TCS-503: Design and Analysis of Algorithms

Unit II

Advanced Data Structures: Binomial Heaps

Unit II

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

Why should we learn Binomial Heaps?

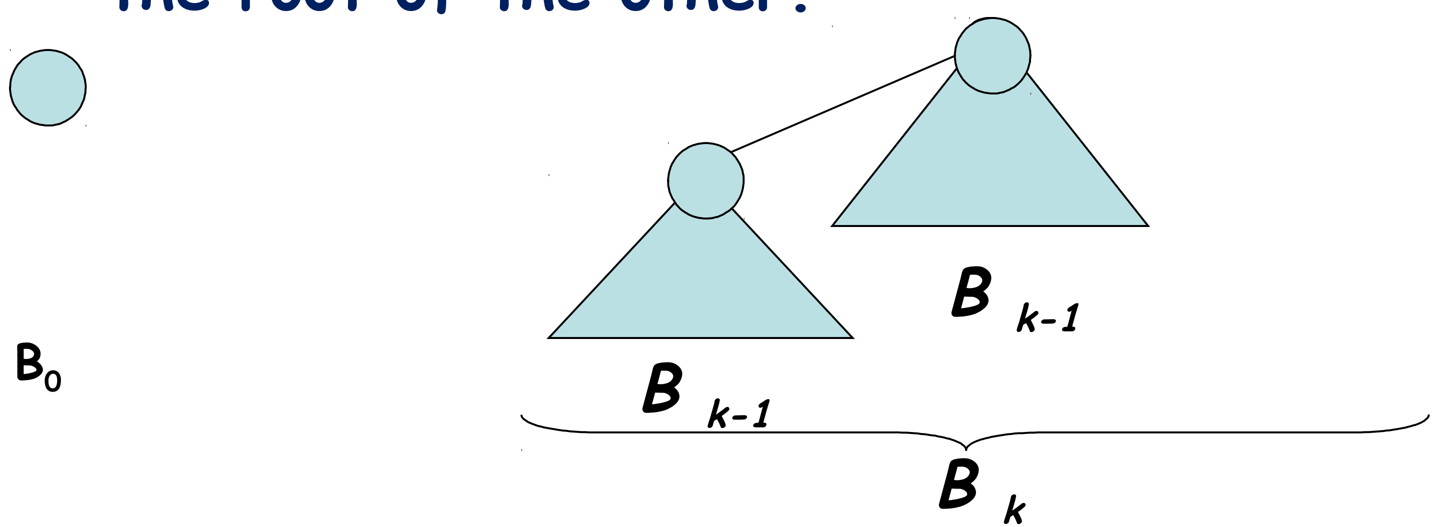
The worst case running time of Union operation that merge two binomial heaps takes only $O(\lg n)$ time where as the worst case running time of Union operation that merge two binary heaps is $\Theta(n)$.

What is Binomial Tree?

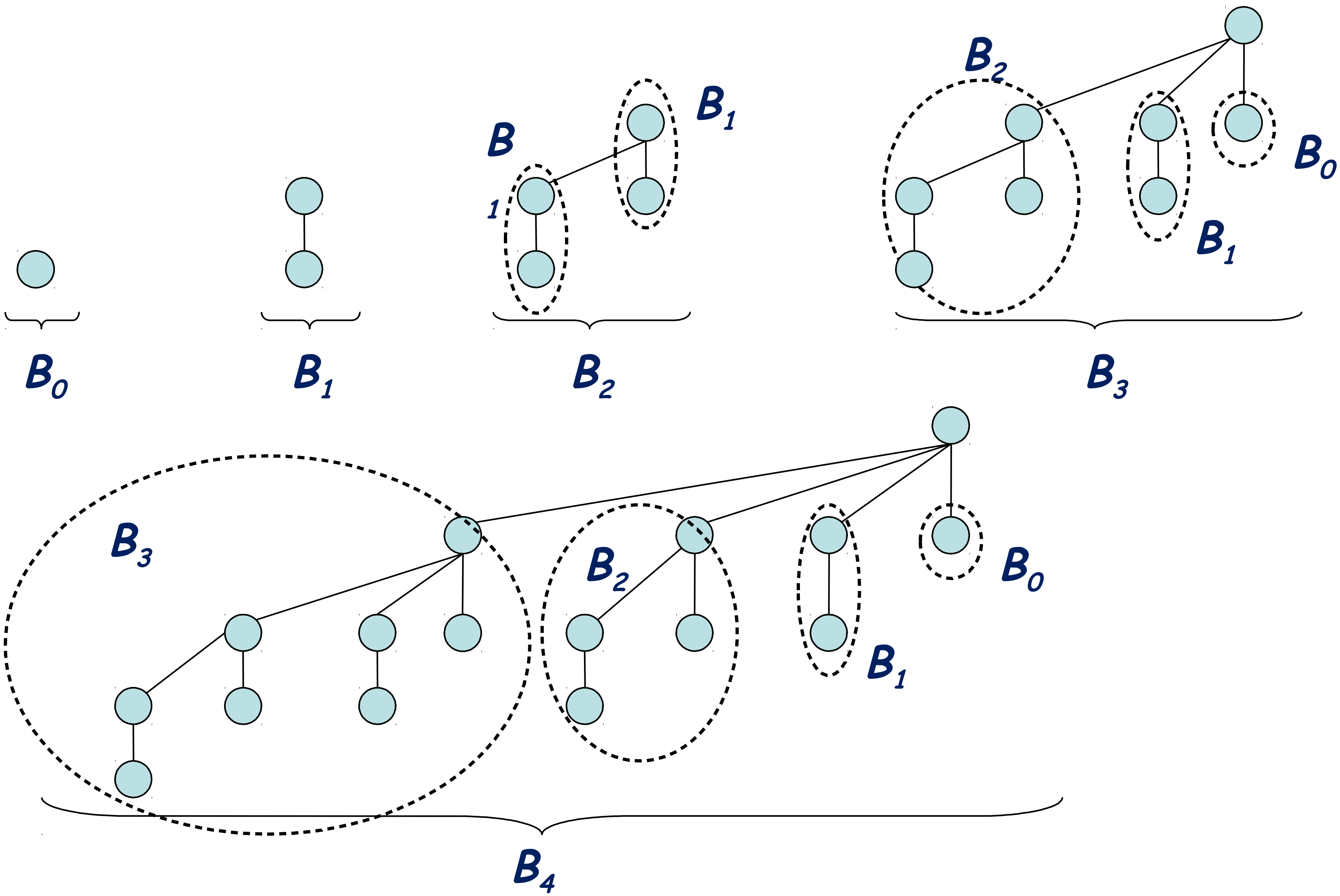
The *binomial tree* B_k *is an ordered tree defined recursively*

1) The binomial tree B_0 *consists of a single node.*
2) The binomial tree B_k *consists of two binomial trees* B_{k-1} *that are* *linked together such that* :

   the root of one is the leftmost child of the root of the other.



What is Binomial Tree?



Learn DAA: From B K Sharma

What are Properties of Binomial Tree?

For the binomial tree B_k,

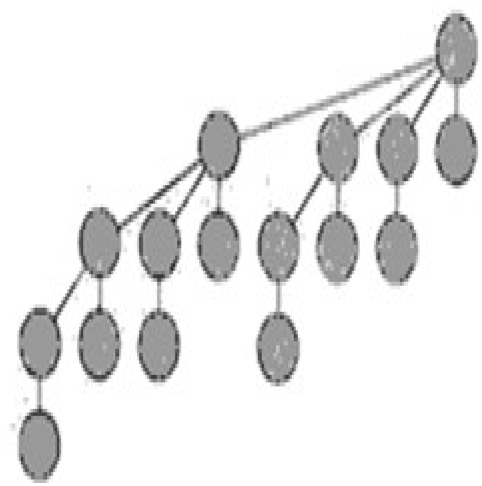
1. The number of nodes of B_k is 2^k.

2. The height of the tree is k

3. B_k has exactly $\binom{k}{i}$ nodes at depth i *for* $i = 0, 1, \ldots, k$, *and*

4. The root degree of B_k is greater than the degree of every other node in B_k.

What are Properties of Binomial Tree?



B_4

i	n	
0	$\binom{4}{0}$	= 1
1	$\binom{4}{1}$	= 4
2	$\binom{4}{2}$	= 6
3	$\binom{4}{3}$	= 4
4	$\binom{4}{4}$	= 1

$$\binom{k}{i} = \frac{k!}{i!\ (k-i)!}$$

Learn DAA: From B K Sharma

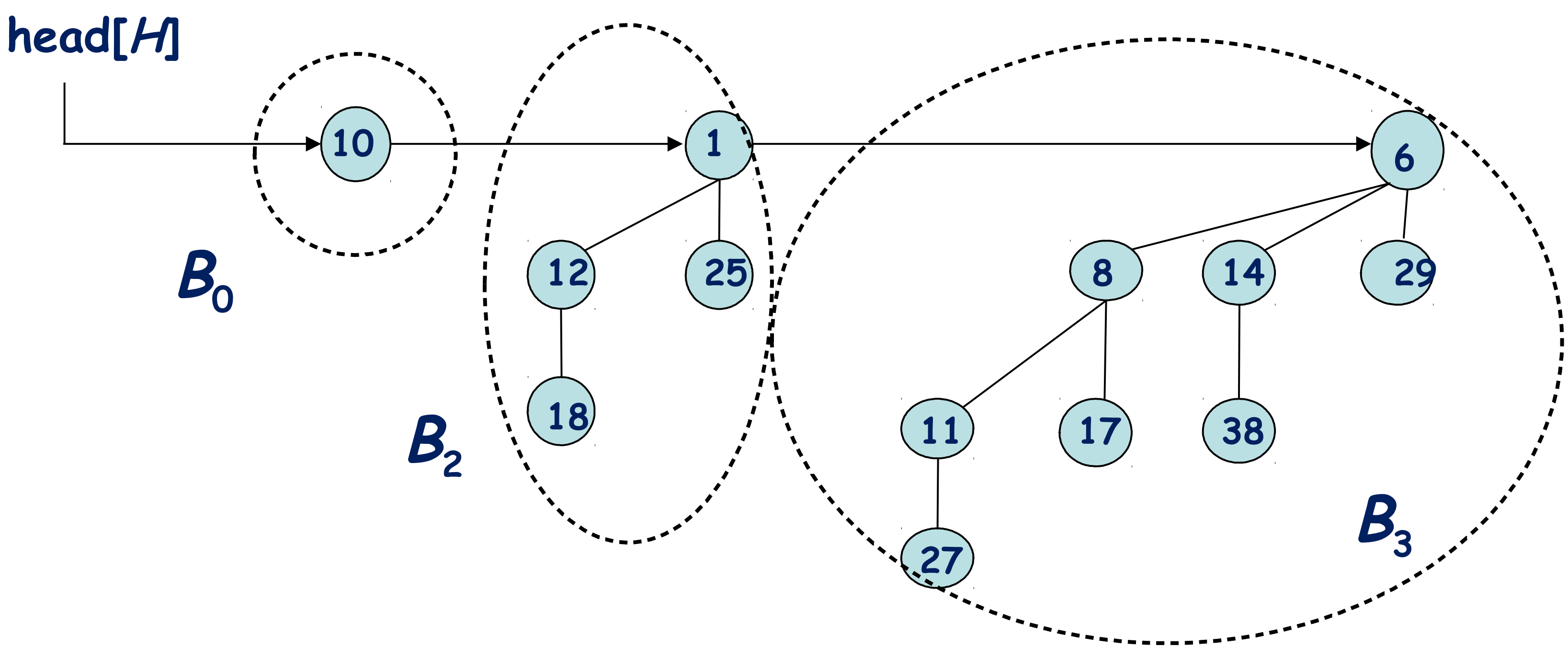
Binomial Heap (H)

is a set of Binomial Trees

that satisfies the following Properties:-

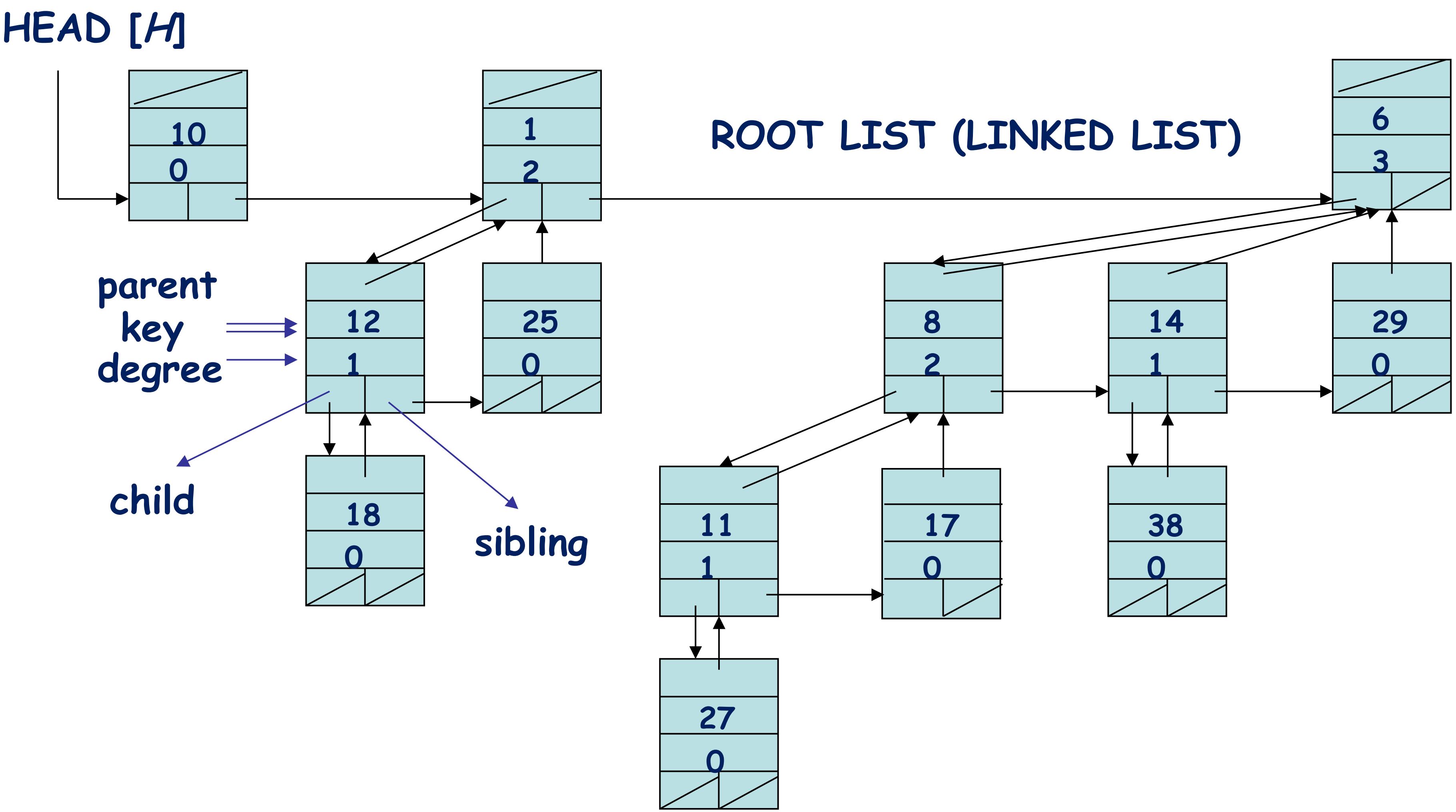
1. Each Binomial tree in H obeys the min- heap property

2. For any non-negative integer k, there is at most one binomial tree in H whose root has degree k



Learn DAA: From B K Sharma

Representation of Binomial Heap



Uniting Two Binomial Heaps

Uses Two supporting Procedures

BINOMIAL-LINK (y, z)

and

BINOMIAL-HEAP-MERGE (H1, H2)

BINOMIAL-LINK (y, z)

Links the BINOMIAL tree B_{k-1} rooted at node *y* to the BINOMIAL tree B_{k-1} rooted at node z it makes z the parent of *y* i.e. Node *z* becomes the root of a B_k tree.

BINOMIAL-LINK (*y*,*z*)

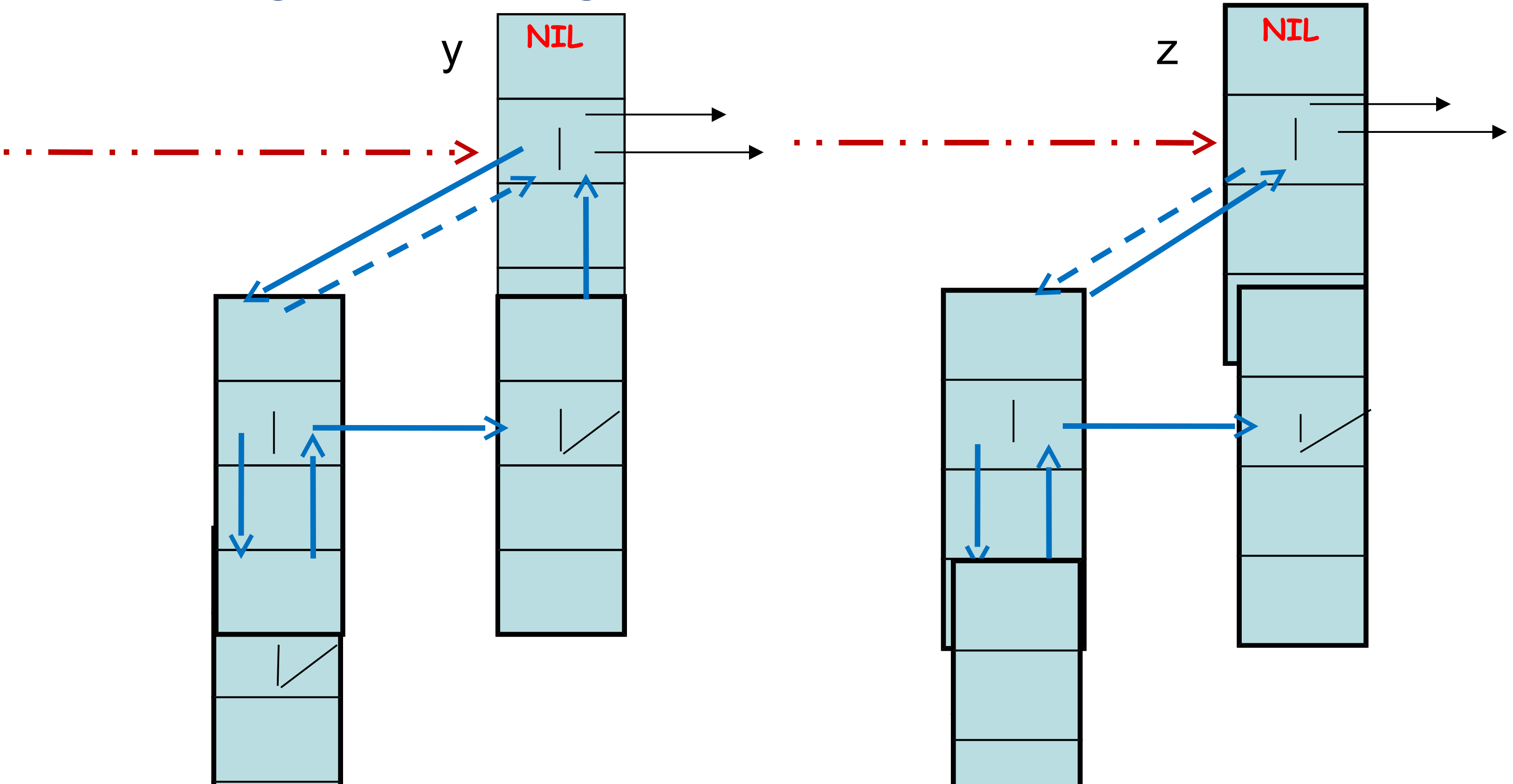
1. p [*y*] ← *z*
2. sibling [*y*] ← child [*z*]
3. child [*z*] ← *y*
4. degree [*z*] ← degree [*z*] + 1

Learn DAA: From B K Sharma

BINOMIAL-LINK (y, z)

BINOMIAL-LINK (y,z)

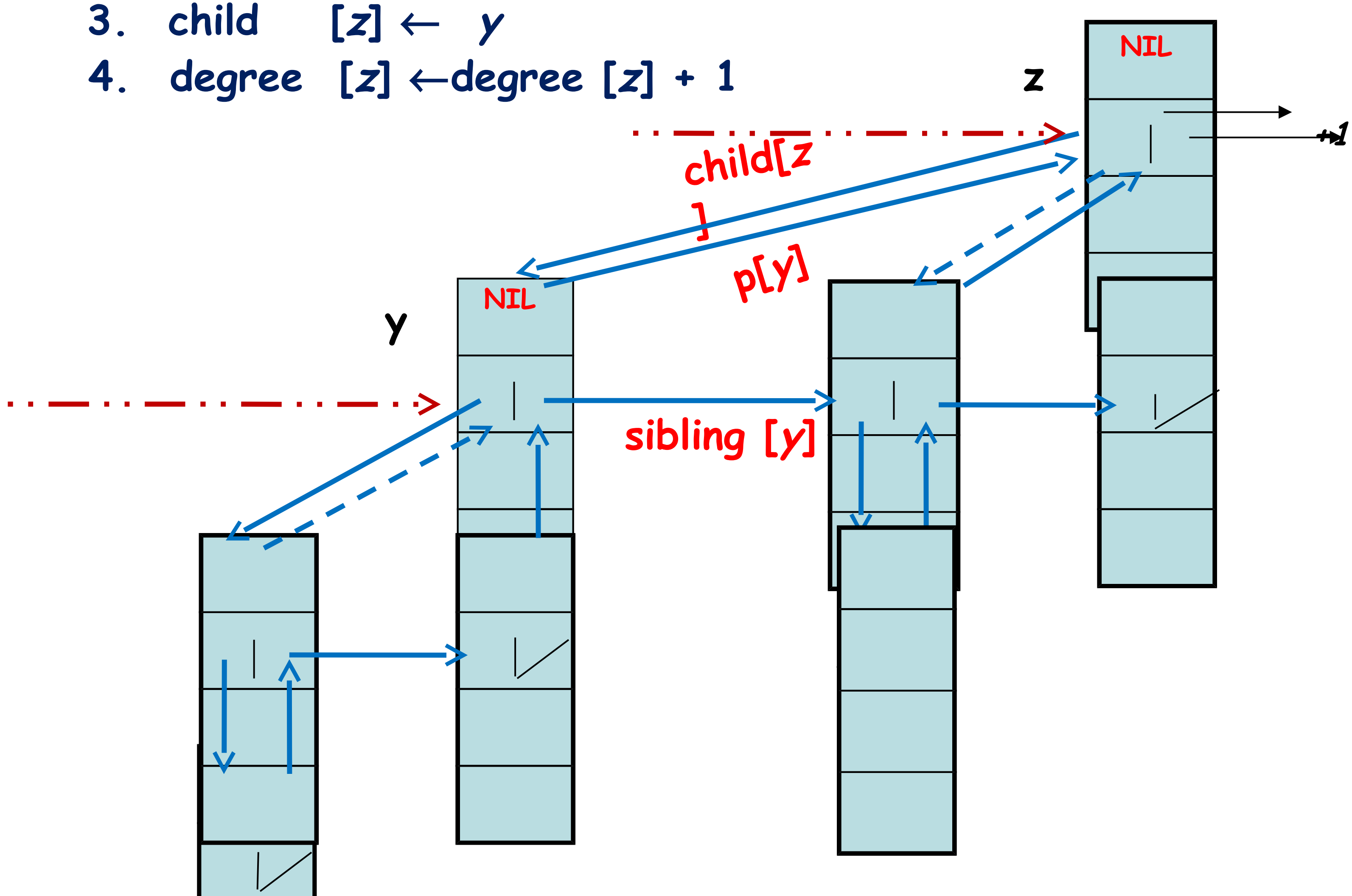
1. p [y] ← z
2. sibling [y] ← child [z]
3. child [z] ← y
4. degree [z] ← degree [z] + 1



BINOMIAL-LINK (y, z)

BINOMIAL-LINK (*y*,*z*)

1. p [*y*] ← *z*
2. sibling [*y*] ← child [*z*]
3. child [*z*] ← *y*
4. degree [*z*] ← degree [*z*] + 1

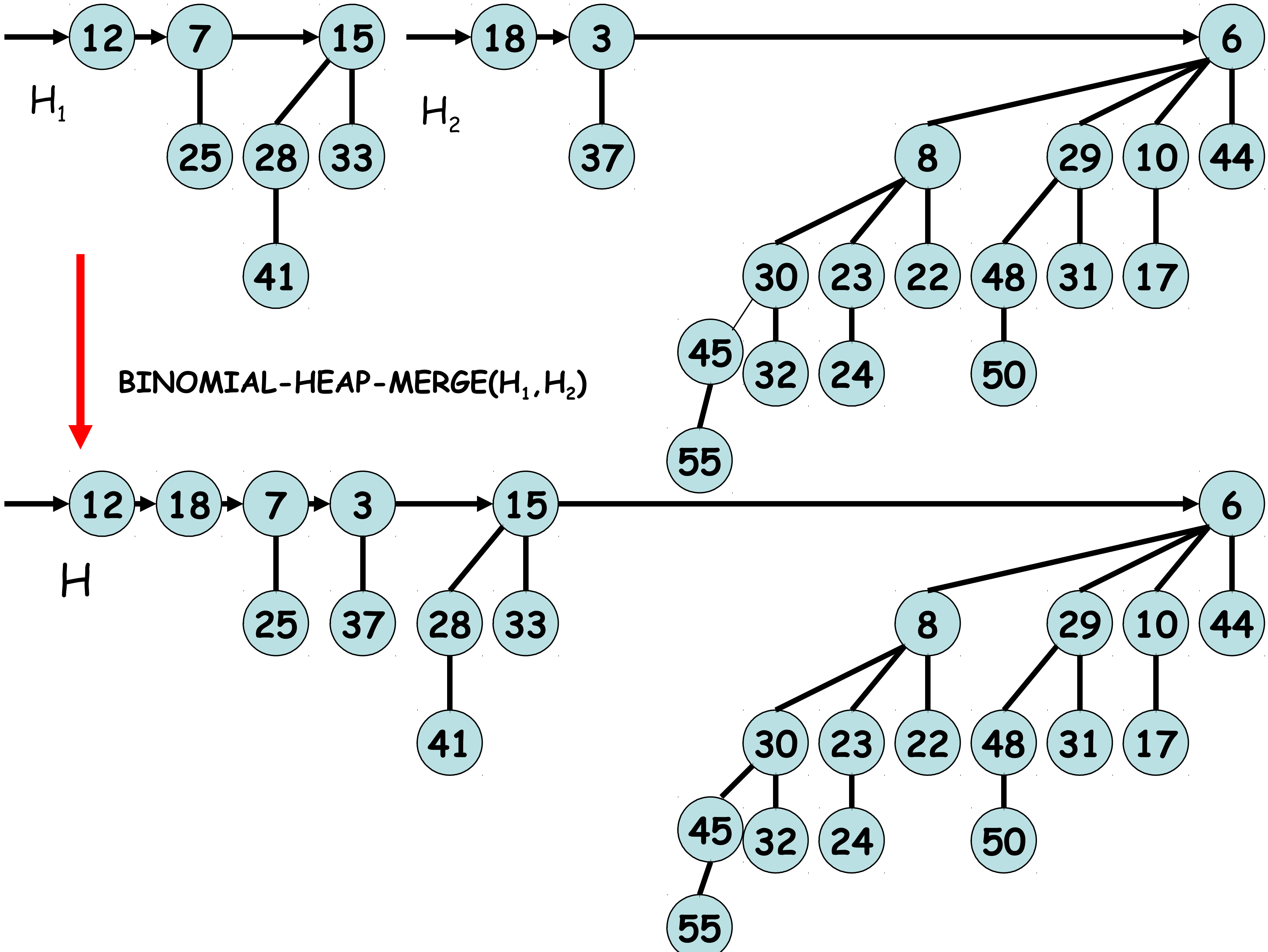


BINOMIAL-HEAP-MERGE (H1, H2)

Merges the root lists of H_1 & H_2 into a single linked-list.

Sorted by degree into monotonically increasing order.

BINOMIAL-HEAP-MERGE guarantees that if two roots in H have the same degree, they are adjacent in the root list.



H1
H2
BINOMIAL-HEAP-MERGE(H1,H2)
H

Uniting Two Binomial Heaps

Uses Two supporting Procedures

BINOMIAL-LINK (y, z)

and

BINOMIAL-HEAP-MERGE (H1, H2)

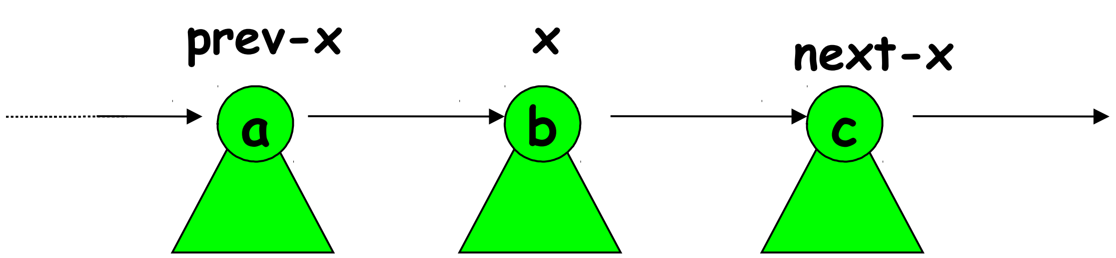
Uniting Two Binomial Heaps

We maintain 3 pointers into the root list:

x : points to the root currently being examined.

prev-x : points to the root PRECEDING *x* on the root list, sibling[prev-*x*] = *x*

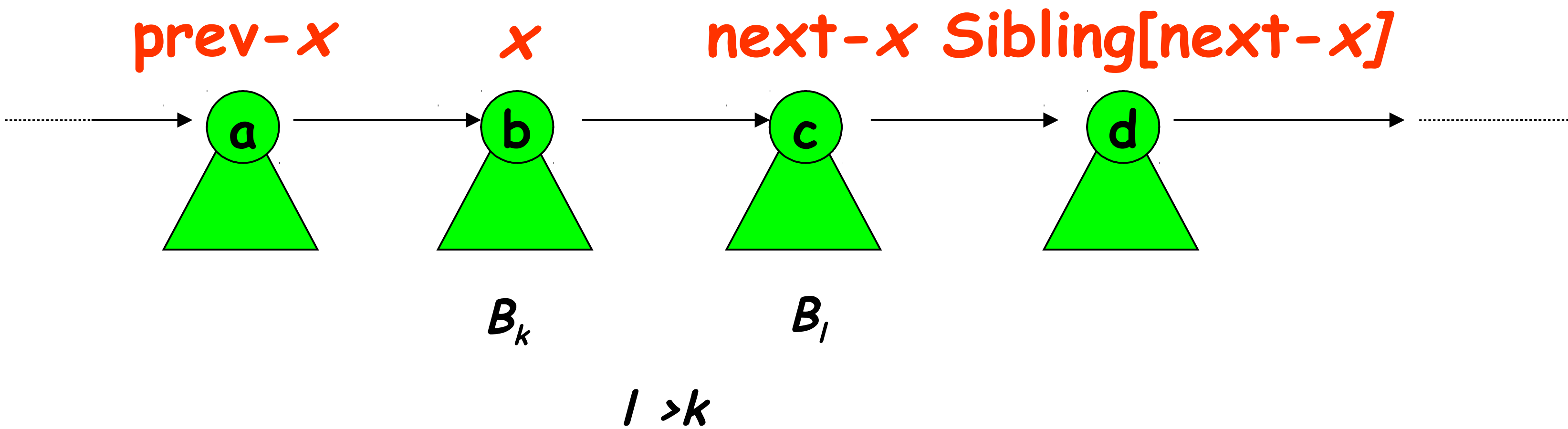
next-x : points to the root FOLLOWING *x* on the root list, sibling[*x*] = next-*x*

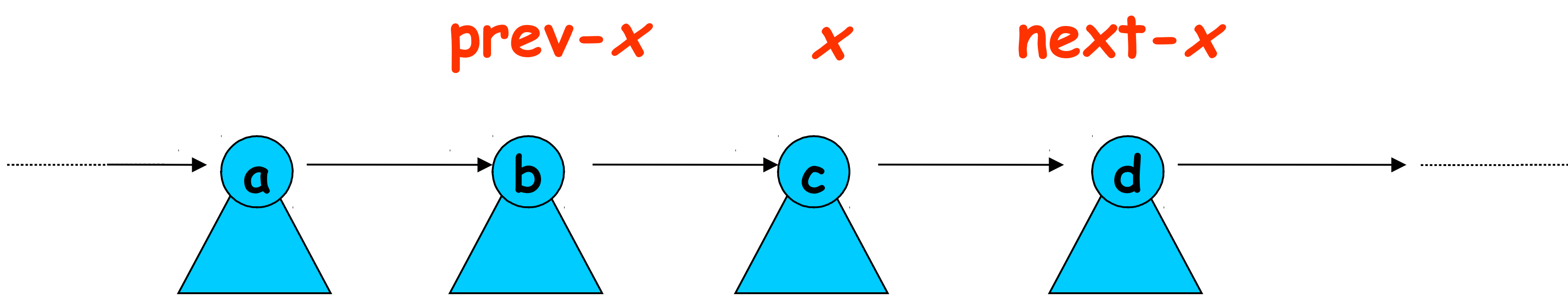


Learn DAA: From B K Sharma

Uniting Two Binomial Heaps

CASE 1: Occurs when degree [x] ≠ degree [next-x]

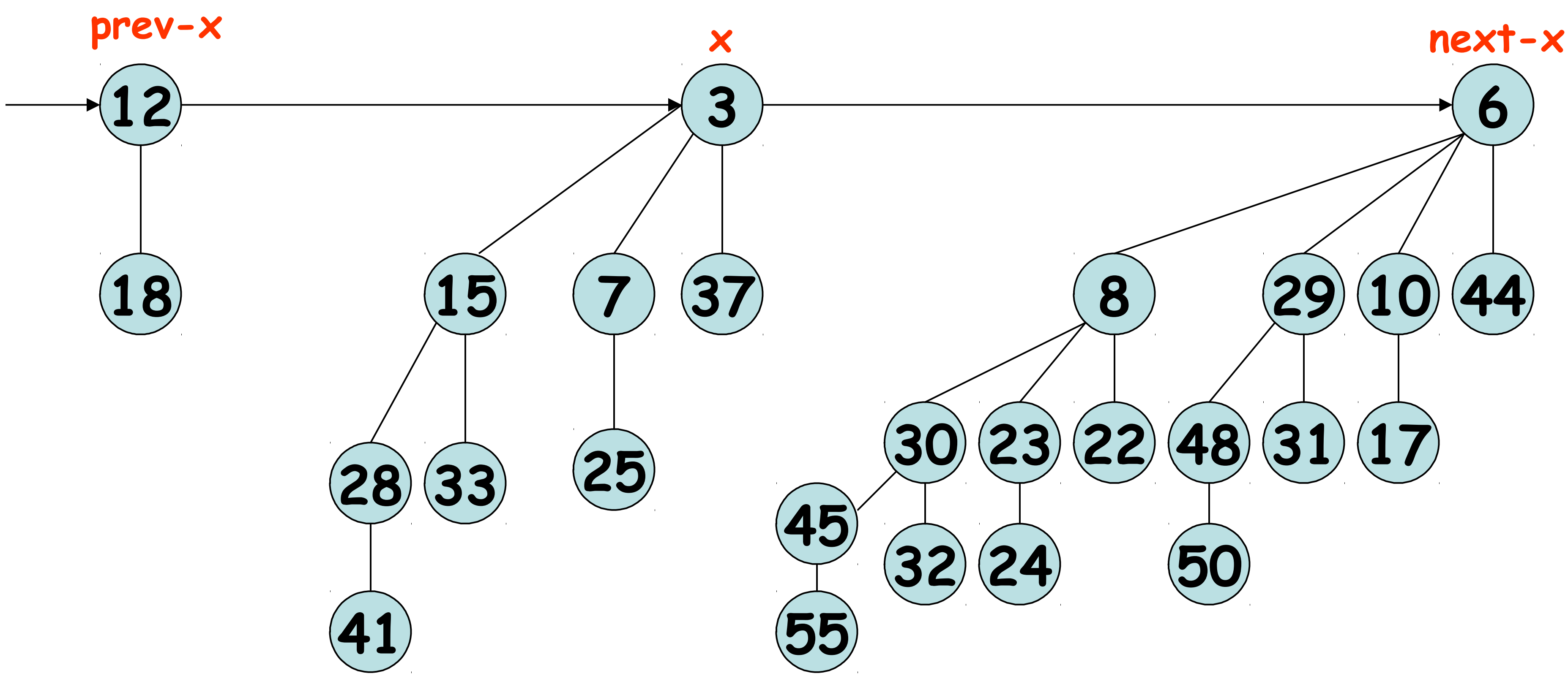




Learn DAA: From B K Sharma

Uniting Two Binomial Heaps

CASE 1: Occurs when degree [x] $\neq$ degree [next-x]

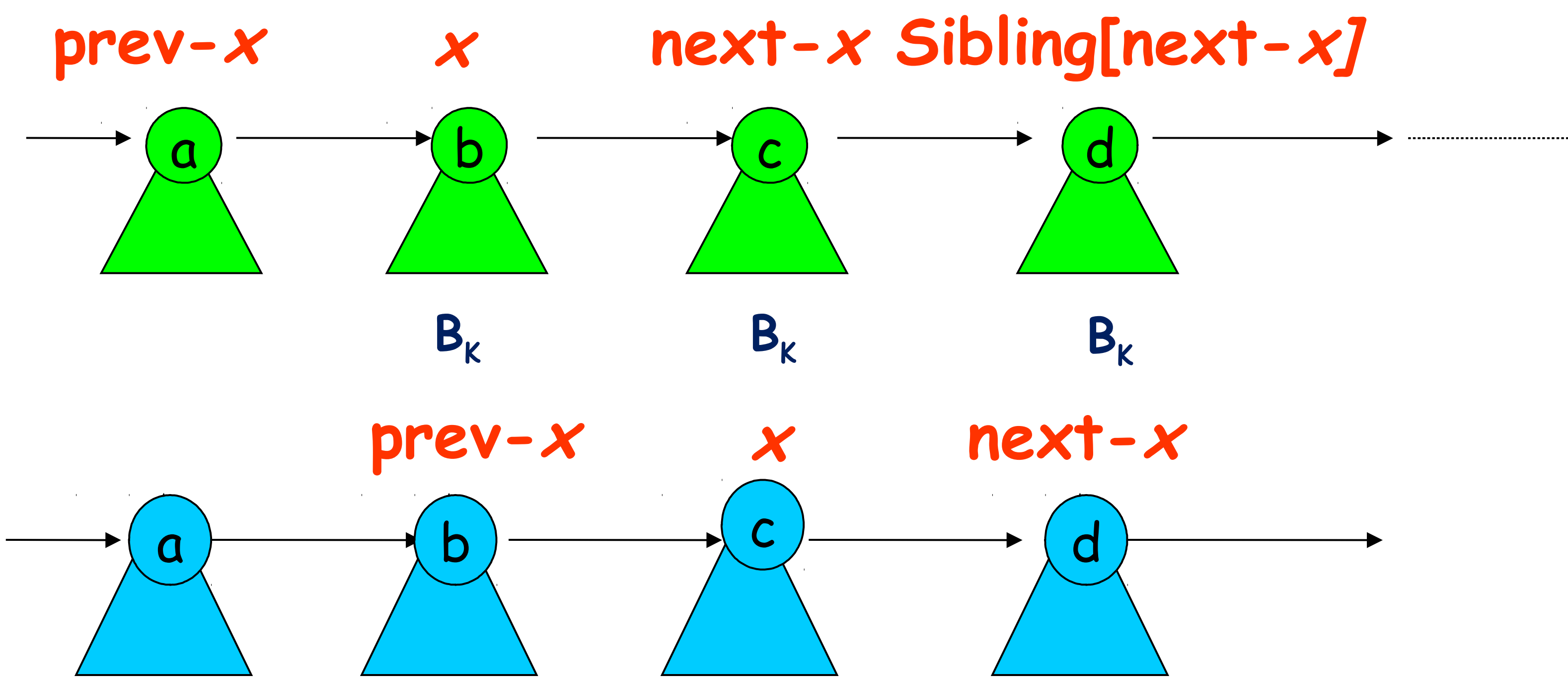


CASE 1

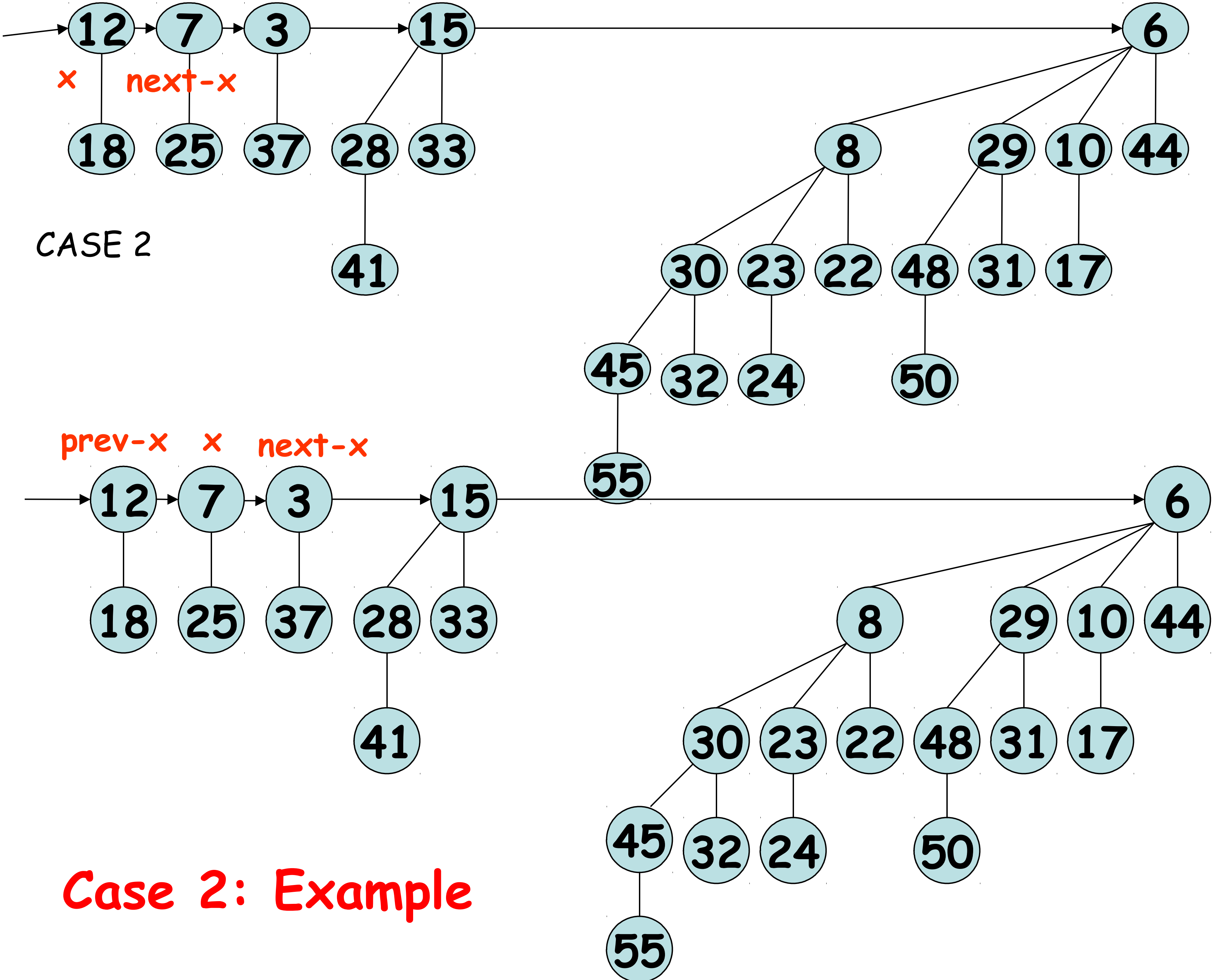
Uniting Two Binomial Heaps

CASE 2: Occurs when x is the first of 3 roots of equal degree

degree [*x*] = degree [next-*x*] = degree [sibling[next-*x*]]



Case 2: Example



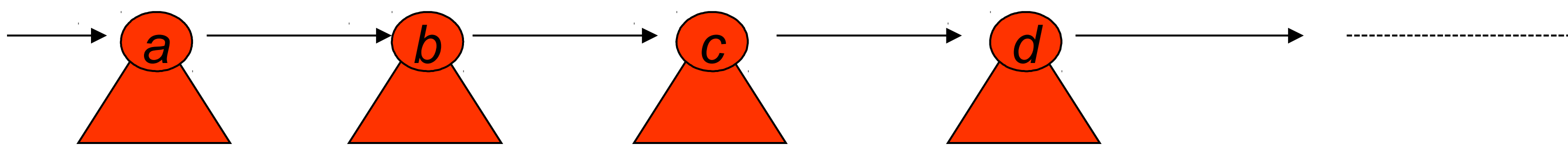
Learn DAA: From B K Sharma

Uniting Two Binomial Heaps

CASE 3 & 4: Occur when *x* is the first of 2 roots of equal degree

degree [*x*]=degree[next-*x*] ≠ degree [sibling [next-*x*]]

prev-*x* *x* next-*x* Sibling[next-*x*]



B_K B_K B_l l >k

CASE 3

key [x] ≤ key [next-x]

key [*b*] ≤ key [*c*]

CASE 4

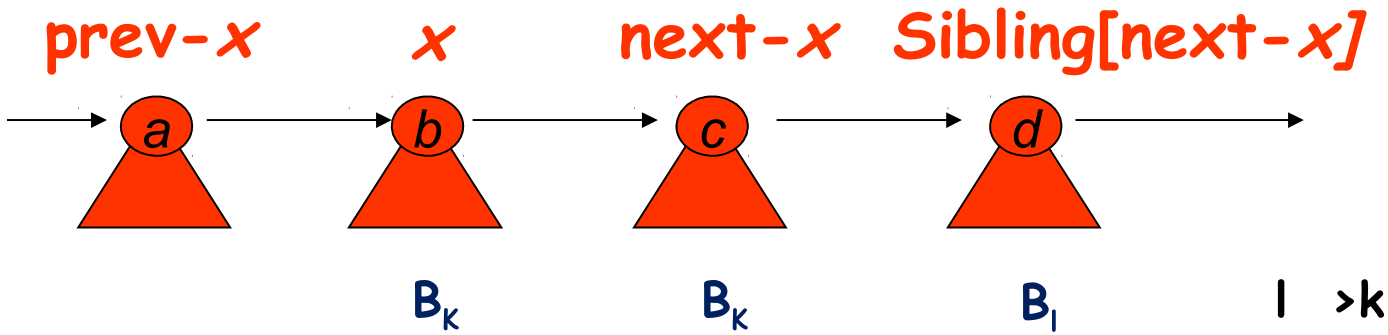
key [next-x] ≤ key [x]

key [c] ≤ key [b]

Uniting Two Binomial Heaps

CASE 3 & 4: Occur when *x* is the first of 2 roots of equal degree

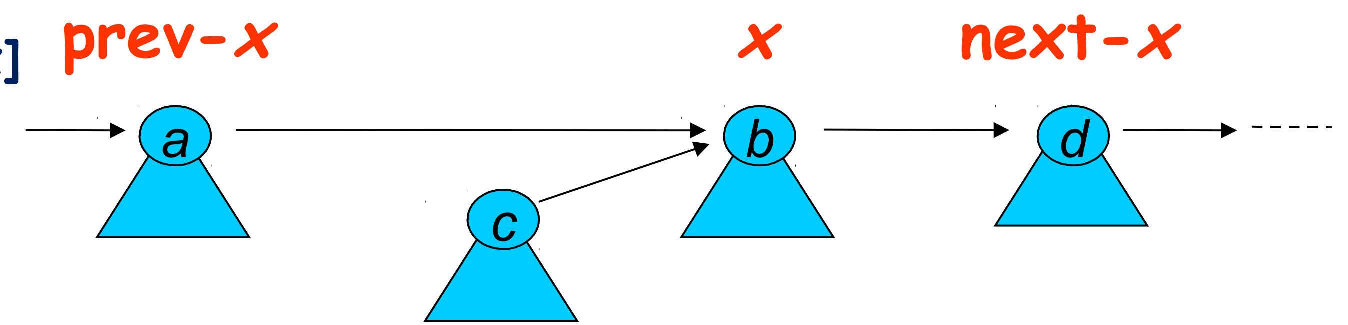
degree [*x*]=degree[next-*x*] ≠ degree [sibling [next-*x*]]



CASE 3

key [x] ≤ key [next-x]

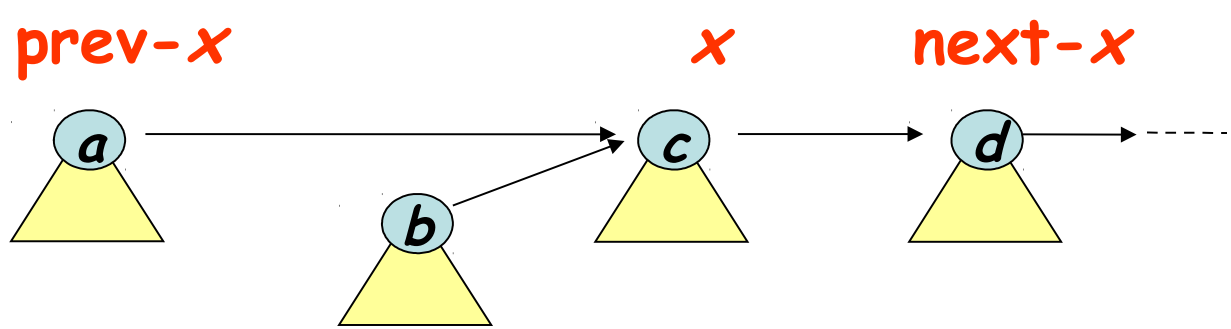
key [*b*] ≤ key [*c*]



CASE 4

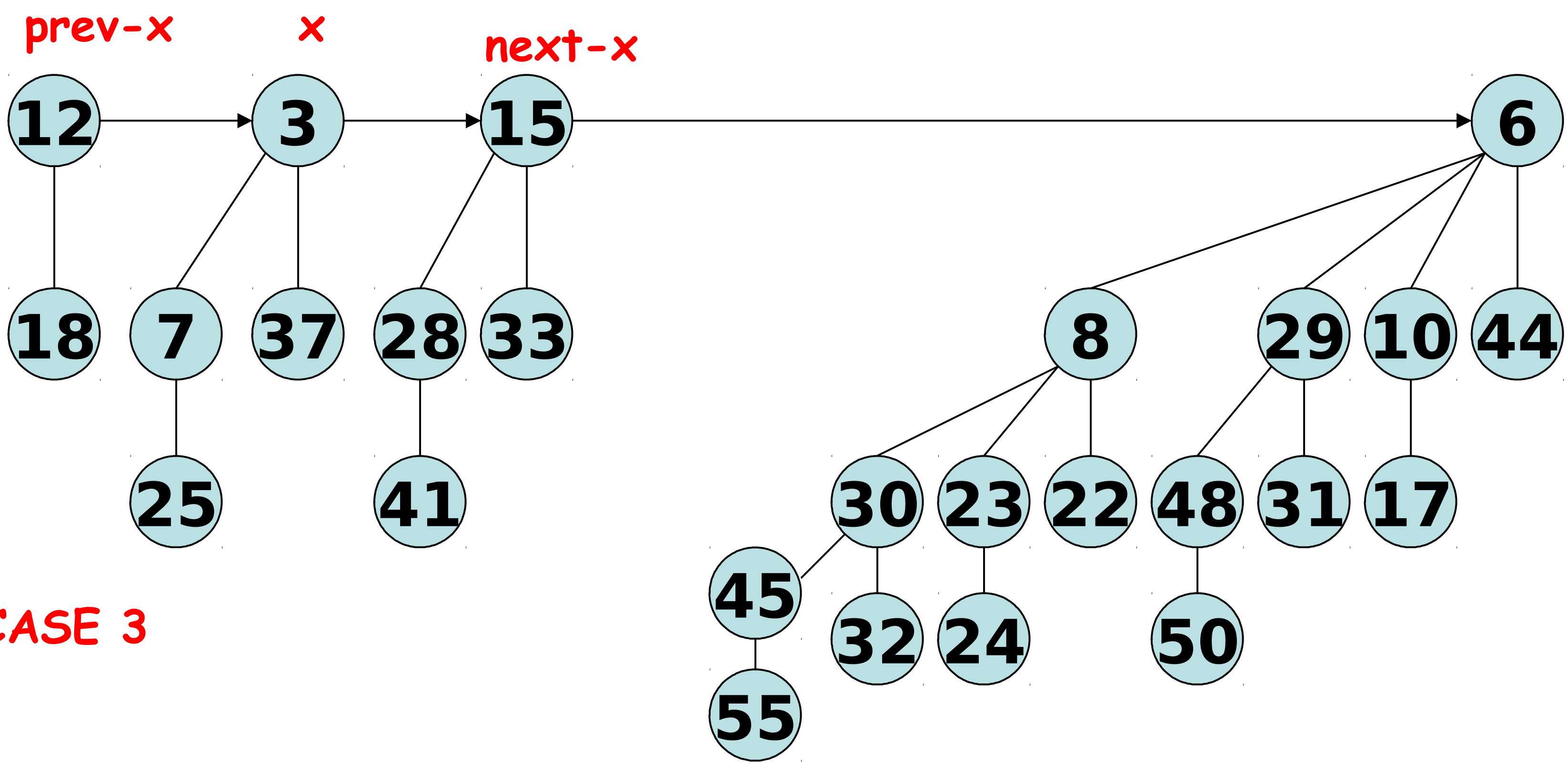
key [next-x] ≤ key [x]

key [c] ≤ key [b]



Uniting Two Binomial Heaps

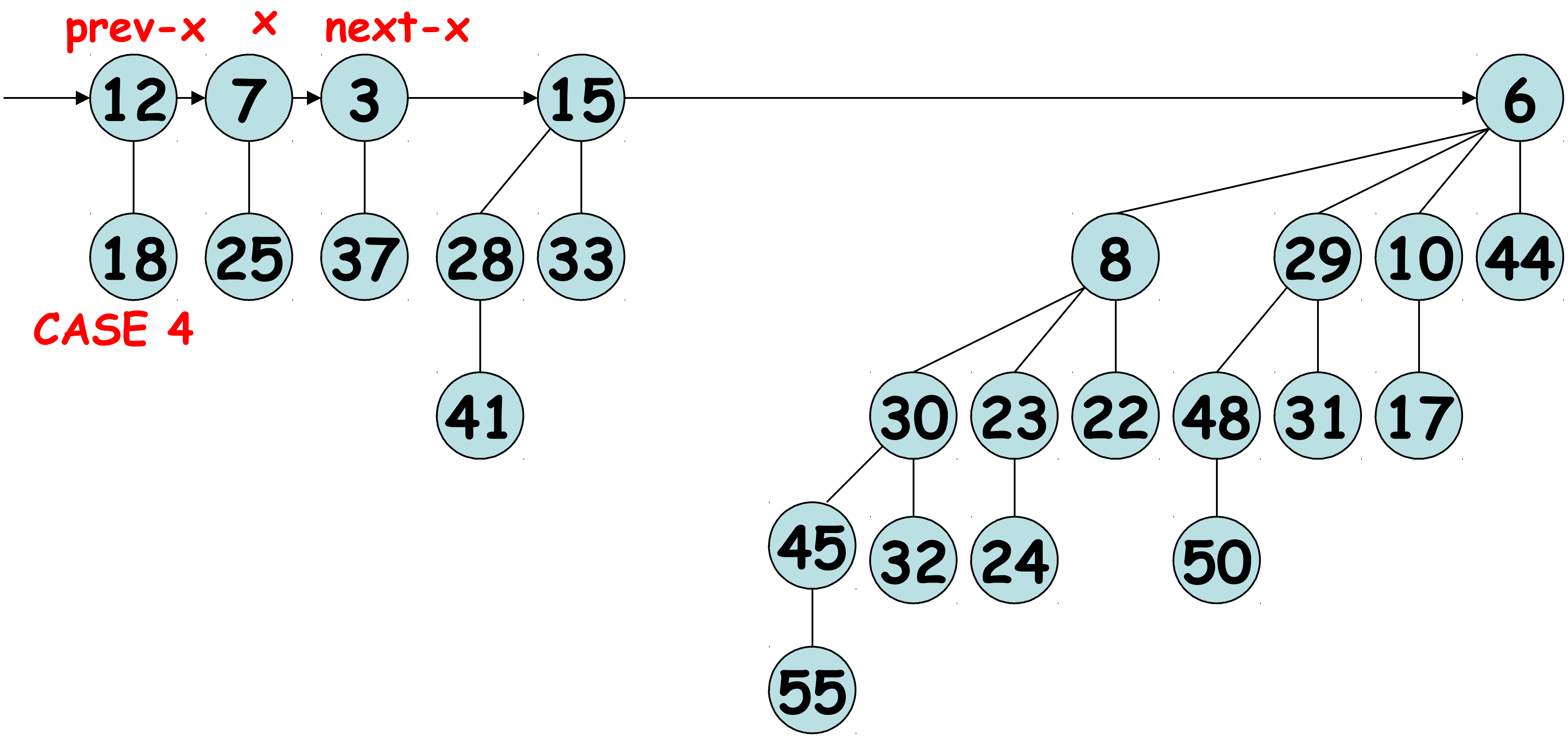
Case 3: Example



CASE 3

Uniting Two Binomial Heaps

Case 4: Example

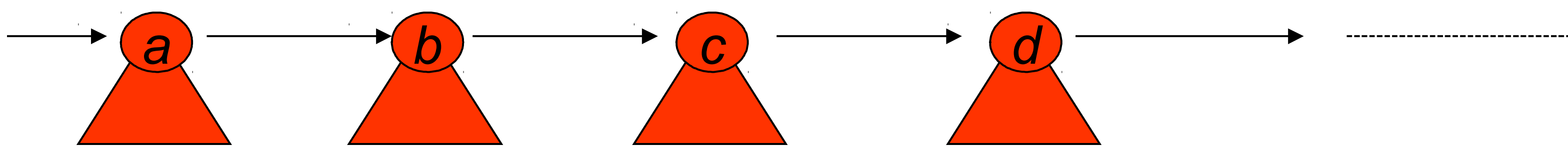


Uniting Two Binomial Heaps

CASE 3 & 4: Occur when *x* is the first of 2 roots of equal degree

degree [*x*]=degree[next-*x*] ≠ degree [sibling [next-*x*]]

prev-*x* *x* next-*x* Sibling[next-*x*]



B_K B_K B_l l >k

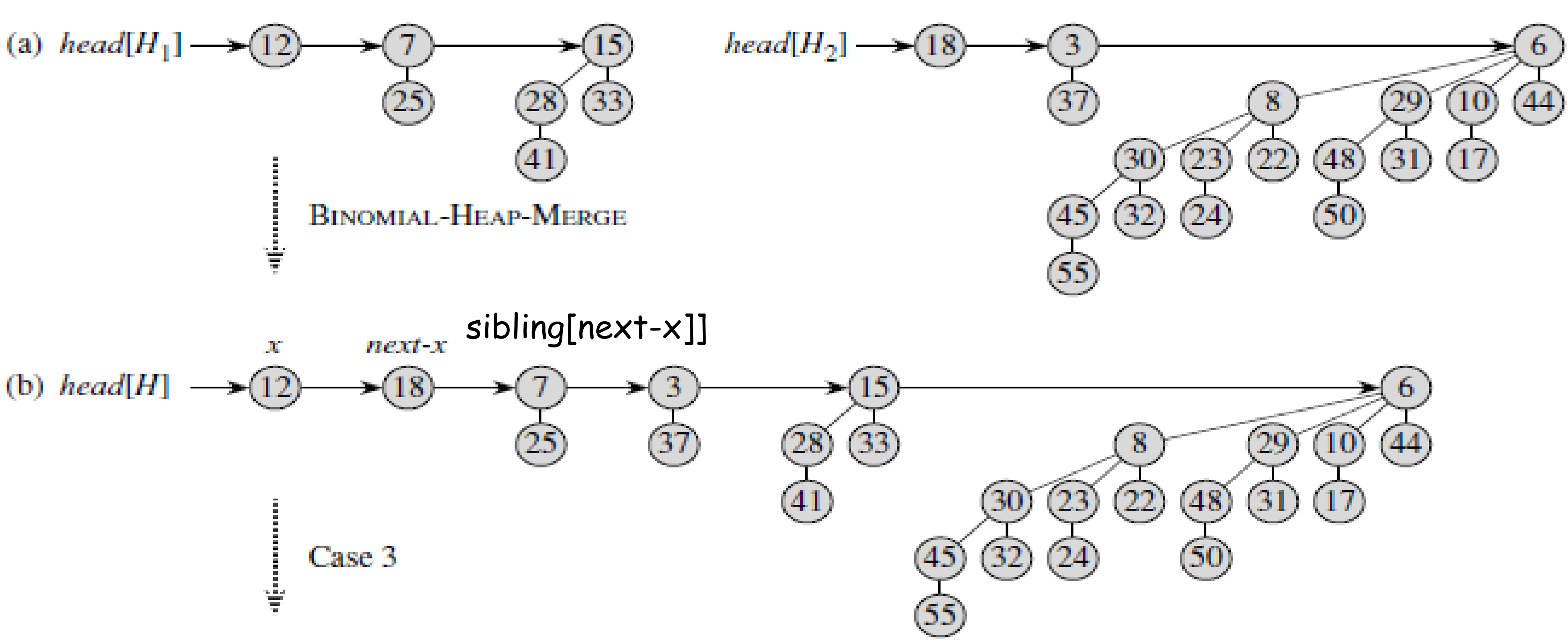
Occur on the next iteration after any case.

Always occur immediately following CASE 2.

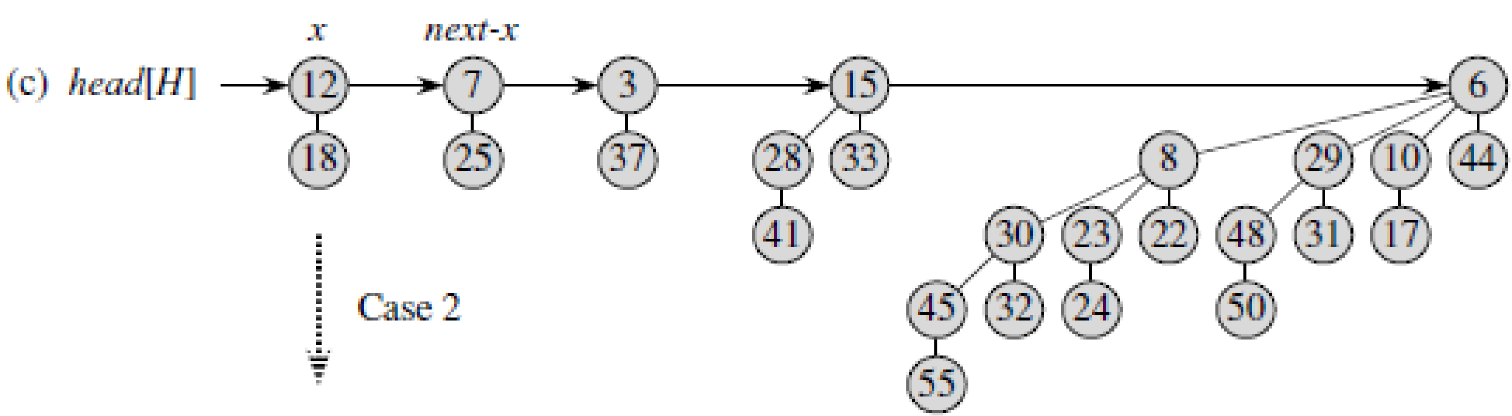
Two cases are distinguished by whether *x* or next-*x* has the smaller key.

The root with the smaller key becomes the root of the linked tree.

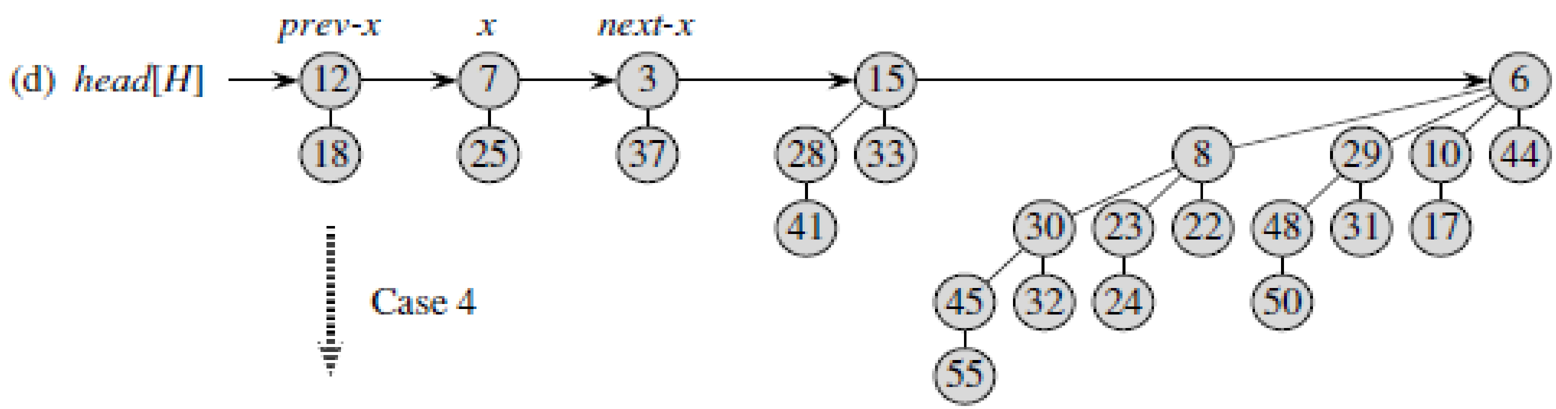
Union Example: First Merge H_1 and H_2



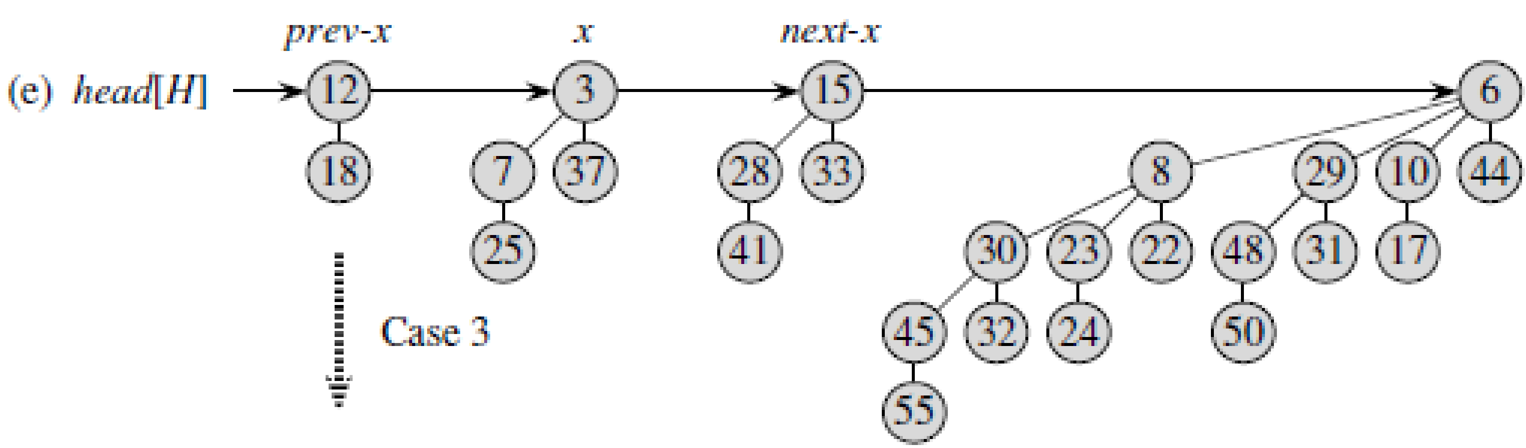
Result of Case 3



Result of Case 2



Result of Case 4



Result of Case 3