Fibonacci Heaps

He Sun

University of BRISTOL

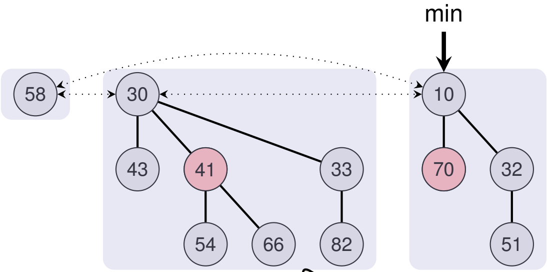
Priority Queues Overview

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

Structure of Fibonacci Heaps

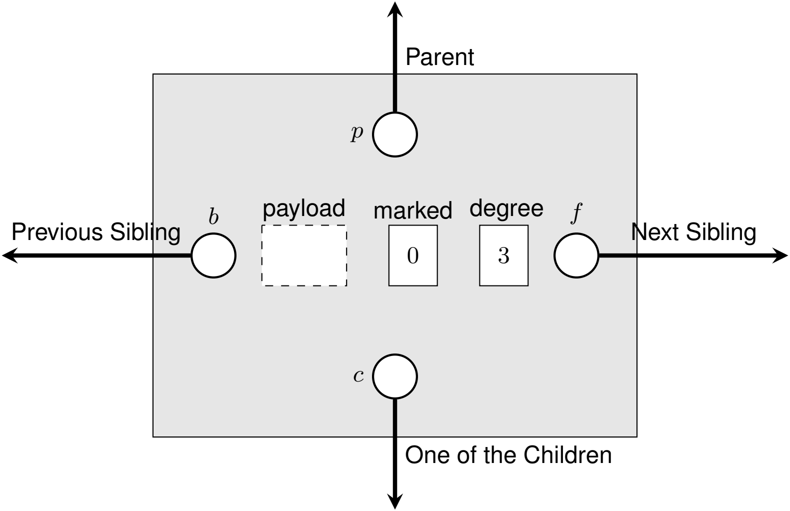
Fibonacci Heap

- Forest of MIN-HEAPs
- Nodes can be marked (roots are always unmarked)
- Tree roots are stored in a circular, doubly-linked list
- Min-Pointer pointing to the smallest element

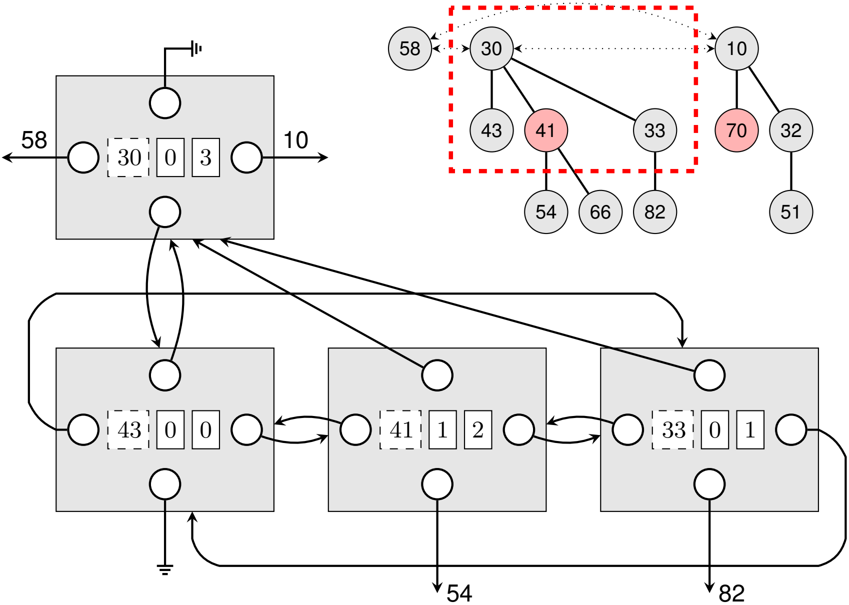


How do we implement a Fibonacci Heap?

A single Node



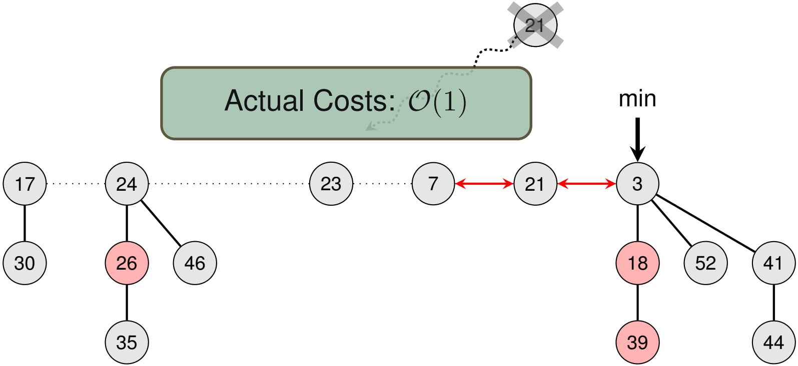
Magnifying a Four-Node Portion



Fibonacci Heap: INSERT

INSERT

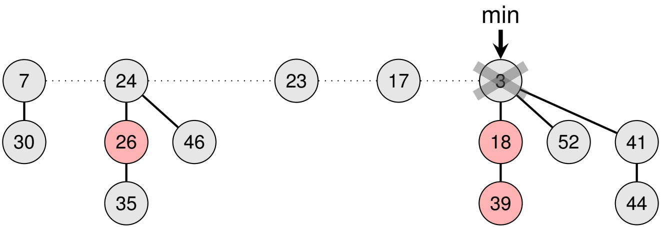
- Create a singleton tree
- Add to root list and update min-pointer (if necessary)



Fibonacci Heap: EXTRACT-MIN

EXTRACT-MIN

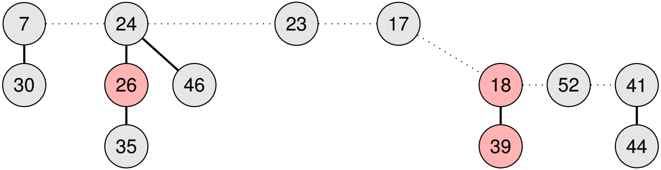
- Delete min



Fibonacci Heap: EXTRACT-MIN

EXTRACT-MIN

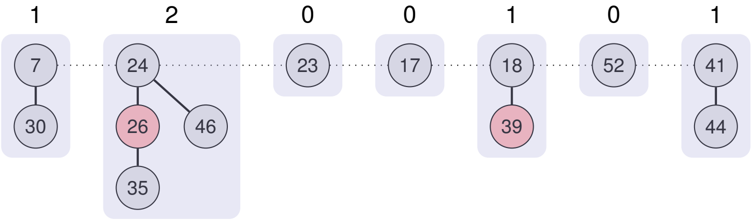
- Delete min ✓
- Add children to root list and unmark them



Fibonacci Heap: Extract-Min

Extract-Min

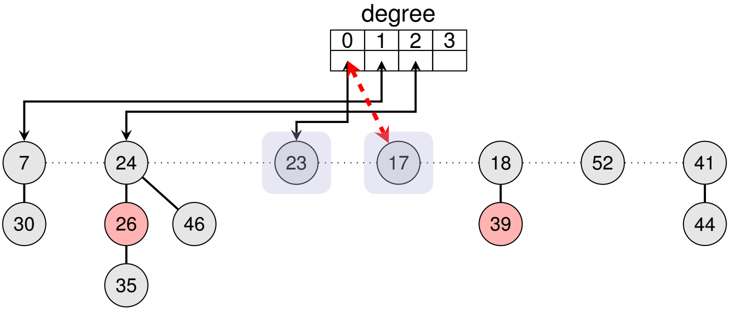
- Delete min ✓
- Add children to root list and unmark them ✓
- Consolidate so that no roots have the same degree (# children)



Fibonacci Heap: EXTRACT-MIN

EXTRACT-MIN

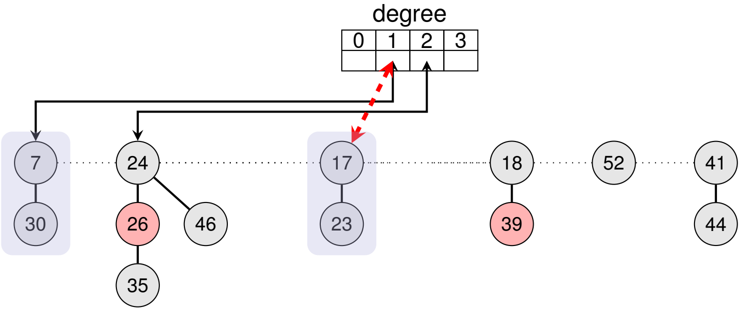
- Delete min ✓
- Add children to root list and unmark them ✓
- Consolidate so that no roots have the same degree (# children)



Fibonacci Heap: EXTRACT-MIN

EXTRACT-MIN

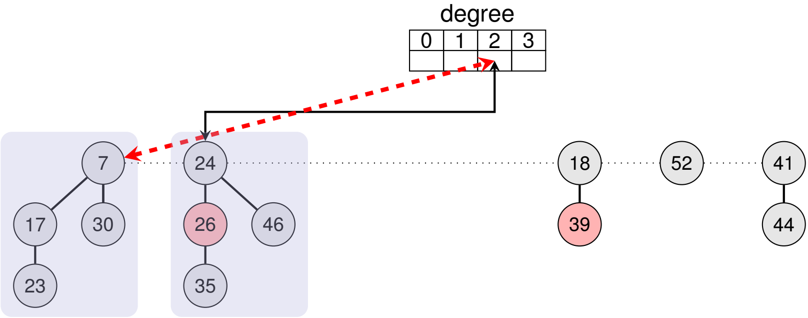
- Delete min ✓
- Add children to root list and unmark them ✓
- Consolidate so that no roots have the same degree (# children)



Fibonacci Heap: EXTRACT-MIN

EXTRACT-MIN

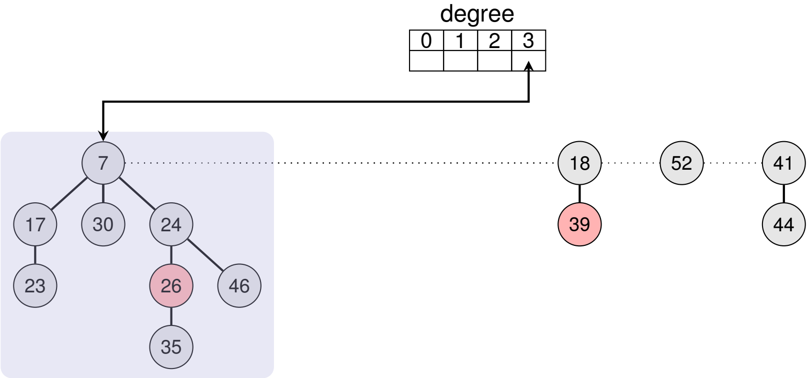
- Delete min ✓
- Add children to root list and unmark them ✓
- Consolidate so that no roots have the same degree (# children)



Fibonacci Heap: EXTRACT-MIN

EXTRACT-MIN

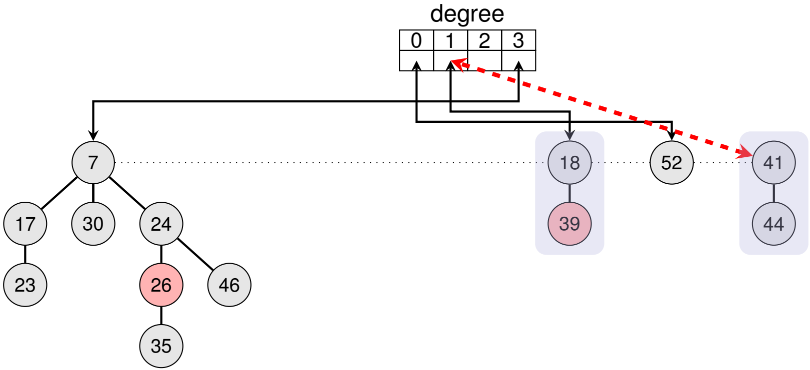
- Delete min ✓
- Add children to root list and unmark them ✓
- Consolidate so that no roots have the same degree (# children)



Fibonacci Heap: EXTRACT-MIN

EXTRACT-MIN

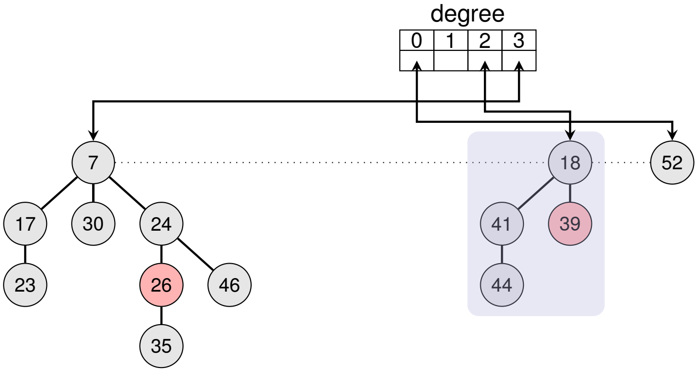
- Delete min ✓
- Add children to root list and unmark them ✓
- Consolidate so that no roots have the same degree (# children)



Fibonacci Heap: EXTRACT-MIN

EXTRACT-MIN

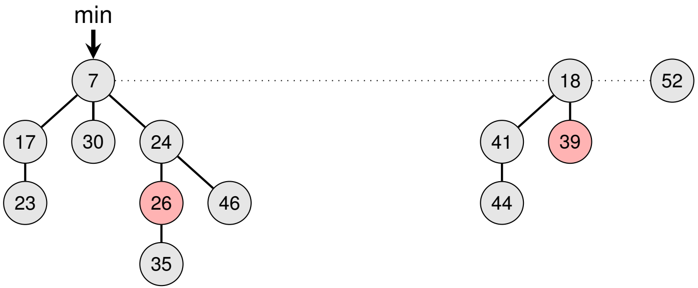
- Delete min ✓
- Add children to root list and unmark them ✓
- Consolidate so that no roots have the same degree (# children) ✓



Fibonacci Heap: EXTRACT-MIN

EXTRACT-MIN

- Delete min ✓
- Add children to root list and unmark them ✓
- Consolidate so that no roots have the same degree (# children) ✓
- Update minimum



Fibonacci Heap: EXTRACT-MIN

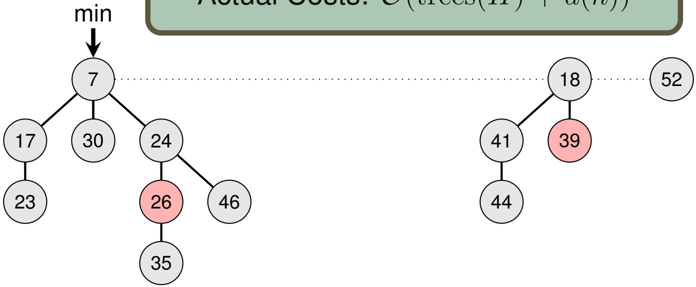
EXTRACT-MIN

- Delete min ✓
- Add children to root list and unmark them ✓
- Consolidate so that no roots have the same degree (# children) ✓
- Update minimum ✓

Every root becomes

$d(n)$ is the maximum degree of a root in any Fibonacci heap of size n

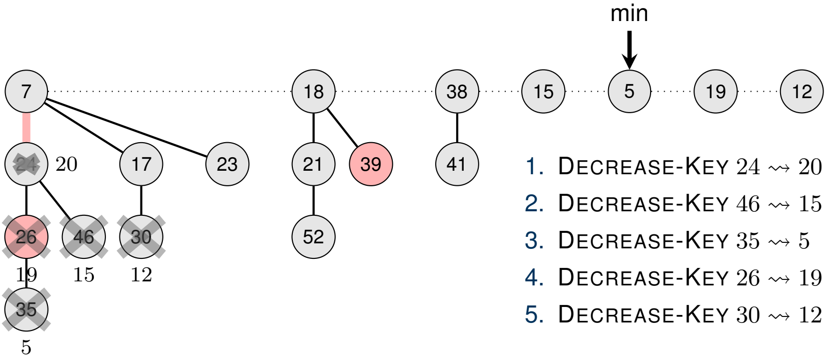
Actual Costs: $\mathcal{O}(\text{trees}(H) + d(n))$



Fibonacci Heap: DECREASE-KEY (First Attempt)

DECREASE-KEY of node x

- Decrease the key of x (given by a pointer)
- Check if heap-order is violated
  - If not, then done.
  - Otherwise, cut tree rooted at x and add it to root list (update min).

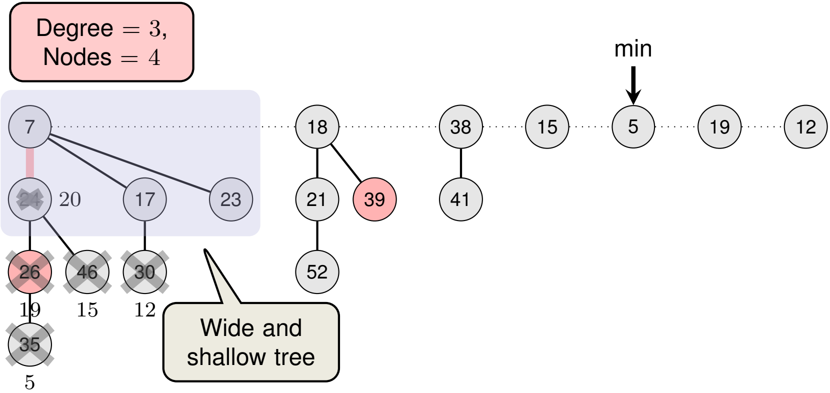


1. DECREASE-KEY $24 \rightsquigarrow 20$
2. DECREASE-KEY $46 \rightsquigarrow 15$
3. DECREASE-KEY $35 \rightsquigarrow 5$
4. DECREASE-KEY $26 \rightsquigarrow 19$
5. DECREASE-KEY $30 \rightsquigarrow 12$

Fibonacci Heap: DECREASE-KEY (First Attempt)

DECREASE-KEY of node x

- Decrease the key of x (given by a pointer)
- Check if heap-order is violated
  - If not, then done.
  - Otherwise, cut tree rooted at x and add it to root list (update min).

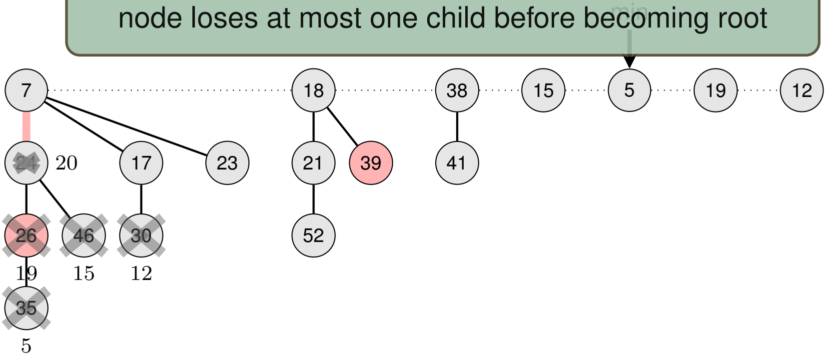


Fibonacci Heap: DECREASE-KEY (First Attempt)

DECREASE-KEY of node x

- Decrease the key of x (given by a pointer)
- Check if heap-order is violated
  - If not, then done.
  - Otherwise, cut tree rooted at x and add it to root list (update min).

Peculiar Constraint: Make sure that each non-root node loses at most one child before becoming root



Fibonacci Heap: DECREASE-KEY

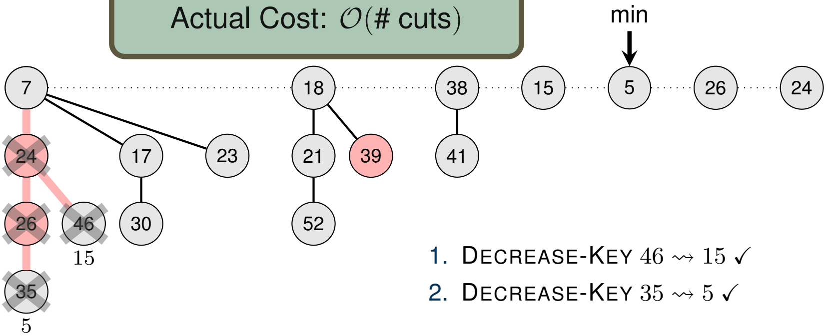
DECREASE-KEY of node x

- Decrease the key of x (given by a pointer)
- (Here we consider only cases where heap-order is violated)

$\Rightarrow$ Cut tree rooted at x, unmark x, and add it to to root list and:

- Check if parent node is marked
  - If unmarked, mark it (unless it is a root)
  - If marked, unmark and add it to root list and recurse (Cascading Cut)

Actual Cost: $\mathcal{O}(\text{\# cuts})$

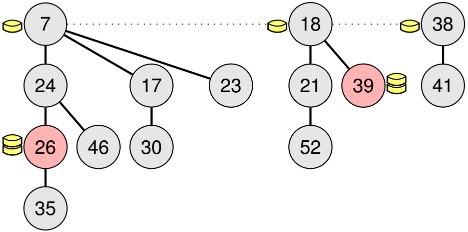


1. DECREASE-KEY $46 \rightsquigarrow 15$ ✓
2. DECREASE-KEY $35 \rightsquigarrow 5$ ✓

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 $\mathcal{O}$(# 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