELETE

# Recent Studies of Fibonacci Heaps

- Fibonacci Numbers were discovered >800 years ago
- Fibonacci Heaps were developed by Fredman and Tarjan in 1984

**Brodal, Lagogiannis, Tarjan: Strict Fibonacci Heap, (STOC'12)**

Strict Fibonacci Heap:

- pointer-based heap implementation similar to Fibonacci Heaps
- achieves the same cost as Fibonacci Heaps, but actual costs!