Learn DAA: From B K Sharma

TCS-503: Design and Analysis of Algorithms

Unit II

Advanced Data Structures: Fibonacci Heaps

Learn DAA: From B K Sharma

Unit II

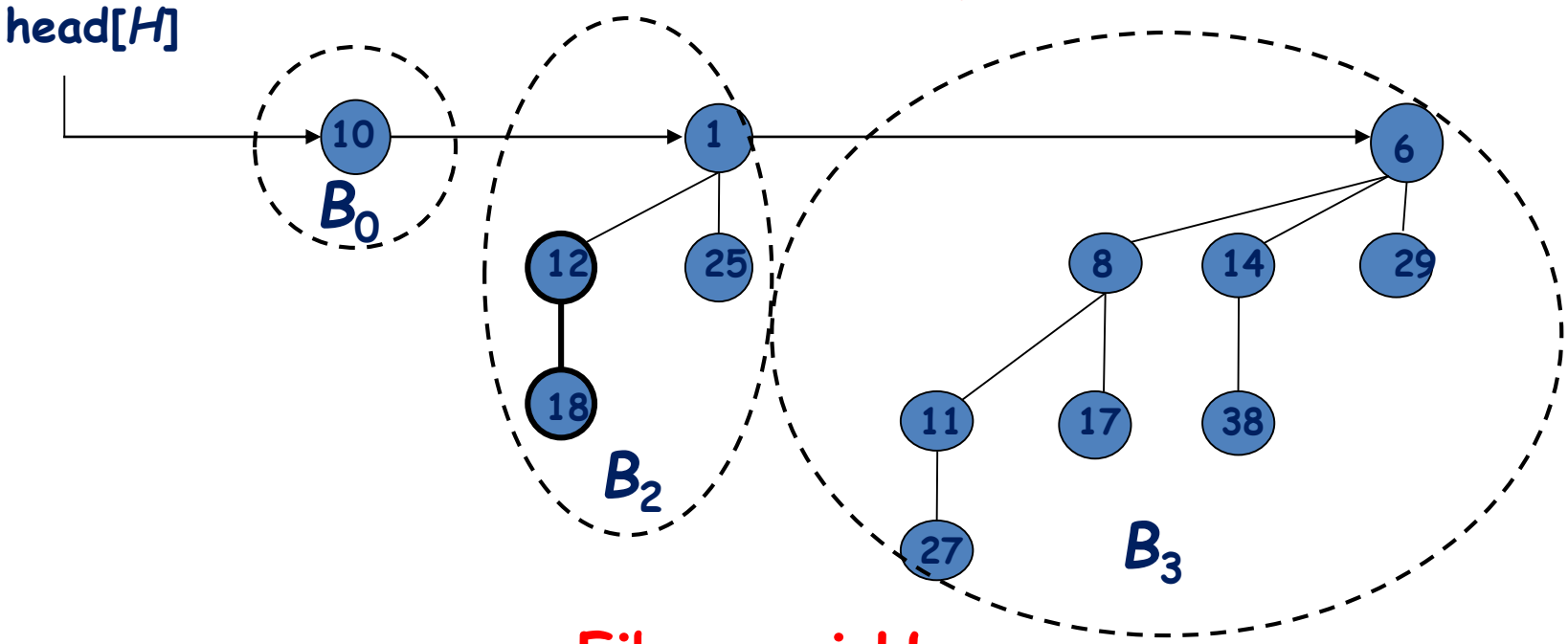
- Advanced Data Structures:
  - Red-Black Trees
  - Augmenting Data Structure
  - B-Trees
  - Binomial Heaps
  - Fibonacci Heaps
  - Data Structure for Disjoint Sets

Fibonacci Heaps

Fibonacci heaps are linked lists of heap-ordered trees(min-heap) with the following characteristics:

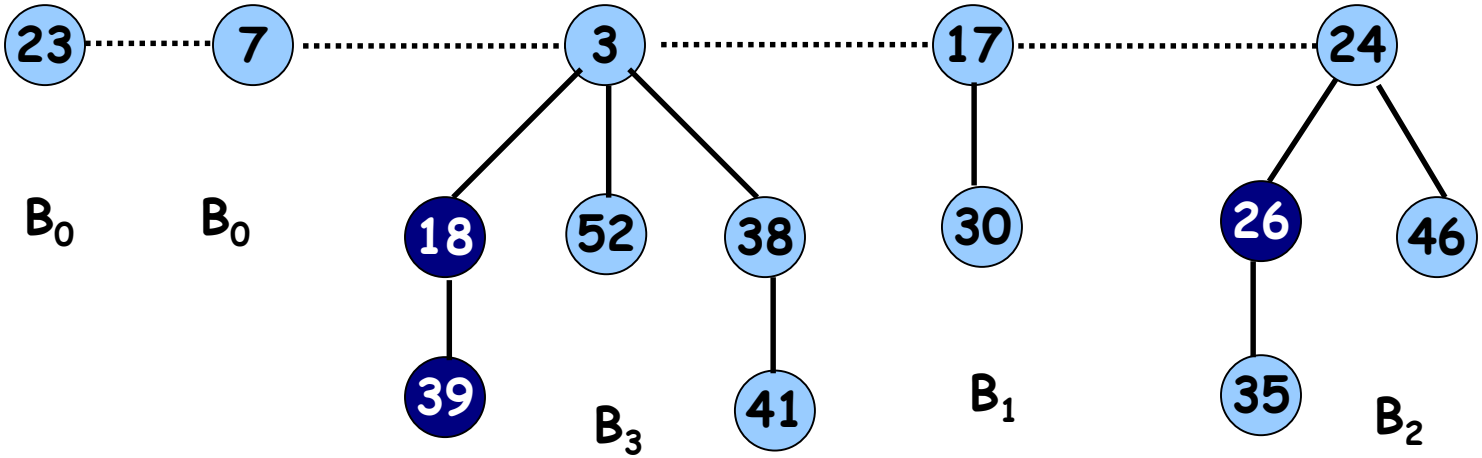
1. The trees are not necessarily binomial.
2. The root degrees are not unique.
3. Siblings are bi-directionally (circularly-doubly) linked.
4. There is a pointer min[H] to the root with the minimum key.
5. A special attribute n[H] maintains the total number of nodes.
6. Each node has an additional attribute mark, indicating whether it has lost a child since the last time it was made the child of another node.

Binomial Heap

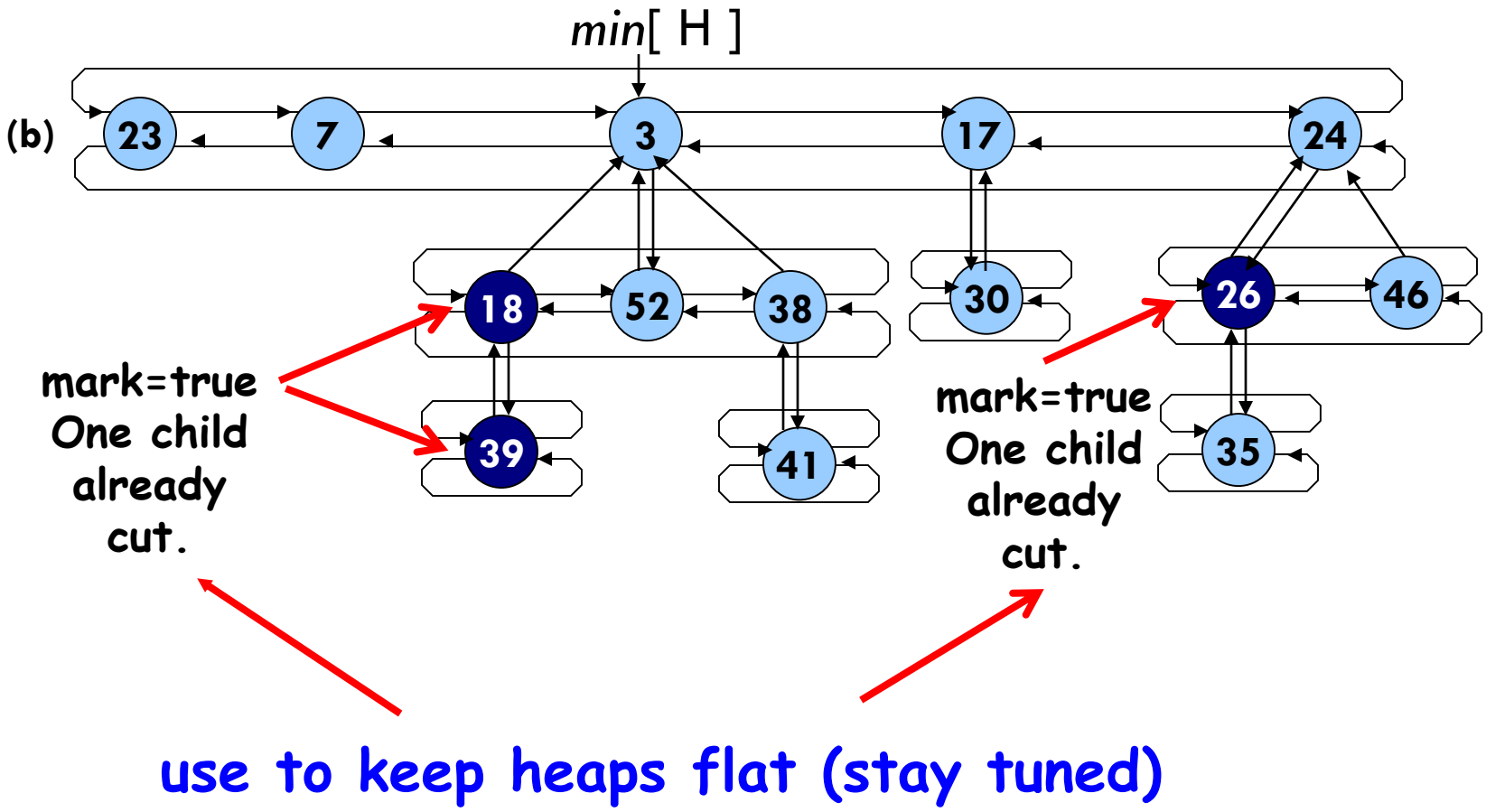


Fibonacci Heap

min[H]



Fibonacci Heap: Representation in Memory



use to keep heaps flat (stay tuned)

Learn DAA: From B K Sharma

Comparison of Binomial And Fibonacci Heaps

With Respect To Complexity

Binomial heaps :

- INSERT
- DELETE
- UNION
- MINIMUM
- EXTRACT-MIN
- DECREASE-KEY
- ...

→ Worst-Case Running Time is O(lg n)

Fibonacci heaps :

- INSERT
- UNION
- MINIMUM
- DECREASE-KEY

→ Worst-Case Running Time is O(1)

- EXTRACT-MIN
- DELETE

→ Worst-Case Running Time is O(lg n)

Any sequence of n INSERTs, m UNIONs, f MINIMUMs, and d DECREASE-KEY takes constant time, O(1)-Amortized Analysis, not worst-case per operation.

Learn DAA: From B K Sharma

Comparison of Binomial And Fibonacci Heaps

With Respect To Structure

Like a binomial heap, a *Fibonacci heap is a collection of min-heap-ordered trees.*

The trees in a Fibonacci heap are not constrained to be binomial trees, however.

Unlike trees within binomial heaps, which are ordered, trees within Fibonacci heaps are rooted but unordered.

The Advantage of Fibonacci Heaps

Fibonacci heaps have the advantage that operations that do not involve deleting an element run in *O(1) amortized time.*

Uses of Fibonacci Heaps

Fibonacci heaps are especially desirable when the number of EXTRACT-MIN and DELETE operations is small relative to the number of other operations performed.

Algorithms for graph problems such as computing minimum spanning trees (Chapter 23) and finding single-source shortest paths (Chapter 24) make essential use of Fibonacci heaps.

Learn DAA: From B K Sharma

Unit IV

- Graph Algorithms:
  - Elementary Graphs algorithms
  - Minimum Spanning Trees
  - Single-Source Shortest Paths
  - All-Pairs Shortest Paths
  - Maximum Flow and
  - Traveling Salesman Problem

Fibonacci Heaps: Operation

DECREASE-KEY (H, x, k)

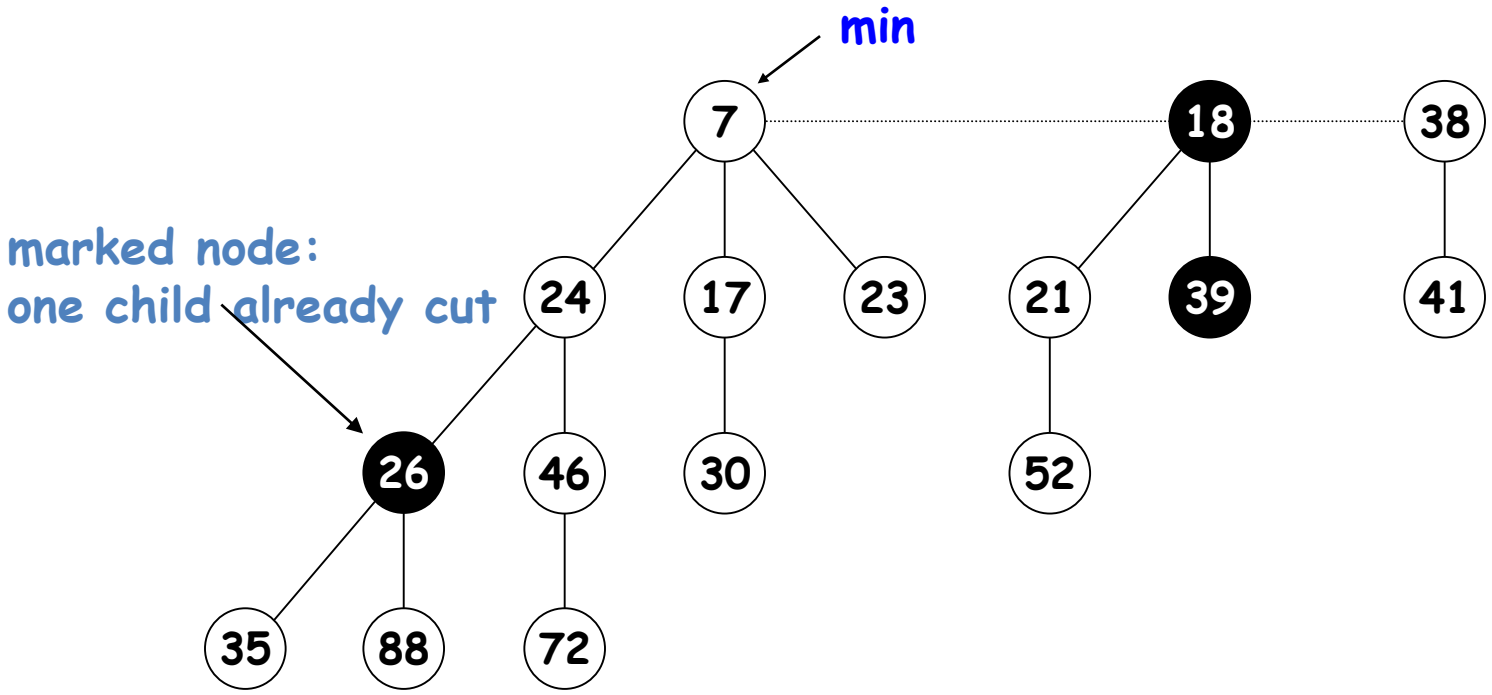
Intuition (Insight) for deceasing the key of node x:

If heap-order is not violated, just decrease the key of x.

Otherwise, cut tree rooted at x and meld (join) into root list.

To keep trees flat: as soon as a node has its second child cut, cut it off and meld into root list (and unmark it).

Fibonacci Heaps: DECREASE-KEY Operation

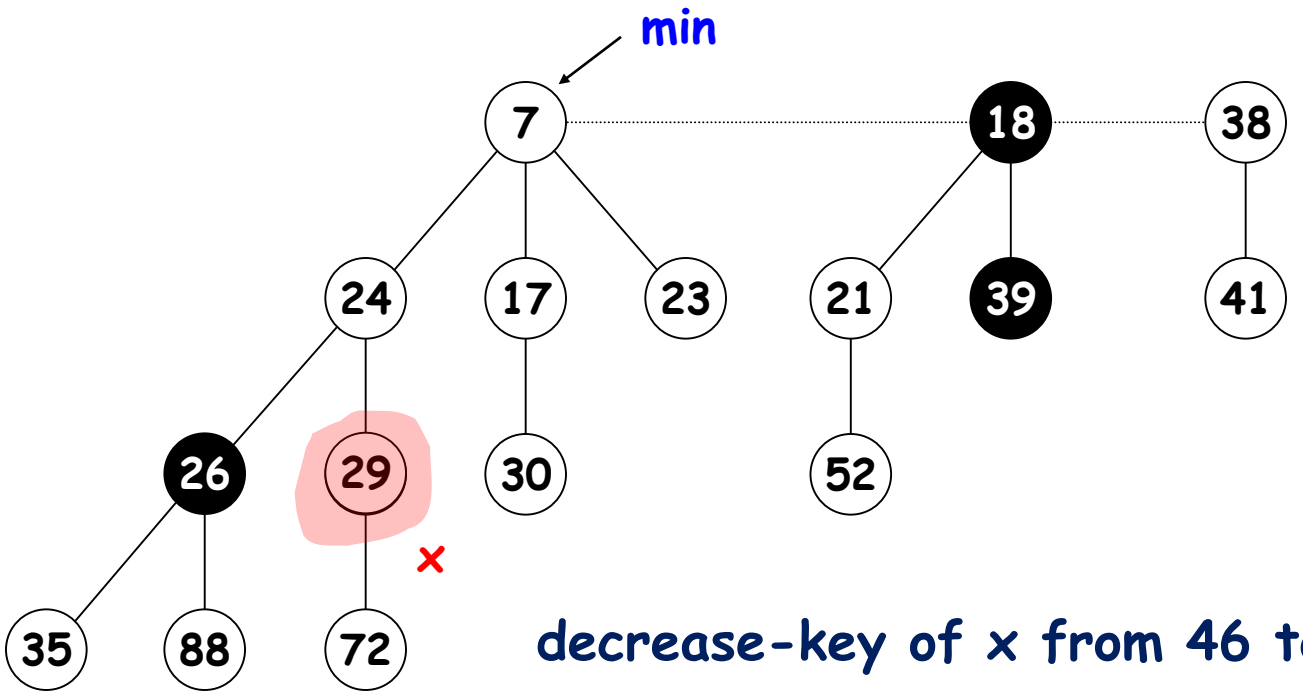


Learn DAA: From B K Sharma

Fibonacci Heaps: DECREASE-KEY Operation

Case 1: [heap order not violated]

Step 1:Decrease key of x.



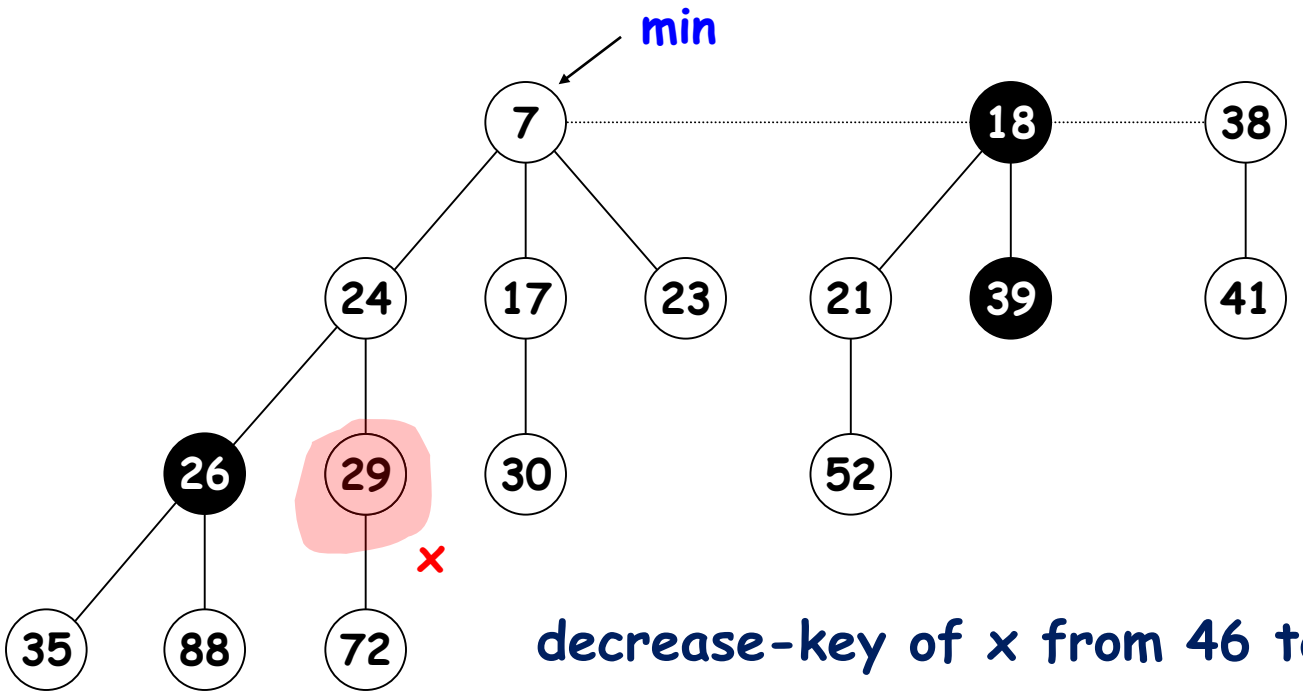
Learn DAA: From B K Sharma

Fibonacci Heaps: DECREASE-KEY Operation

Case 1: [heap order not violated]

Step 1:Decrease key of x.

Step 2:Change heap min pointer (if necessary).



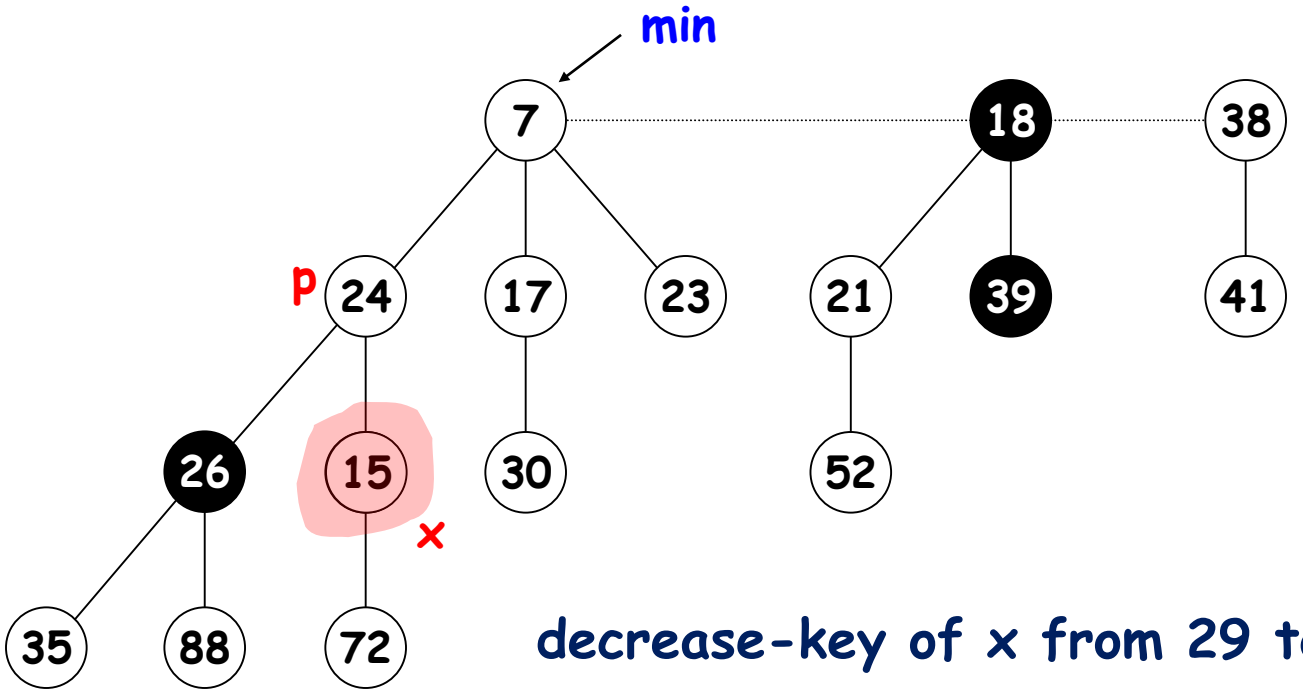
decrease-key of x from 46 to 29

Learn DAA: From B K Sharma

Fibonacci Heaps: DECREASE-KEY Operation

Case 2a: [heap order violated] & [if parent p of x is unmarked]

Step 1: Decrease key of x.

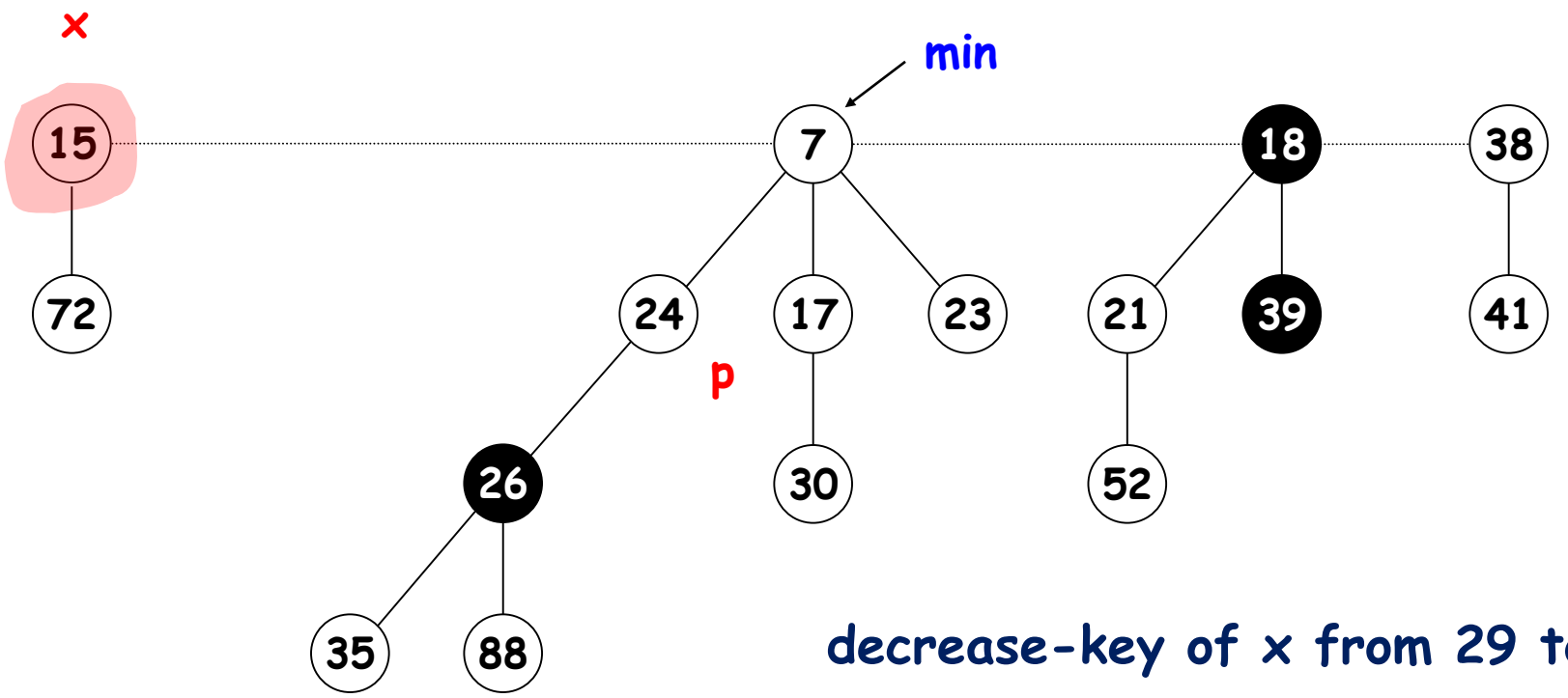


Learn DAA: From B K Sharma

Fibonacci Heaps: DECREASE-KEY Operation

Case 2a: [heap order violated] & [if parent p of x is unmarked]

Step 2: Cut tree rooted at x, meld into root list, and unmark.



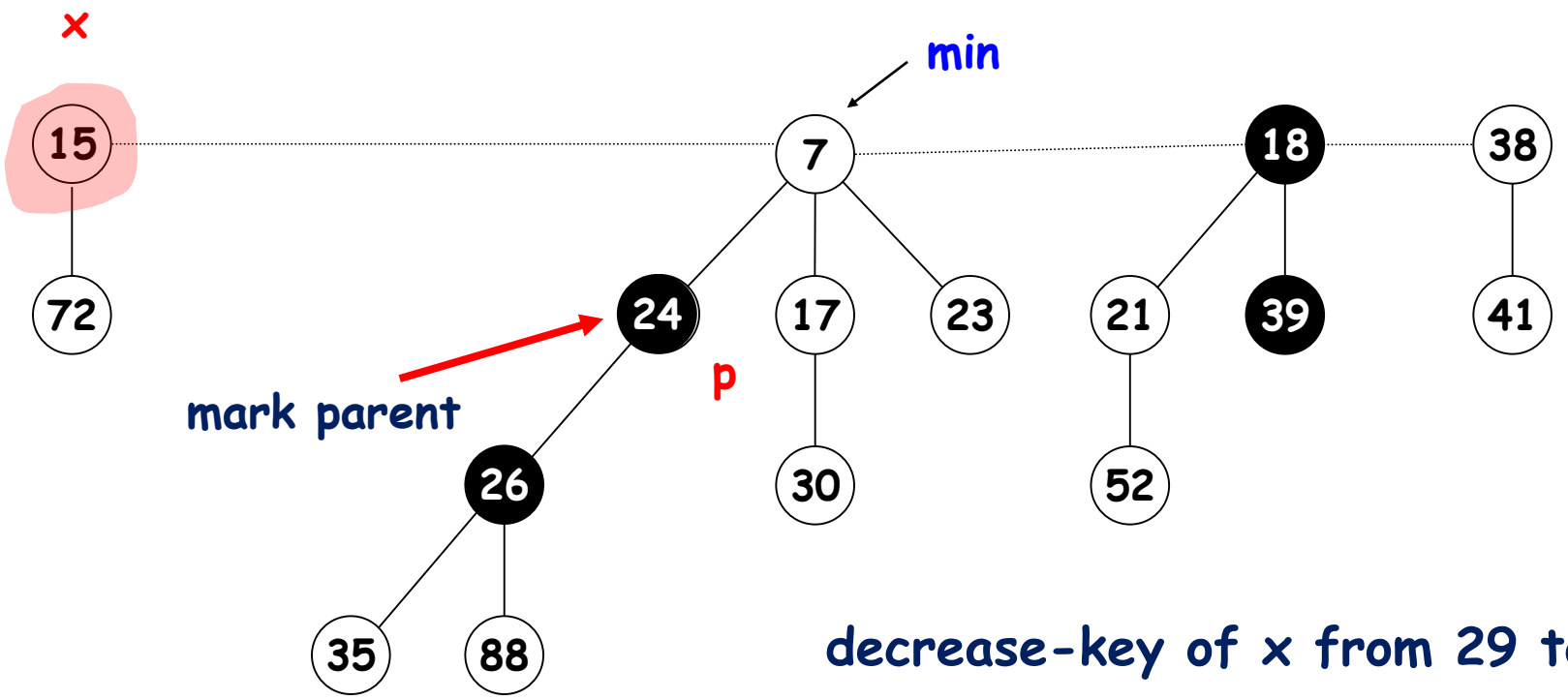
decrease-key of x from 29 to 15

Learn DAA: From B K Sharma

Fibonacci Heaps: DECREASE-KEY Operation

Case 2a: [heap order violated] & [if parent p of x is unmarked]

Step 3: If parent p of x is unmarked (hasn't yet lost a child), mark it;

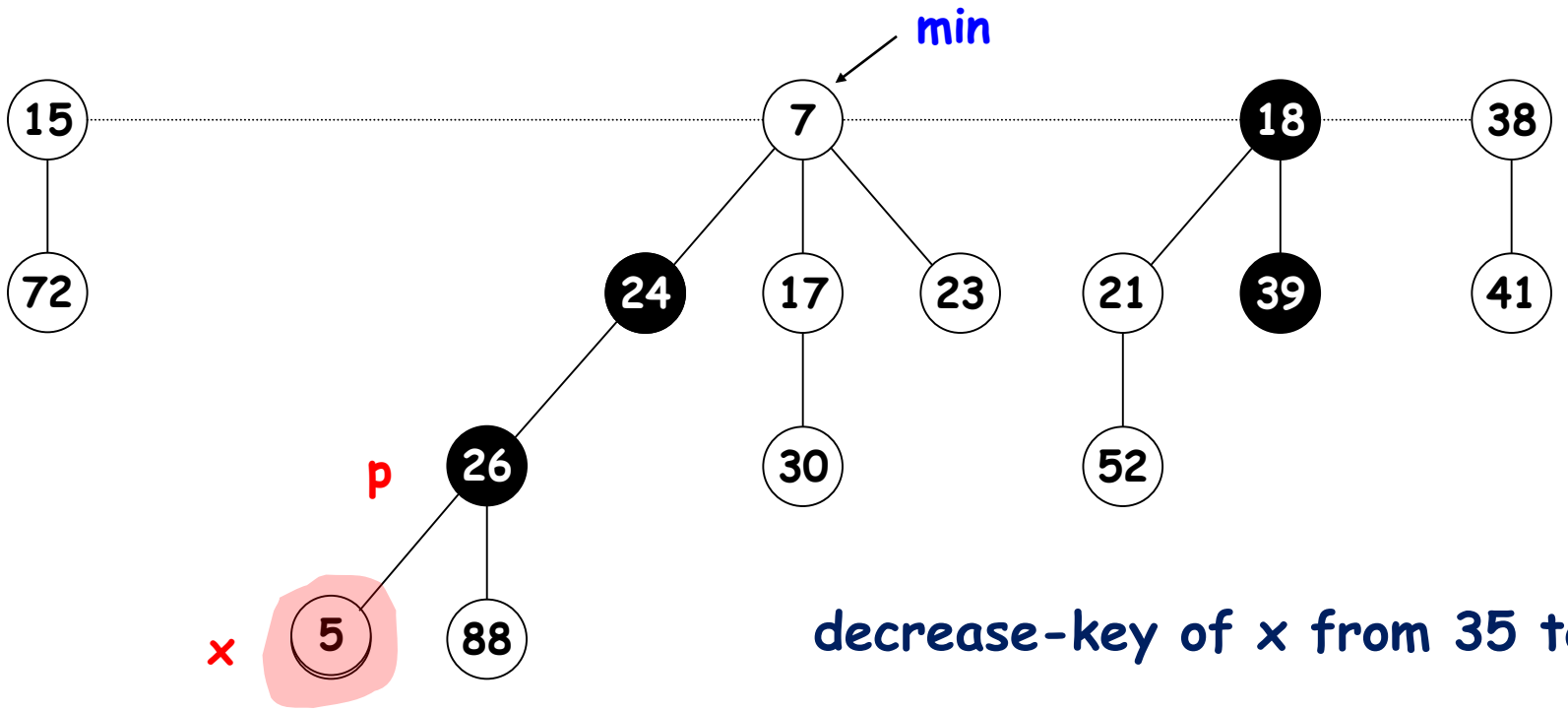


Learn DAA: From B K Sharma

Fibonacci Heaps: DECREASE-KEY Operation

Case 2b: [heap order violated] & [if parent p of x is marked]

Step 1: Decrease key of x.

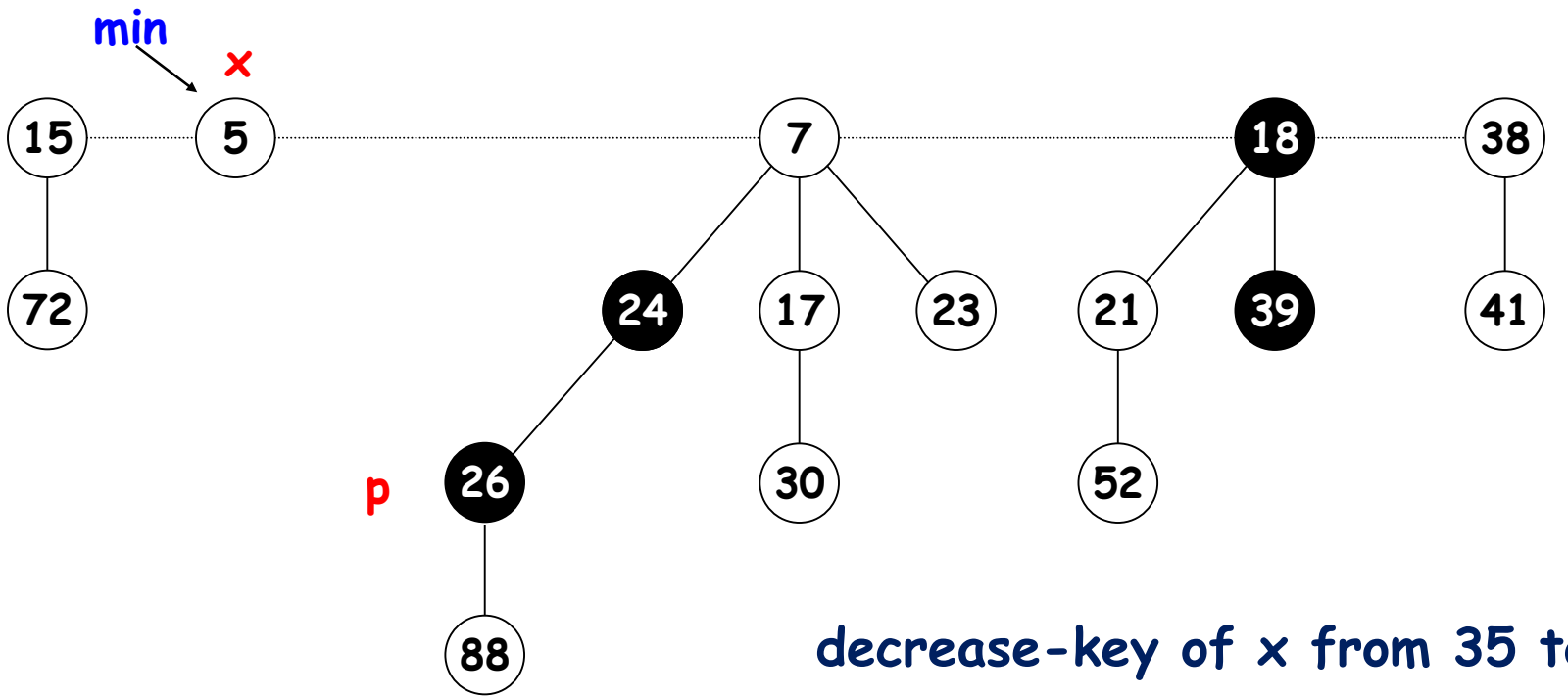


Learn DAA: From B K Sharma

Fibonacci Heaps: DECREASE-KEY Operation

Case 2b: [heap order violated] & [if parent p of x is marked]

Step 2: Cut tree rooted at x, meld into root list, and unmark.

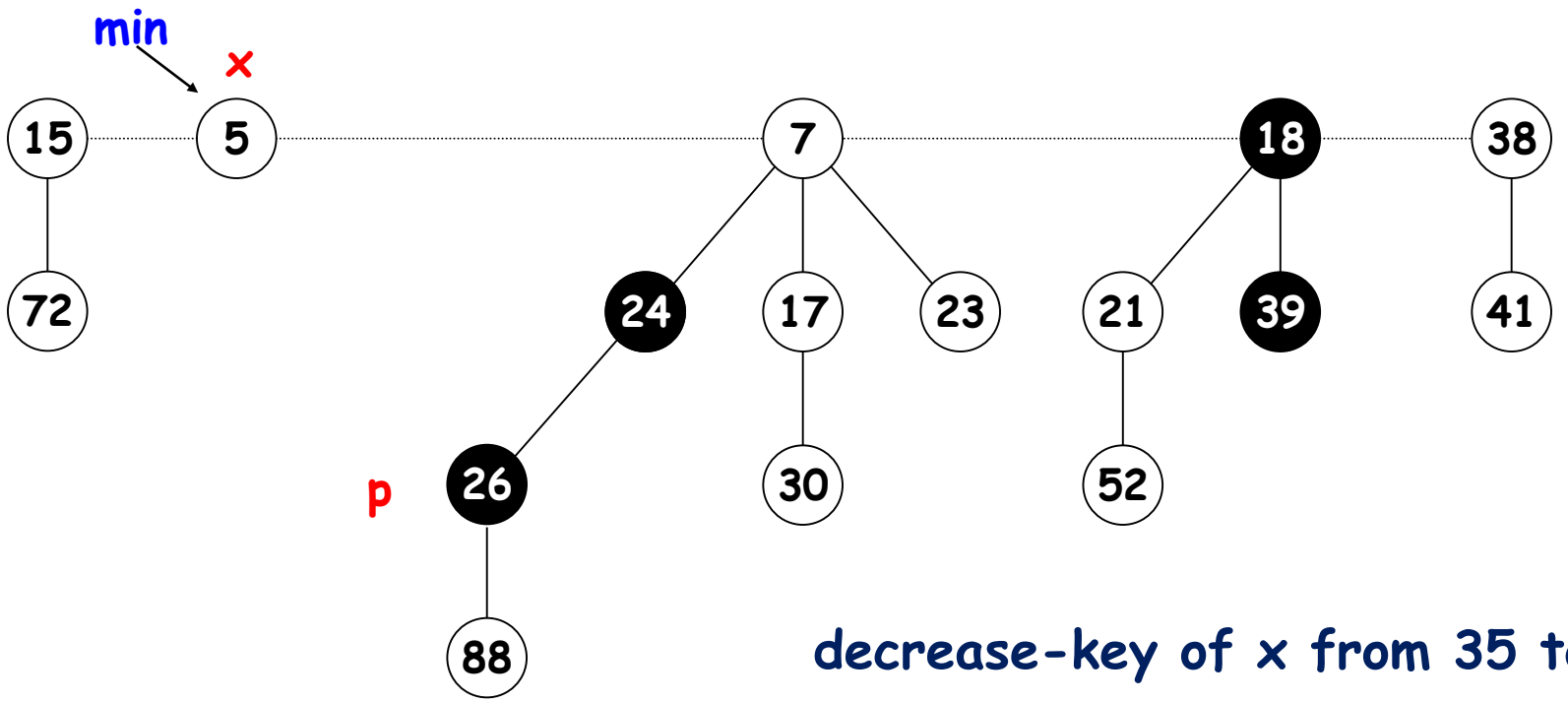


Learn DAA: From B K Sharma

Fibonacci Heaps: DECREASE-KEY Operation

Case 2b: [heap order violated] & [if parent p of x is marked]

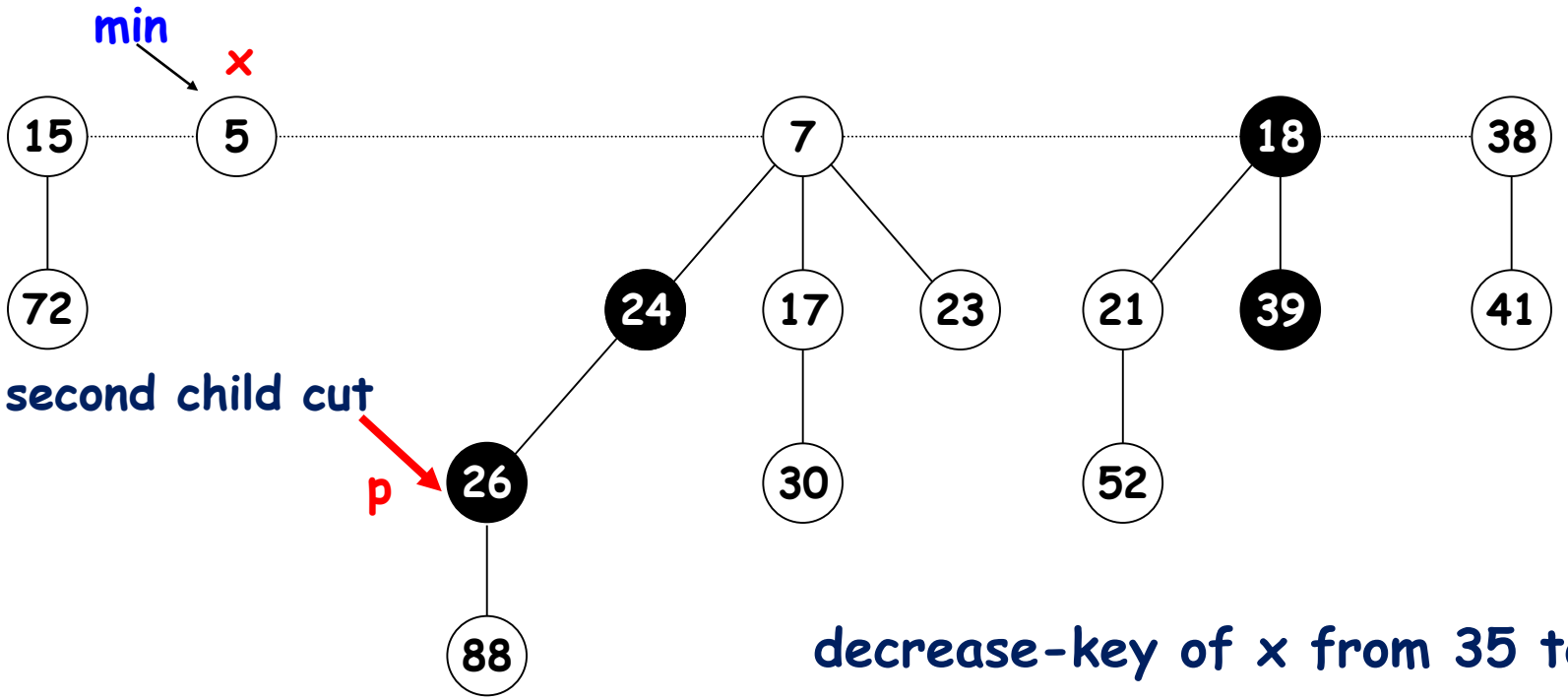
Step 3: If parent p of x is unmarked (hasn't yet lost a child), mark it(parent is already marked)



Fibonacci Heaps: DECREASE-KEY Operation

Case 2b: [heap order violated] & [if parent p of x is marked]

Step 4: Otherwise, cut p, meld into root list, and unmark (and do so recursively for all ancestors that lose a second child).

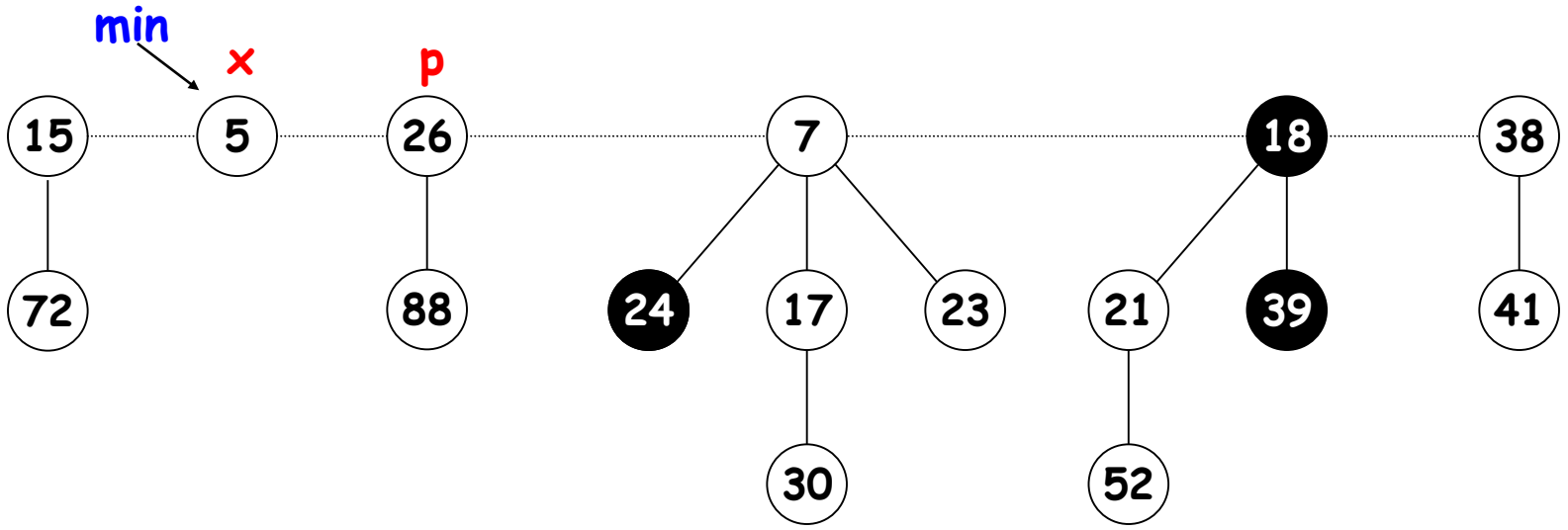


Learn DAA: From B K Sharma

Fibonacci Heaps: DECREASE-KEY Operation

Case 2b: [heap order violated] & [if parent p of x is marked]

Step 4: Otherwise, cut p, meld into root list, and unmark (and do so recursively for all ancestors that lose a second child).



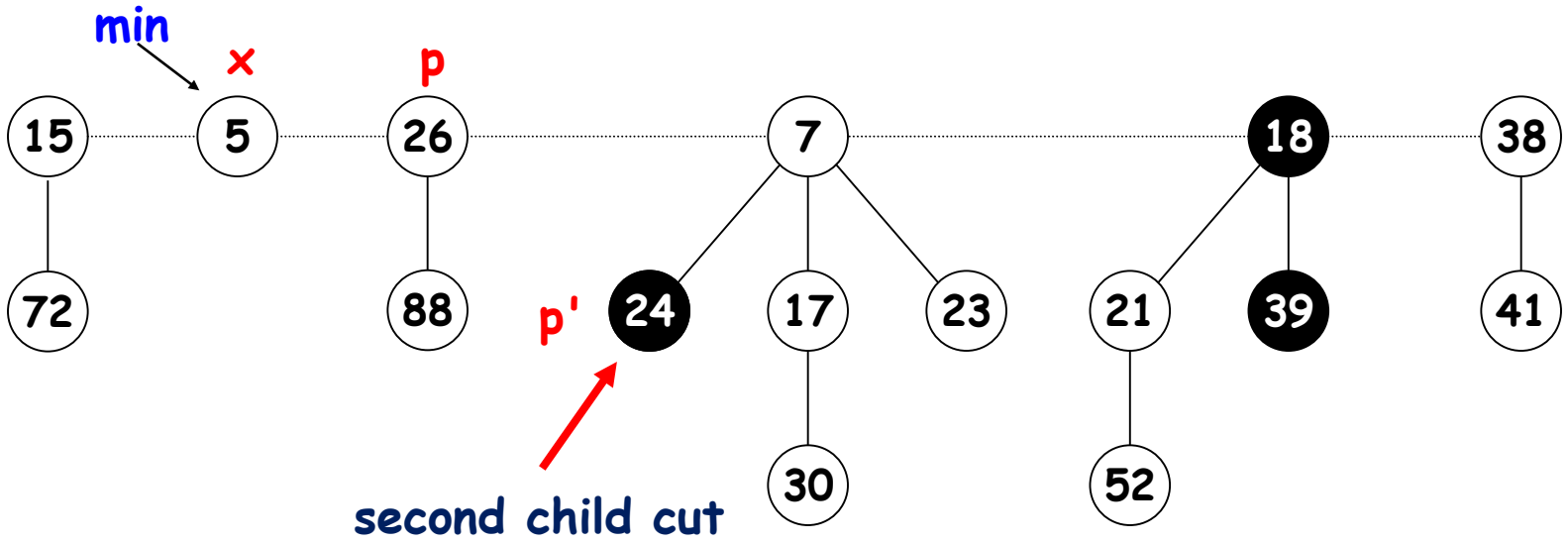
decrease-key of x from 35 to 5

Learn DAA: From B K Sharma

Fibonacci Heaps: DECREASE-KEY Operation

Case 2b: [heap order violated] & [if parent p of x is marked]

Step 4: Otherwise, cut p, meld into root list, and unmark (and do so recursively for all ancestors that lose a second child).



decrease-key of x from 35 to 5

Learn DAA: From B K Sharma

Fibonacci Heaps: DECREASE-KEY Operation

Case 2b: [heap order violated] & [if parent p of x is marked]

Step 4: Otherwise, cut p, meld into root list, and unmark (and do so recursively for all ancestors that lose a second child).