BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Data Structures and Algorithms
CS F211

Vishal Gupta

Department of Computer Science and Information Systems

Birla Institute of Technology and Science

Pilani Campus, Pilani

Agenda: Binomial Heaps

Binomial Heaps

DATA STRUCTURES: MERGEABLE HEAPS

- MAKE-HEAP ( )
  - Creates & returns a new heap with no elements.
- INSERT (H,x)
  - Inserts a node *x* into heap H. key field of the node has already been filled.
- MINIMUM (H)
  - Returns a pointer to the node in heap *H* whose key is minimum.

Mergeable Heaps

- EXTRACT-MIN (*H*)
  - Deletes the node from heap *H* whose key is minimum. Returns a pointer to the node.

- DECREASE-KEY (*H*, *x*, *k*)
  - Assigns to node *x* within heap *H* the new value *k* where *k* is smaller than its current key value.

Mergeable Heaps

- DELETE (H, x)
  - Deletes node x from heap H.
- UNION (H_1, H_2)
  - Creates and returns a new heap that contains all nodes of heaps H_1 & H_2.
  - Heaps H_1 & H_2 are *destroyed* by this operation

Binomial Trees

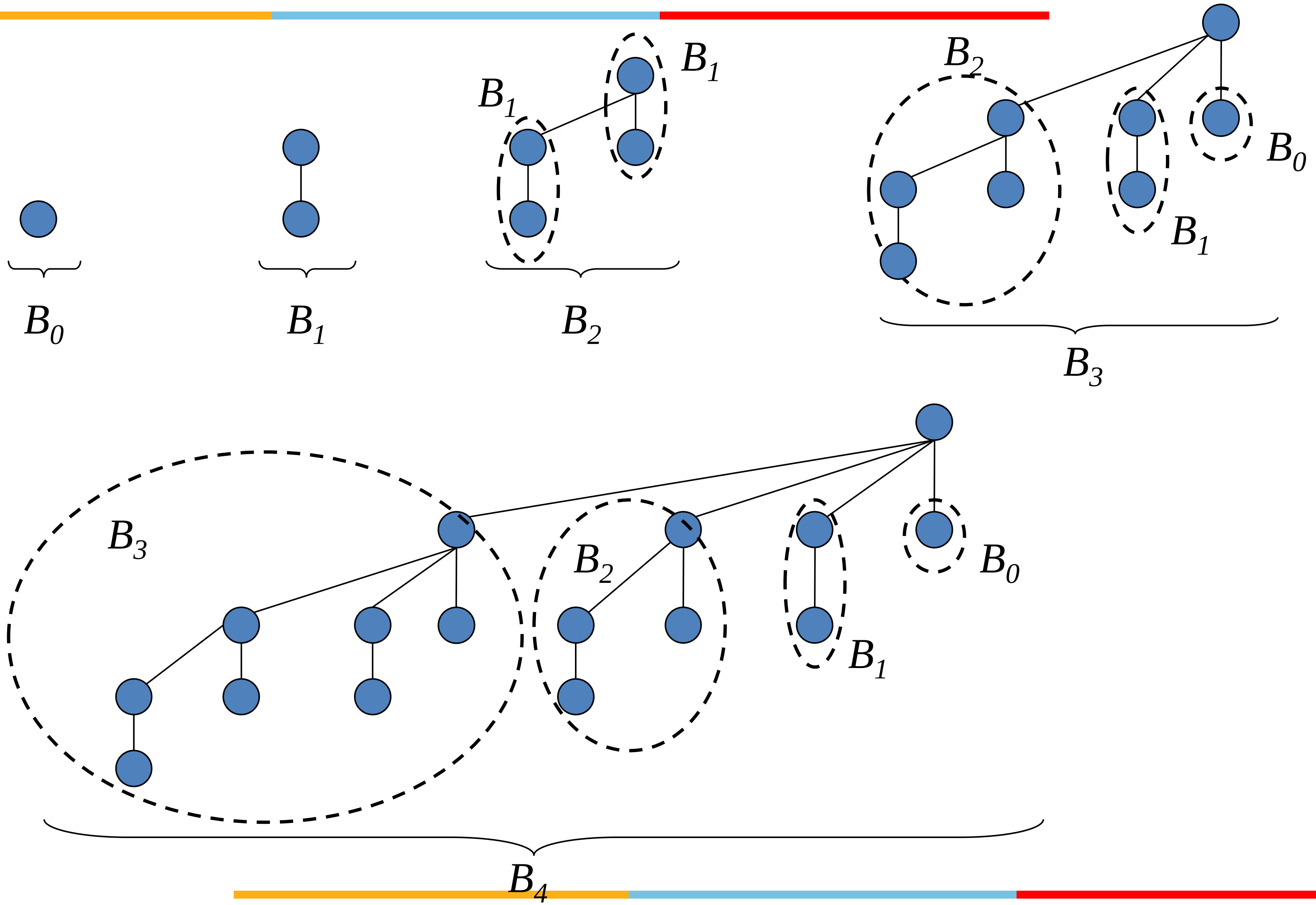
- A binomial heap is a collection of binomial trees.
- The binomial tree B_k is an ordered tree defined recursively

  B_o Consists of a single node

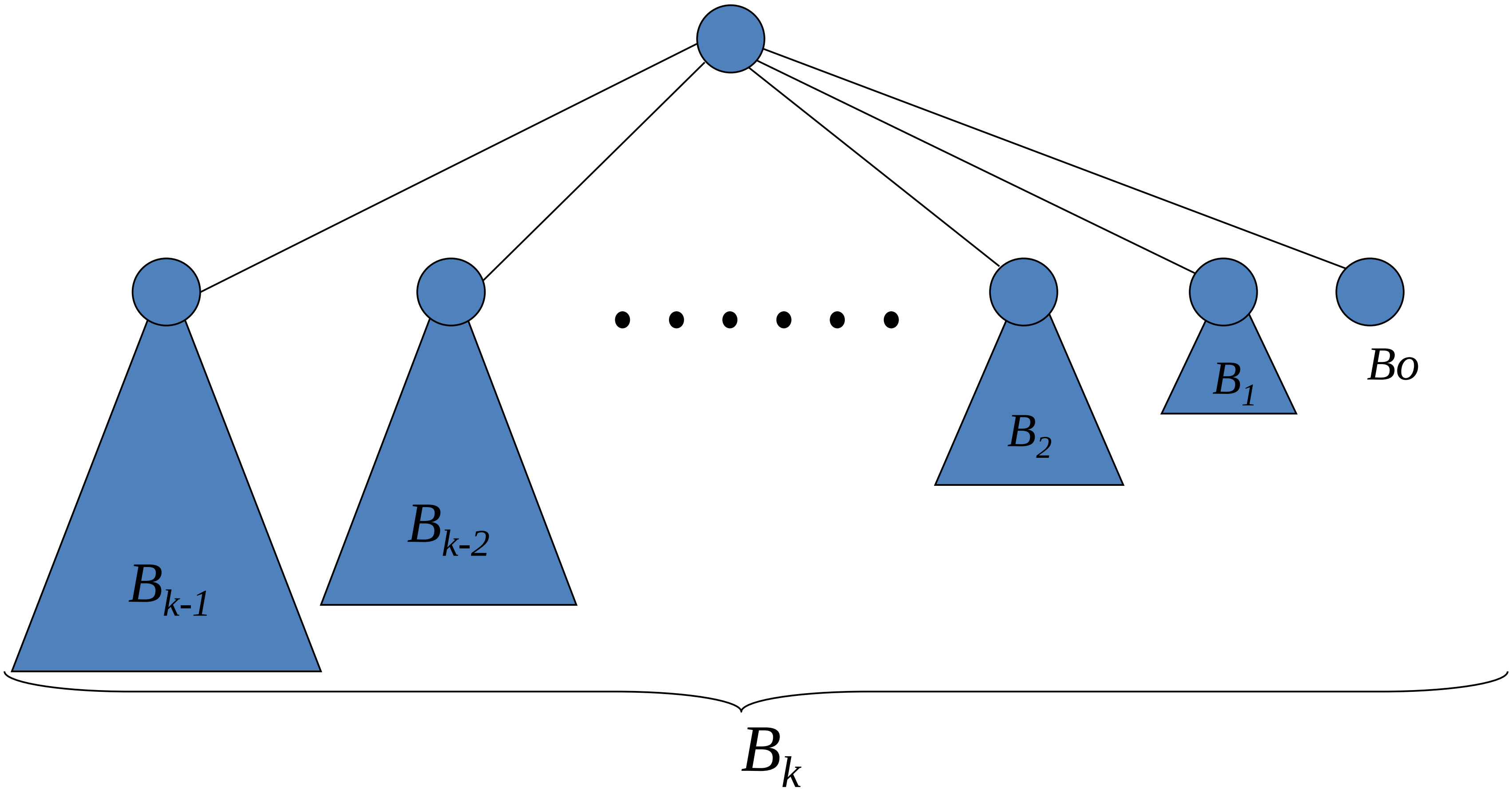
  .
  .
  .

  B_k Consists of two binominal trees B_{k-1} linked together. Root of one is the leftmost child of the root of the other.

Binomial Trees



Binomial Trees



Properties of Binomial Trees

LEMMA: For the binomial tree B_k ;

1. There are _____ nodes,
2. The height of tree is ____,
3. There are exactly _____ nodes at depth i for i = 0,1,..,k and
4. The root has degree ____ > degree of any other node if the children of the root are numbered from left to right as k-1, k-2,...,0; child i is the root of a subtree B_i.

Properties of Binomial Trees

PROOF: By induction on k

Each property holds for the basis B_0

INDUCTIVE STEP: assume that Lemma holds for B_{k-1}

1. B_k consists of two copies of B_{k-1}

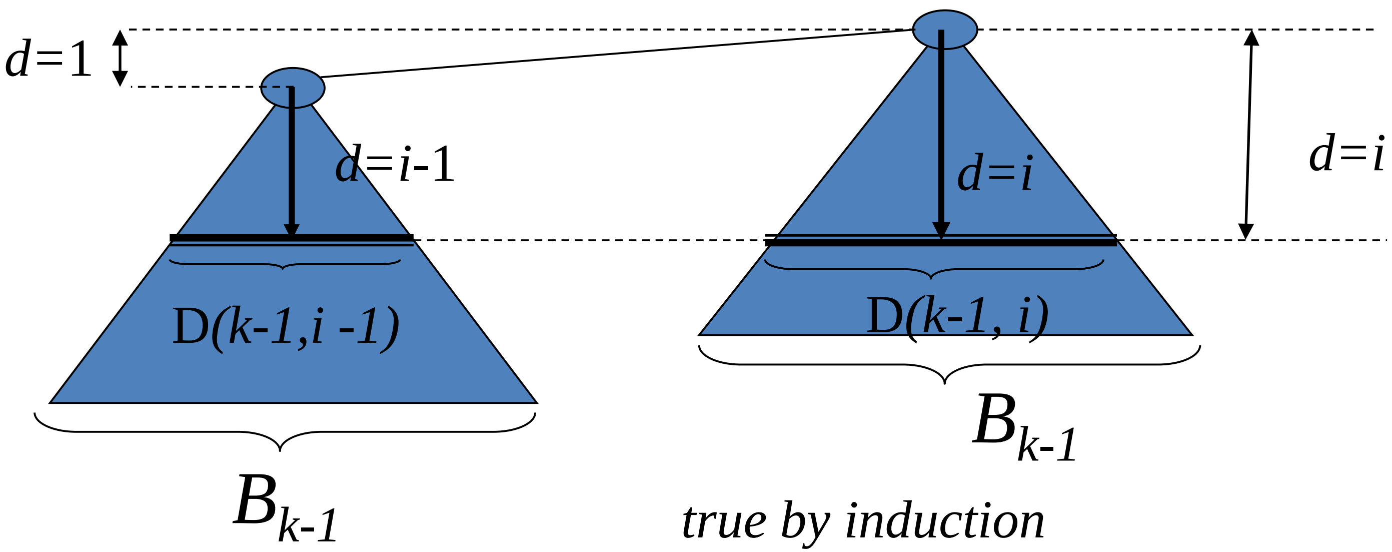
   $\therefore \mid B_k \mid = \mid B_{k-1} \mid + \mid B_{k-1} \mid = 2^{k-1} + 2^{k-1} = 2^k$

2. h_{k-1} = Height (B_{k-1}) = k-1 by induction

   $h_k = h_{k-1} + 1 = k\text{-}1 + 1 = k$

Properties of Binomial Trees

3. Let D(k,i) denote the number of nodes at depth i of a B_k ;



$$D(k,i)=D(k-1,i-1) + D(k-1,i) = \binom{k-1}{i-1} + \binom{k-1}{i} = \binom{k}{i}$$

Properties of Binomial Trees(Cont.)

4. Only node with greater degree in B_k than those in B_{k-1} is the root,

- The root of B_k has one more child than the root of B_{k-1},

- Degree of root B_k=Degree of B_{k-1}+1=(k-1)+1=k

Properties of Binomial Trees (Cont.)

- COROLLARY: The maximum degree of any node in an n-node binomial tree is lg(n)

  The term BINOMIAL TREE comes from the 3rd property.

  i.e. There are $\binom{k}{i}$ nodes at depth i of a B_k

  terms $\binom{k}{i}$ are the binomial coefficients.

Binomial Heaps

A BINOMIAL HEAP *H* is a set of BINOMIAL TREES that satisfies the following "Binomial Heap Properties"

1. Each binomial tree in *H* is HEAP-ORDERED
   - the key of a node is $\geq$ the key of the parent
   - Root of each binomial tree in *H* contains the smallest key in that tree.

Binomial Heaps

2. There is at most one binomial tree in *H* whose root has a given degree,
   - n-node binomial heap H consists of at most $[\lg n] + 1$ binomial trees.
   - Binary represantation of n has $\lg(n) + 1$ bits,

$$n \le b_{\lfloor \lg n \rfloor}, b_{\lfloor \lg n \rfloor -1}, \ldots b_1, b_0 > = \sum_{i=0}^{\lfloor \lg n \rfloor} b_i 2^i$$

By property 1 of the lemma (B_i contains 2^i nodes) B_i appears in H iff bit b_i=1

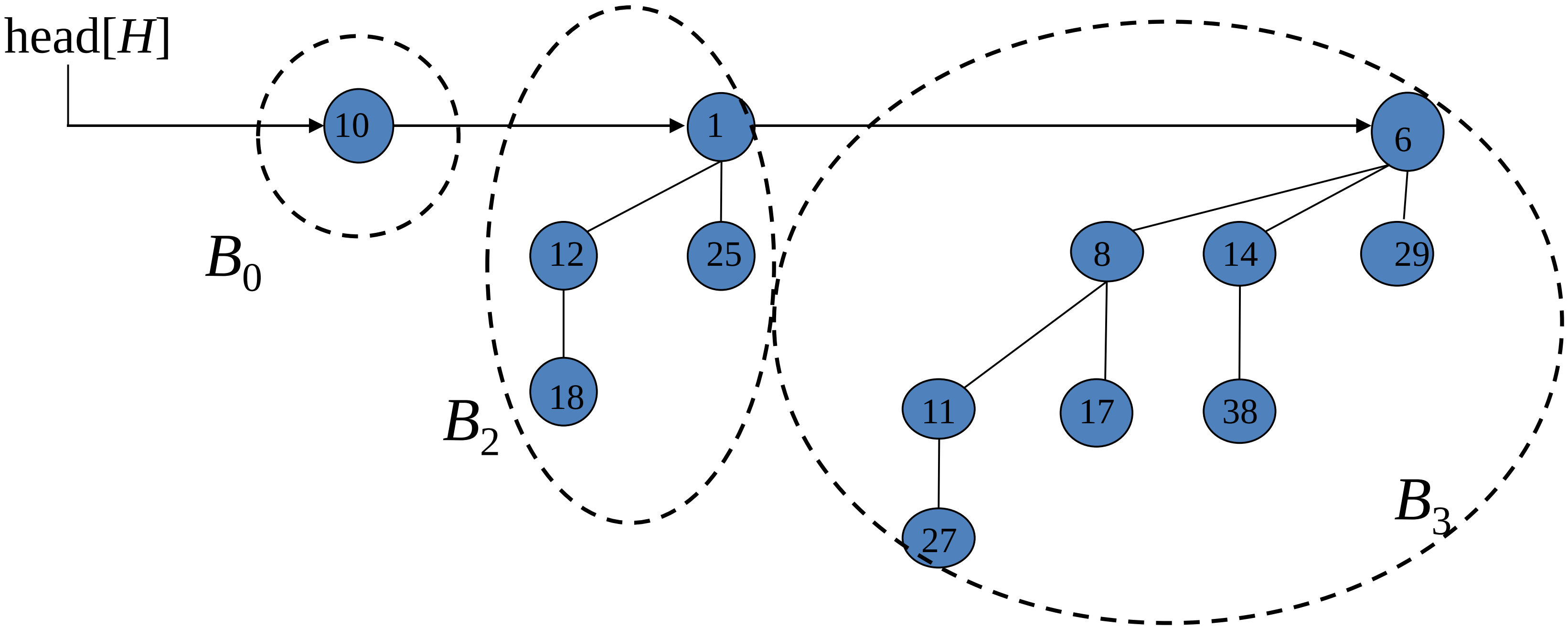
Binomial Heaps

Example: A binomial heap with n = 13 nodes

$$\begin{array}{cccc} 3 & 2 & 1 & 0 \end{array}$$

$13 =< 1, 1, 0, 1>_2$

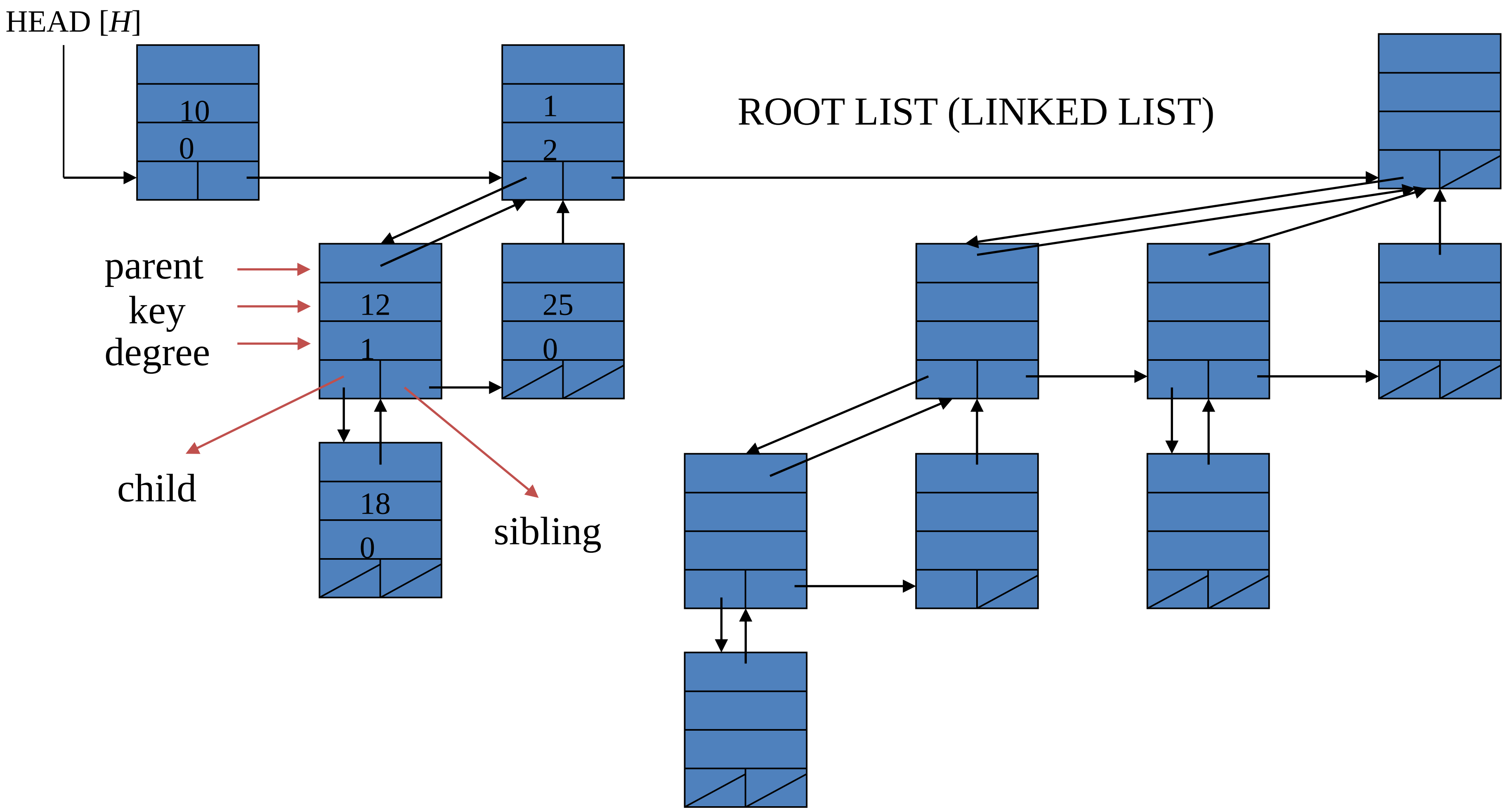
Consists of B_0, B_2, B_3



Representation of Binomial Heaps

- Each binomial tree within a binomial heap is stored in the left-child, right-sibling representation
- Each node *X* contains POINTERS
  - *p*[*x*] to its parent
  - child[*x*] to its leftmost child
  - sibling[*x*] to its immediately right sibling
- Each node *X* also contains the field degree[*x*] which denotes the number of children of *X*.

Representation of Binomial Heaps



Representation of Binomial Heaps

- Let x be a node with sibling[x] $\neq$ NIL
  - Degree [sibling [x]]=degree[x]-1
    if x is NOT A ROOT
  - Degree [sibling [x]] > degree[x]
    if x is a root

Operations on Binomial Heaps

CREATING A NEW BINOMIAL HEAP

MAKE-BINOMIAL-HEAP ( )

allocate *H*

head [*H*] ← NIL

return *H*

end

RUNNING-TIME= Θ(1)

Operations on Binomial Heaps

BINOMIAL-HEAP-MINIMUM (H)

$x \leftarrow$ Head [H]
min $\leftarrow$ key [x]
$x \leftarrow$ sibling [x]
while $x \neq$ NIL do
    if key [x] < min then
        min $\leftarrow$ key [x]
        $y \leftarrow x$
    endif
        $x \leftarrow$ sibling [x]
endwhile
return y
end

Operations on Binomial Heaps

Since binomial heap is HEAP-ORDERED

The minimum key must reside in a ROOT NODE

Above procedure checks all roots

NUMBER OF ROOTS $\leq \lfloor \lg n \rfloor + 1$

$\therefore$ RUNNING–TIME = O (lg$n$)

Uniting Two Binomial Heaps

BINOMIAL-HEAP-UNION

Procedure repeatedly link binomial trees whose roots have the same degree

BINOMIAL-LINK

Procedure links the B_{k-1} tree rooted at node *y* to the B_{k-1} tree rooted at node z to make z the parent of *y* i.e. Node *z* becomes the root of a B_k tree

Uniting Two Binomial Heaps

BINOMIAL-LINK (y,z)

p $[y] \leftarrow z$

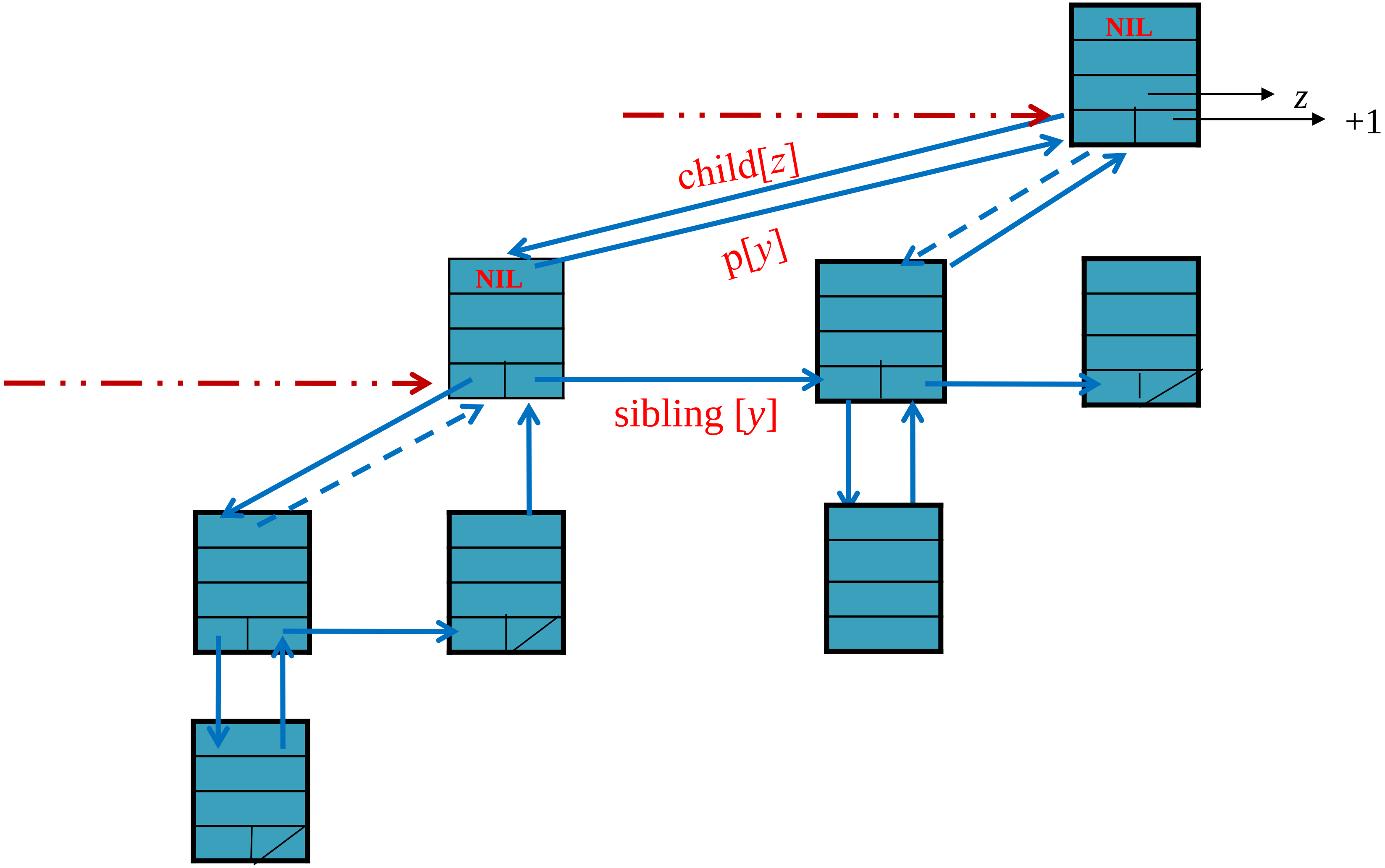
sibling $[y] \leftarrow$ child $[z]$

child $[z] \leftarrow y$

degree $[z] \leftarrow$ degree $[z] + 1$

end

Uniting Two Binomial Heaps



Uniting Two Binomial Heaps: Cases

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

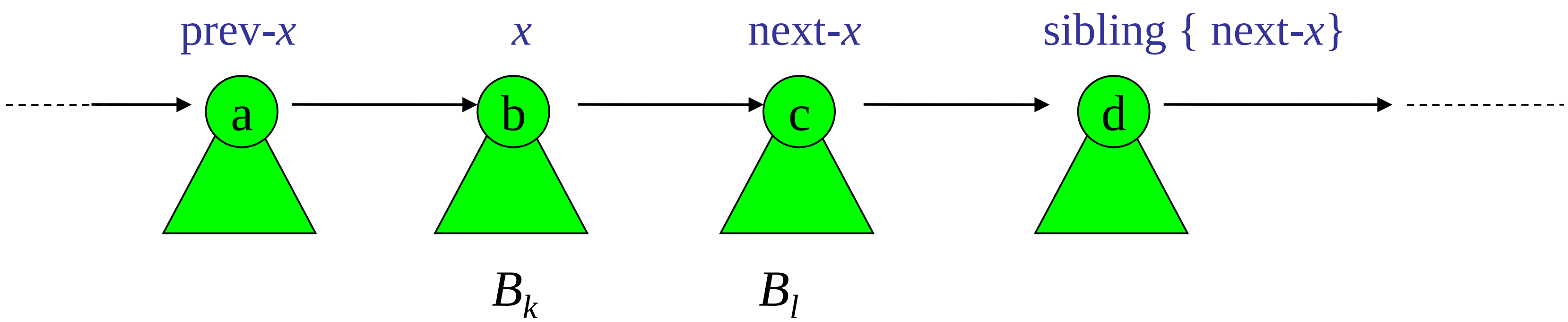
next-x = points to the root FOLLOWING x on the root list sibling [x] = next-x

Uniting Two Binomial Heaps

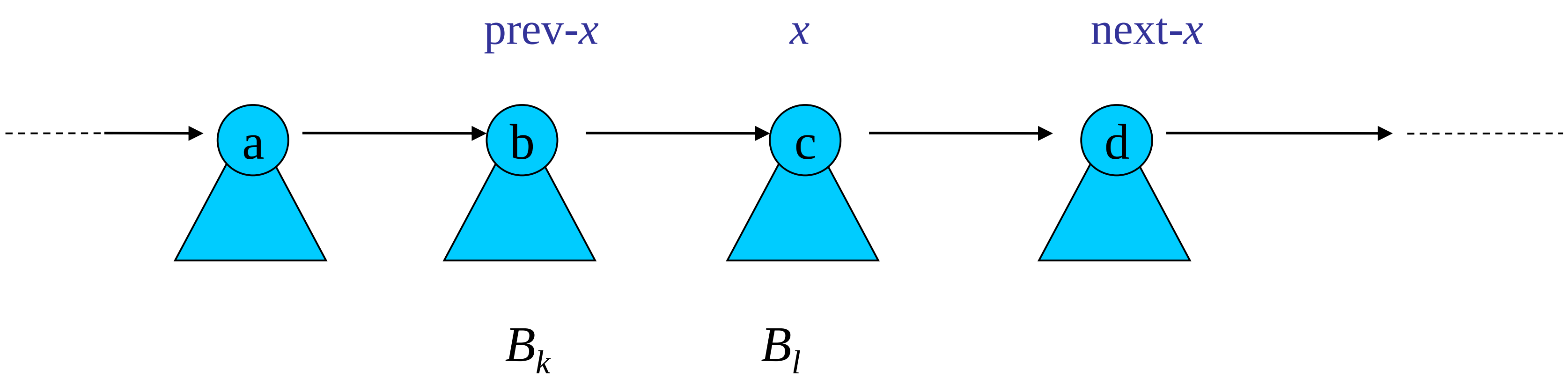
- Initially, there are at most two roots of the same degree
- Binomial-heap-merge guarantees that if two roots in h have the same degree they are adjacent in the root list
- During the execution of union, there may be three roots of the same degree appearing on the root list at some time

Uniting Two Binomial Heaps

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



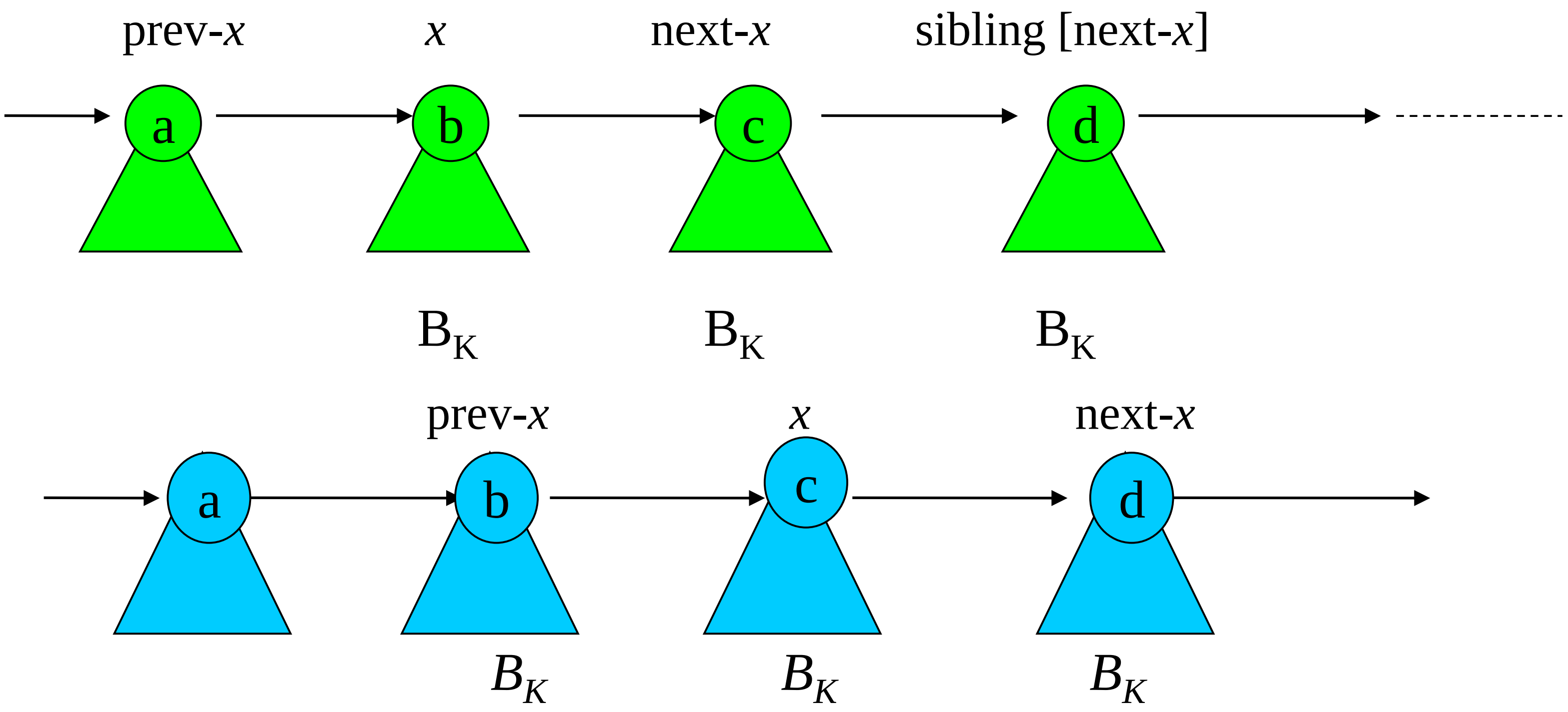
$l > k$



Uniting Two Binomial Heaps: Cases

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

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



Uniting Two Binomial Heaps: Cases

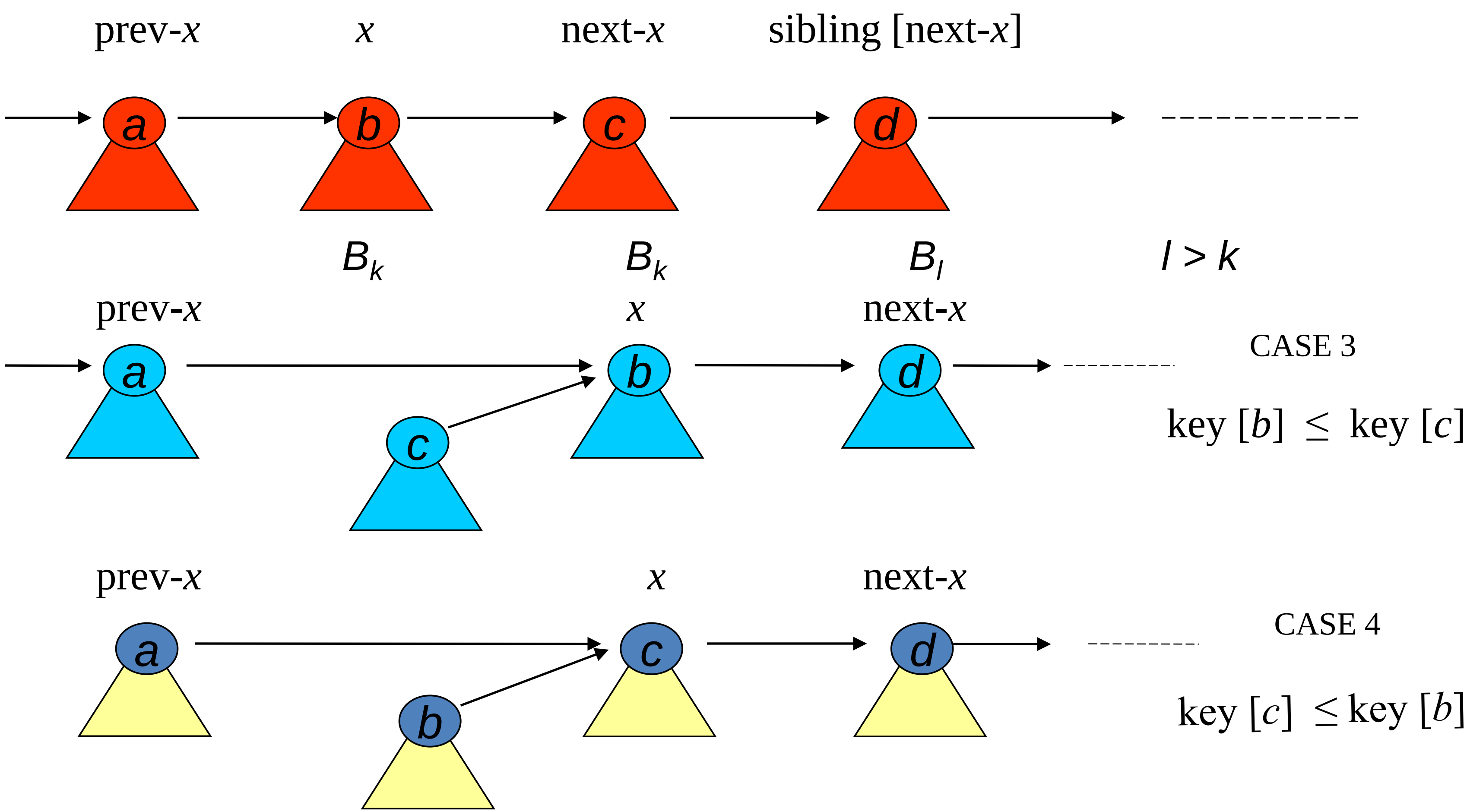
CASE 3 & 4: Occur when x is the first of 2 roots of equal degree

$$\text{degree}[x] = \text{degree}[\text{next-}x] \neq \text{degree}[\text{sibling}[\text{next-}x]]$$

- 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

Uniting Two Binomial Heaps: Cases

CASE 3 & 4 CONTINUED



Uniting Two Binomial Heaps: Cases

The running time of binomial-heap-union operation is $O(\lg n)$

- Let H_1 & H_2 contain n_1 & n_2 nodes respectively where $n = n_1 + n_2$

- Then, H_1 contains at most $\lfloor \lg n_1 \rfloor + 1$ roots
  H_2 contains at most $\lfloor \lg n_2 \rfloor + 1$ roots

Uniting Two Binomial Heaps: Cases

- So H contains at most $\lfloor \lg n_1 \rfloor + \lfloor \lg n_2 \rfloor + 2 \leq 2\lfloor \lg n \rfloor + 2 = O(\lg n)$ roots immediately after BINOMIAL-HEAP-MERGE

- Therefore, BINOMIAL-HEAP-MERGE runs in $O(\lg n)$ time and

- BINOMIAL-HEAP-UNION runs in $O(\lg n)$ time

Binomial-Heap-Union Procedure

BINOMIAL-HEAP-MERGE PROCEDURE

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

- Sorted by degree into monotonically increasing order

Binomial-Heap-Union Procedure

BINOMIAL-HEAP-UNION (H_1,H_2)

H $\leftarrow$ MAKE-BINOMIAL-HEAP ( )

head [H] $\leftarrow$ BINOMIAL-HEAP-MERGE (H_1,H_2)

free the objects H_1 & H_2 but not the lists they point to

prev-x $\leftarrow$ NIL

x $\leftarrow$ HEAD [H]

next-x $\leftarrow$ sibling [x]

while next-$x \neq$ NIL do

if (degree [x] $\neq$ degree [next-x] OR

(sibling [next-x] $\neq$ NIL and degree[sibling [next-x]] = degree [x]) then

prev-$x \leftarrow x$ CASE 1 and 2

$x \leftarrow$ next-x CASE 1 and 2

elseif key [x] $\leq$ key [next-x] then

sibling [x] $\leftarrow$ sibling [next -x] CASE 3

Binomial-Heap-Union Procedure (Cont.)

```
                BINOMIAL- LINK (next-x, x)                        CASE 3
      else
                if prev-x = NIL then
                        head [H] ←  next-x                        CASE 4
                else                                              CASE 4
                        sibling [prev-x] ←  next-x                CASE 4
                endif
                BINOMIAL-LINK(x, next-x)                          CASE 4
                x  ← next-x                                       CASE 4
      endif
      next-x ← sibling [x]
   endwhile
   return H
end
```

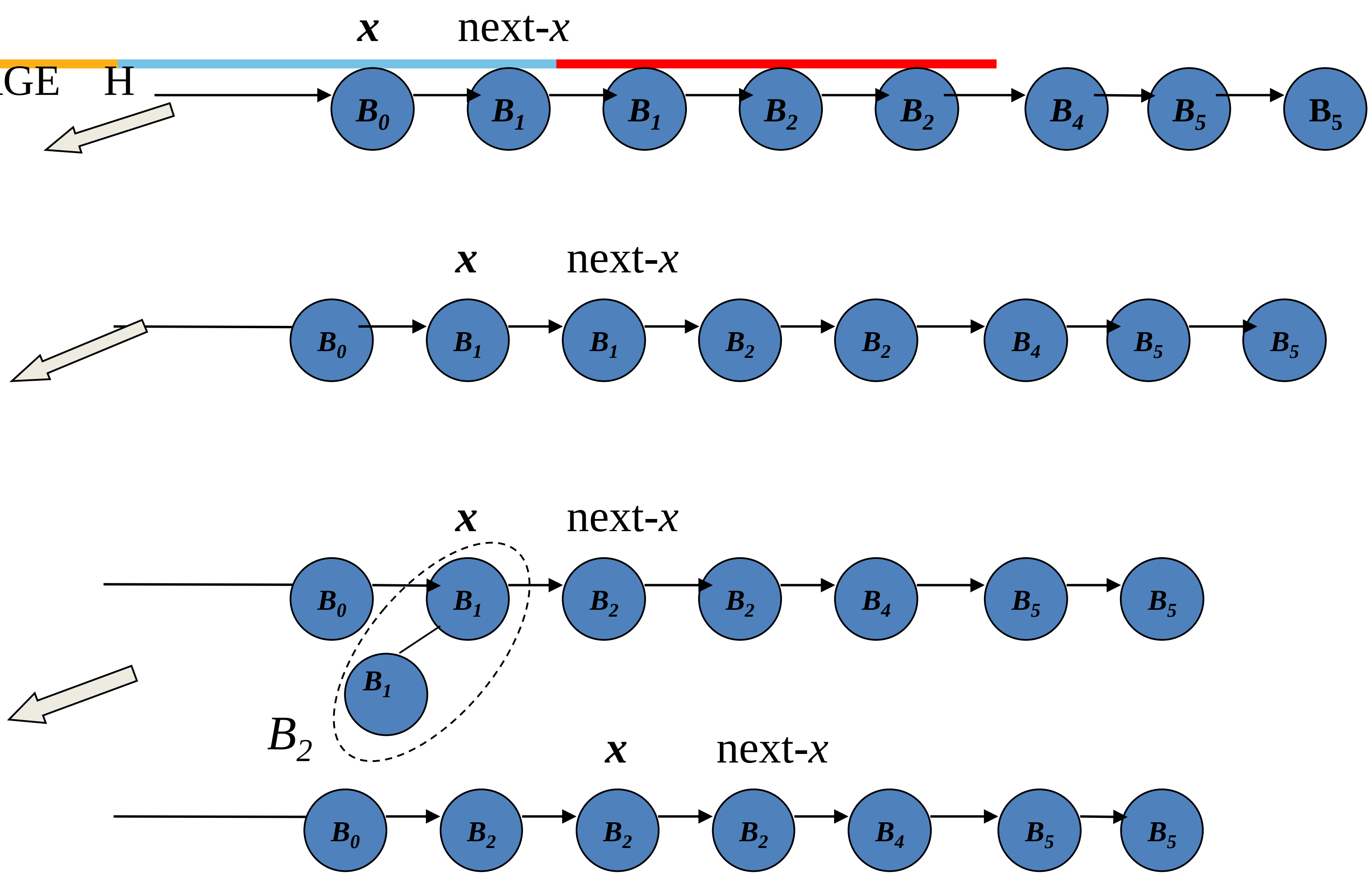
Uniting Two Binomial Heaps vs Adding Two Binary Numbers

H_1 with n_1 NODES : $H_1 =$

H_2 with n_2 NODES : $H_2 =$

$$\begin{array}{lllll} & & & & \;\;5\;4\;3\;2\;1\;0 \\ \text{ex:} & n_1 = 39 & : H_1 = & < 1\;0\;0\;1\;1\;1 > = & \{ B_0, B_1, B_2, B_5 \} \\ & n_2 = 54 & : H_2 = & < 1\;1\;0\;1\;1\;0 > = & \{ B_1, B_2, B_4, B_5 \} \end{array}$$

MERGE



CASE1 MARCH
Cin=0
1+0=1

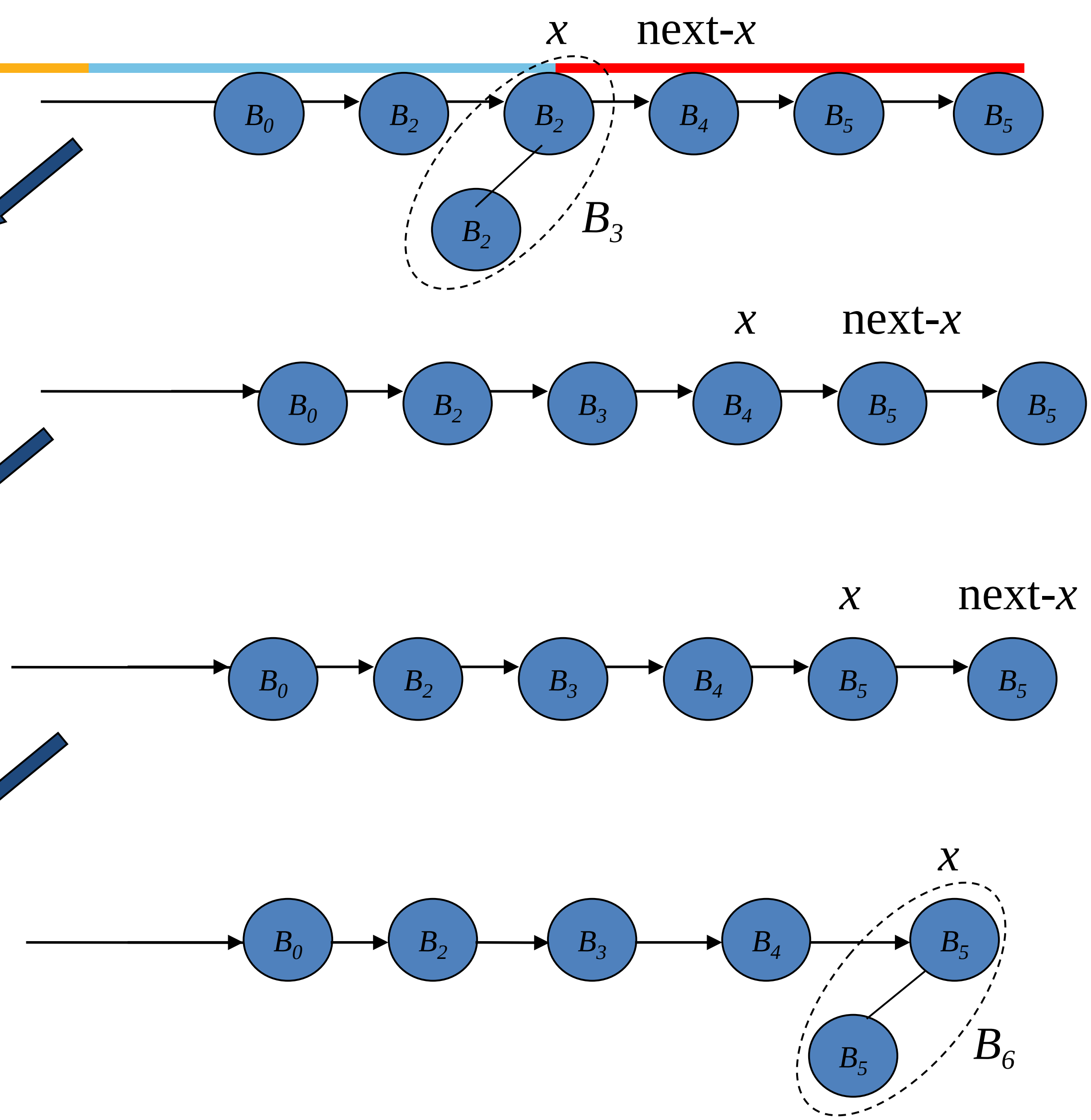
CASE3 or 4
LINK Cin=0
1+1=10

CASE2 MARCH
then CASE3 and
CASE4 LINK
Cin=1
1+1=11

CASE1
MARCH
Cin=1
0+0=1

CASE1
MARCH
Cin=0
0+1=1

CASE3 OR 4
LINK Cin=0
1+0=10



Inserting a Node

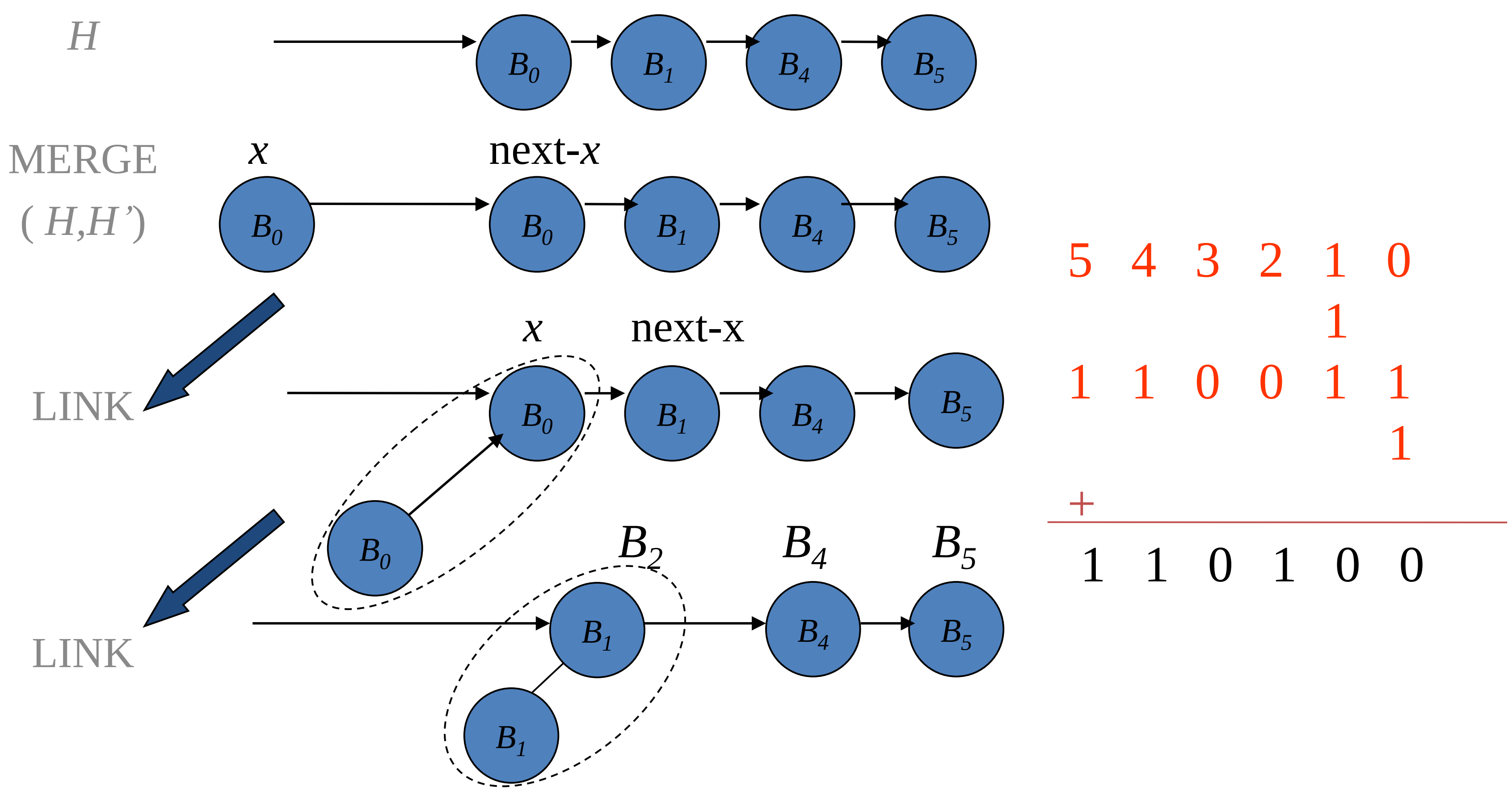
BINOMIAL-HEAP-INSERT (H, x)

$H' \leftarrow$ MAKE-BINOMIAL-HEAP (H, x)
$P[x] \leftarrow$ NIL
child $[x] \leftarrow$ NIL
sibling $[x] \leftarrow$ NIL
degree $[x] \leftarrow$ O
head $[H'] \leftarrow x$
$H \leftarrow$ BINOMIAL-HEAP-UNION (H, H')
end

RUNNING-TIME= O(lg n)

Relationship Between Insertion & Incrementing a Binary Number

$H : n_1 = 51 \qquad H = <110011> \quad = \quad \{ B_0, B_1, B_4, B_5 \}$



Extracting the Node with the Minimum Key

BINOMIAL-HEAP-EXTRACT-MIN (*H*)

(1) find the root *x* with the minimum key in the root list of *H* and remove *x* from the root list of *H*

(2) *H'* ← MAKE-BINOMIAL-HEAP ( )

(3) reverse the order of the linked list of *x*' children and set head [*H*'] ← head of the resulting list

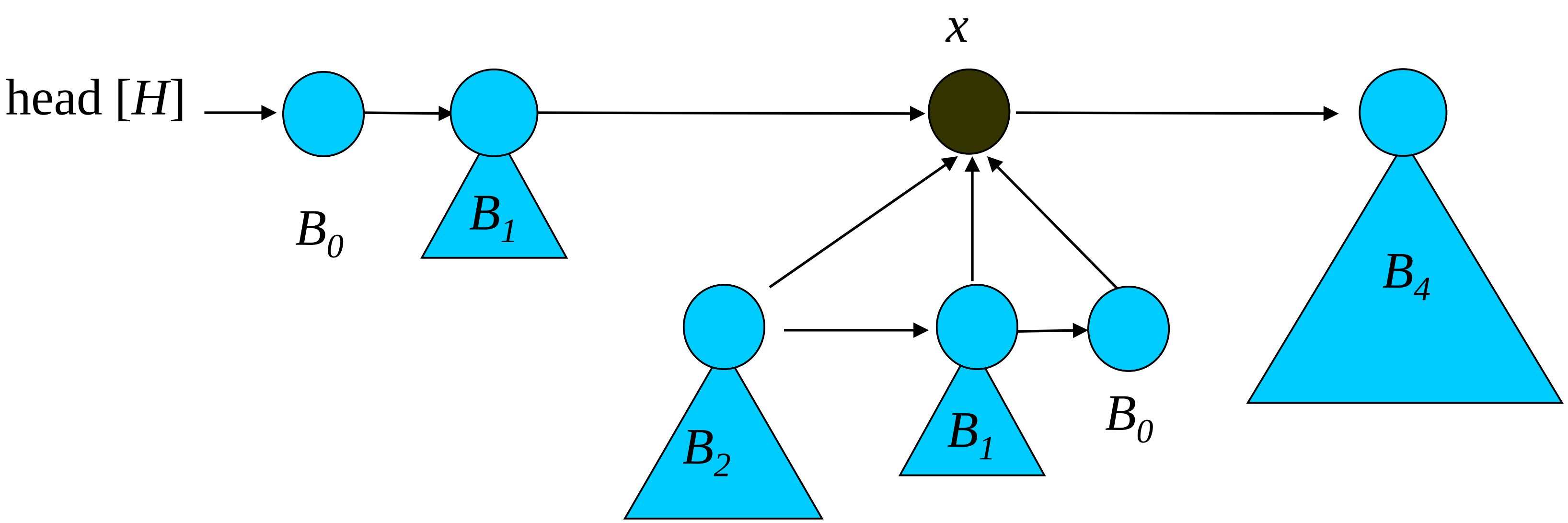
(4) *H* ← BINOMIAL-HEAP-UNION (*H*, *H'*)

return *x*

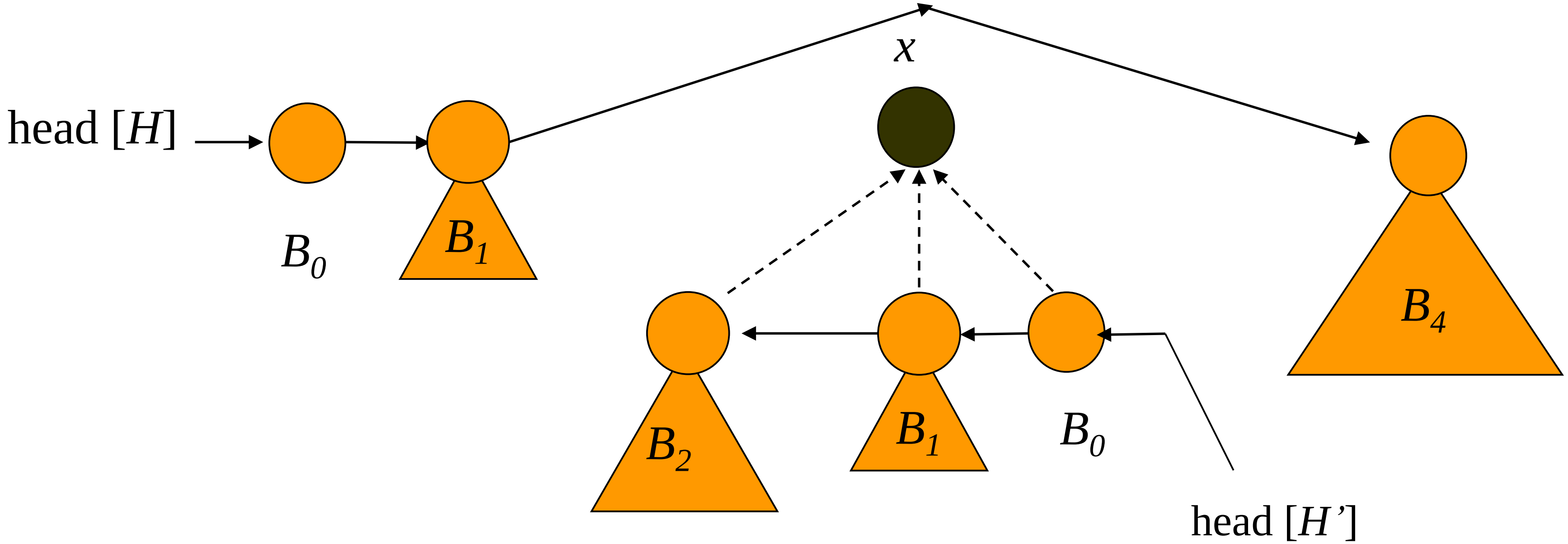
end

Extracting the Node with the Minimum Key

Consider H with $n = 27$, $H = \langle 1\ 1\ 0\ 1\ 1 \rangle = \{B_0, B_1, B_3, B_4\}$
assume that x = root of B_3 is the root with minimum key



Extracting the Node with the Minimum Key



Extracting the Node with the Minimum Key

- Unite binomial heaps $H = \{B_0, B_1, B_4\}$ and $H' = \{B_0, B_1, B_2\}$
- Running time if H has n nodes
- Each of lines 1-4 takes O(lg$n$) time

  it is O(lg$n$).

Decreasing a Key

BINOMIAL-HEAP-DECREASE-KEY (H, x, k)

key [x] $\leftarrow$ k

y $\leftarrow$ x

z $\leftarrow$ p[y]

while z $\neq$ NIL and key [y] < key [z] do

    exchange key [y] $\leftarrow$ key [z]

    exchange satellite fields of y and z

    y $\leftarrow$ z

    z $\leftarrow$ p [y]

    endwhile

end

Decreasing a Key

- Similar to DECREASE-KEY in BINARY HEAP
- BUBBLE-UP the key in the binomial tree it resides in
- RUNNING TIME: O(lg*n*)

Deleting a Key

BINOMIAL- HEAP- DELETE (H,x)

$y \leftarrow x$

$z \leftarrow$ p $[y]$

while $z \neq$ NIL do

key $[y] \leftarrow$ key $[z]$

satellite field of y $\leftarrow$ satellite field of z

$y \leftarrow z$; $z \leftarrow$ p $[y]$

endwhile

$H' \leftarrow$ MAKE-BINOMIAL-HEAP

remove root z from the root list of H

reverse the order of the linked list of z's children

and set head $[H'] \leftarrow$ head of the resulting list

$H \leftarrow$ BINOMIAL-HEAP-UNION (H, H')

RUNNING-TIME= O(lg n)

Thank You