



中国科学技术大学 计算机科学与技术系

University of Science and Technology of China

DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

算法基础

Foundation of Algorithms

主讲人 徐云

Fall 2018, USTC



Part 1 Foundation

Part 2 Sorting and Order Statistics

Part 3 Data Structure

Part 4 Advanced Design and Analysis Techniques

Part 5 Advanced Data Structures

chap 18 B-Tree

chap 19 Fibonacci Heaps (Binomial Heaps in v2)

chap 20 Van Emde Boas Trees

chap 21 Data Structures for Disjoint Sets

Part 6 Graph Algorithms

Part 7 Selected Topics

Part 8 Supplement



Chapter 19 Binomial Heap (二项堆, in v2)

19.1 Priority Queue and Union op

19.2 Binomial Trees and Binomial Heap

19.3 Operations on a Binomial Heap

19.1 Priority Queue and Union op

- Priority queue
- Various implementations
- Comparison of efficiency
- Union operation

Priority Queue

- *Priority Queue* is an ADT (抽象数据类型) for maintaining a set S of elements, each with a *key* value and supports the following operations:

- $\text{INSERT}(S, x)$ *inserts element x into S (also write as $S \leftarrow S \cup \{x\}$)*
- $\text{MINIMUM}(S)$ *returns element in S with *min* key*
- $\text{EXTRACT-MIN}(S)$ *removes and returns element in S with *min* key*
- $\text{DECREASE-KEY}(S, x, k)$ *decreases the value of element x 's key to a new value k*

PQ Implementations...

- Many data structures proposed for PQ:

1964	Binary Heap	<i>J. W. J. Williams</i>
1972	Leftist Heap	<i>C. A. Crane</i>
1978	Binomial Heap	<i>J. Vuillemin</i>
1984	Fibonacci Heap	<i>M. L. Fredman, R. E. Tarjan</i>
1985	Skew Heap	<i>D. D. Sleator R. E. Tarjan</i>
1988	Relaxed Heap	<i>Driscoll, Gabow Shrairman, Tarjan</i>

Binary Min-Heap (as in Heapsort)

- *Binary min-heap* is an array $A[1..n]$ that can be viewed as a nearly complete *binary tree*.
- Number the nodes using level order traversal.
 - $\text{LEFT}(i) = 2i$ and $\text{RIGHT}(i) = 2i+1$ and
 - $\text{PARENT}(i) = \lfloor i/2 \rfloor$
 - Height of tree $\approx \log n$
- Heap Property: (Each node \geq its parent node)
 - $A[\text{PARENT}(i)] \leq A[i]$

PQ Implementations...

- Time Bounds for different PQ implementations.
 - n is the number of items in the PQ.

Data Str	INSERT	MIN	Extract -MIN	D-KEY	DELETE	Union
Binary H	$O(\lg n)$	$O(1)$	$O(\lg n)$	$O(\lg n)$	$O(\lg n)$	$O(n)$
Binomial H	$O(\lg n)$	$O(\lg n)$	$O(\lg n)$	$O(\lg n)$	$O(\lg n)$	$O(\lg n)$
Fibonacci	$O(1)$	$O(1)$	$O(\lg n)$	$O(1)$	$O(\lg n)$	$O(1)$

Comparison of Efficiency

Procedure	Binary (worst-case)	Binomial (worst-case)	Fibonacci (amortized)
Make-Heap	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$
Insert	$\Theta(\lg n)$	$O(\lg n)$	$\Theta(1)$
Minimum	$\Theta(1)$	$O(\lg n)$	$\Theta(1)$
Extract-Min	$\Theta(\lg n)$	$\Theta(\lg n)$	$O(\lg n)$
Union	$\Theta(n)$	$O(\lg n)$	$\Theta(1)$
Decrease-Key	$\Theta(\lg n)$	$\Theta(\lg n)$	$\Theta(1)$
Delete	$\Theta(\lg n)$	$\Theta(\lg n)$	$O(\lg n)$

Union Operation

- A *mergeable heap* (可合并堆) is any data structure that supports the basic heap operation *plus union*.
- Union (H_1, H_2) creates and returns a new heap.



Chapter 19 Binomial Heap (二项堆, in v2)

19.1 Priority Queue and Union op

19.2 Binomial Trees and Binomial Heap

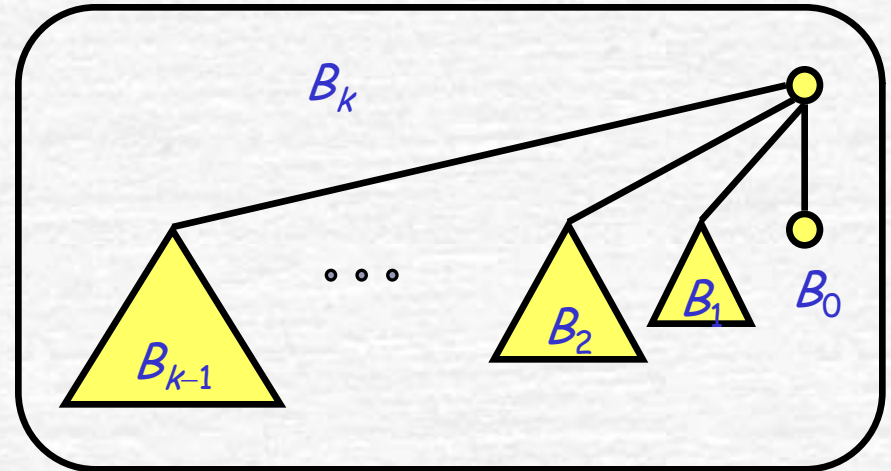
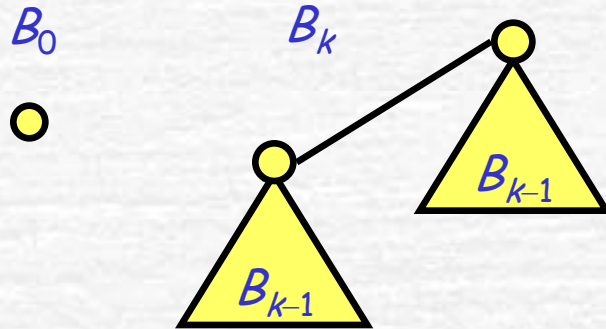
19.3 Operations on a Binomial Heap

19.2 Binomial Trees and Binomial Heap

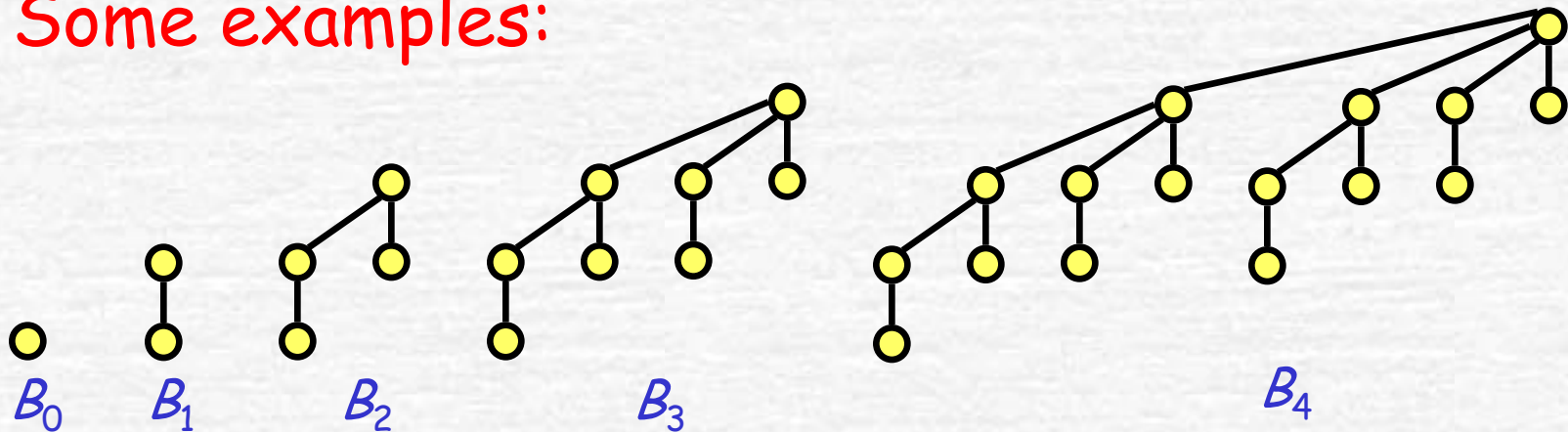
- Binomial trees (二项树)
- Properties of binomial trees
- Binomial heaps
- Representing binomial heaps

Binomial Trees

Recursive definition:



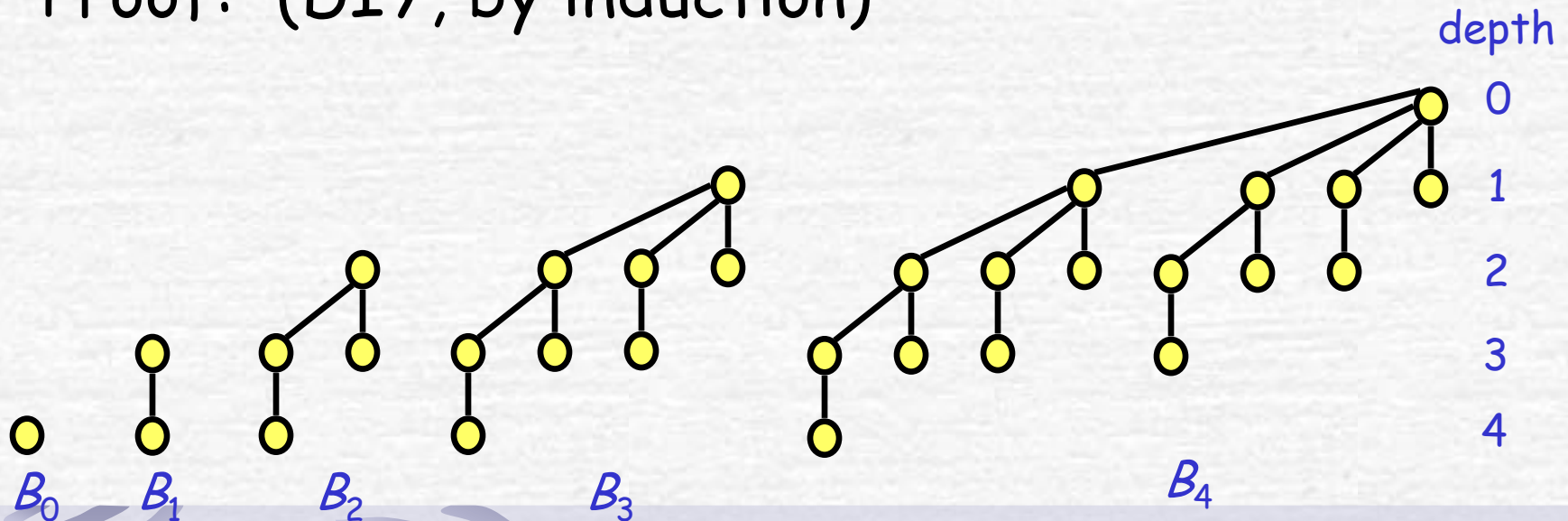
Some examples:



Properties of Binomial Trees

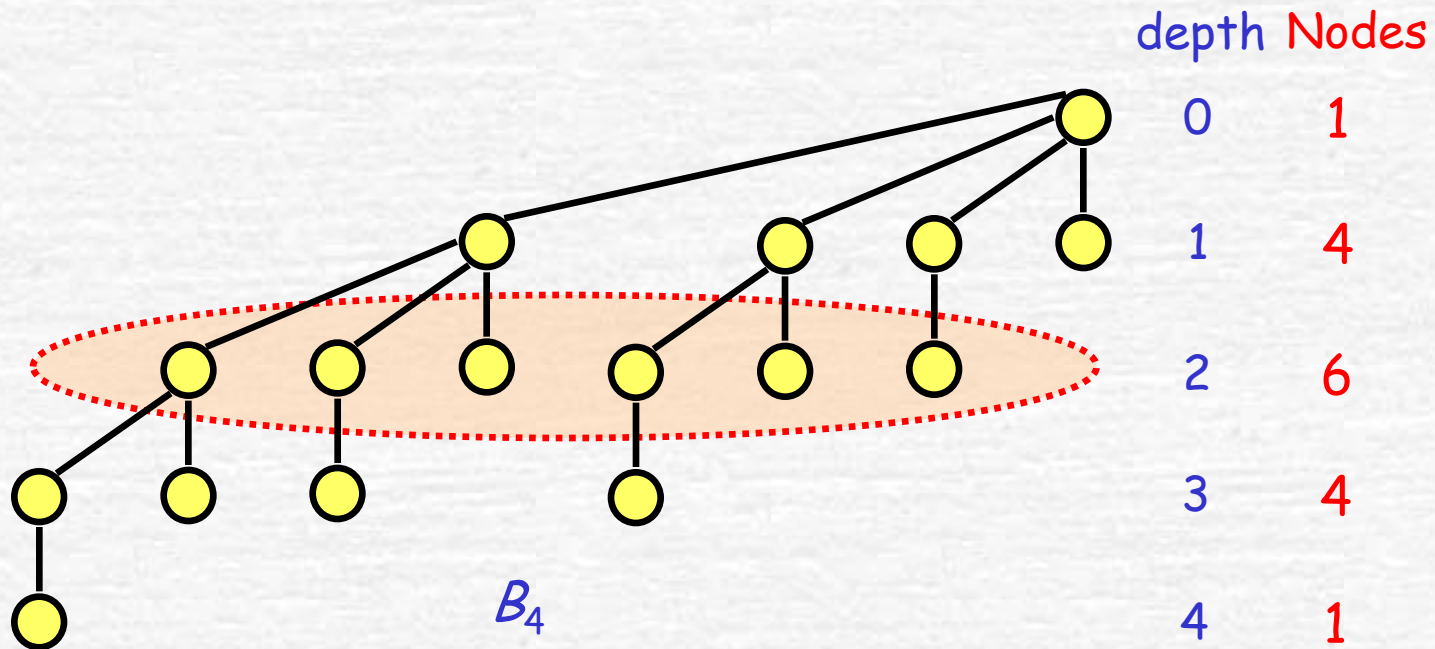
- For a Binomial Tree B_k (of order k)
 1. there are 2^k nodes,
 2. the height of the tree is k ,
 3. root has degree k and
 4. deleting the root gives binomial trees B_0, B_1, \dots, B_{k-1} .

Proof: (DIY, by induction)



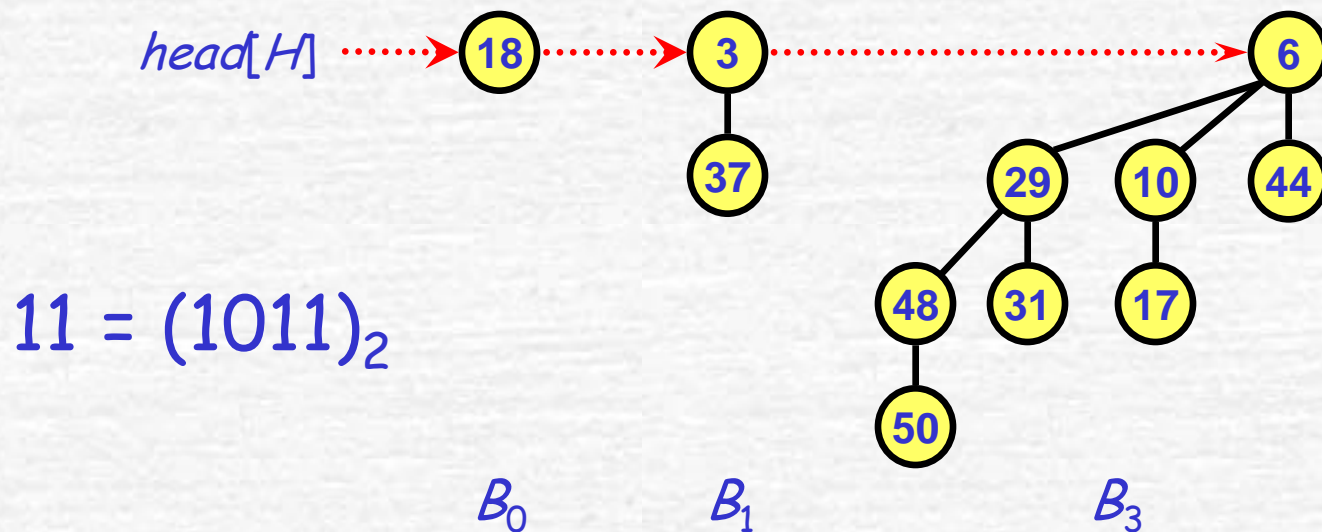
Defining Property of Binomial Trees

There are exactly $\binom{k}{i}$ nodes at depth i , for B_k
($0 \leq i \leq k$)



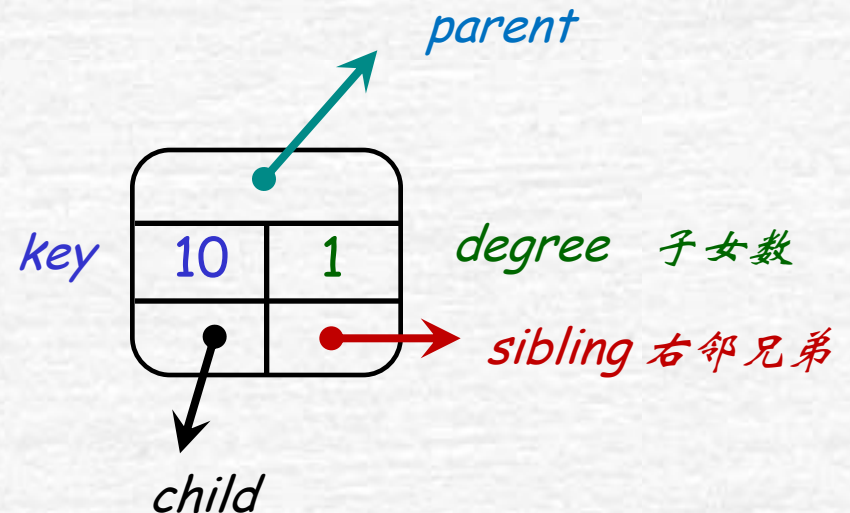
Binomial Heap (Vuillemin, 1978)

- A sequence of binomial trees that satisfy
 - ▣ binomial heap property (each tree B_k is a min-heap)
 - ▣ 0 or 1 binomial tree B_k of order k ,
- There are at most $\lfloor \log n \rfloor + 1$ binomial trees.
- Eg: A binomial heap H with $n = 11$ nodes.

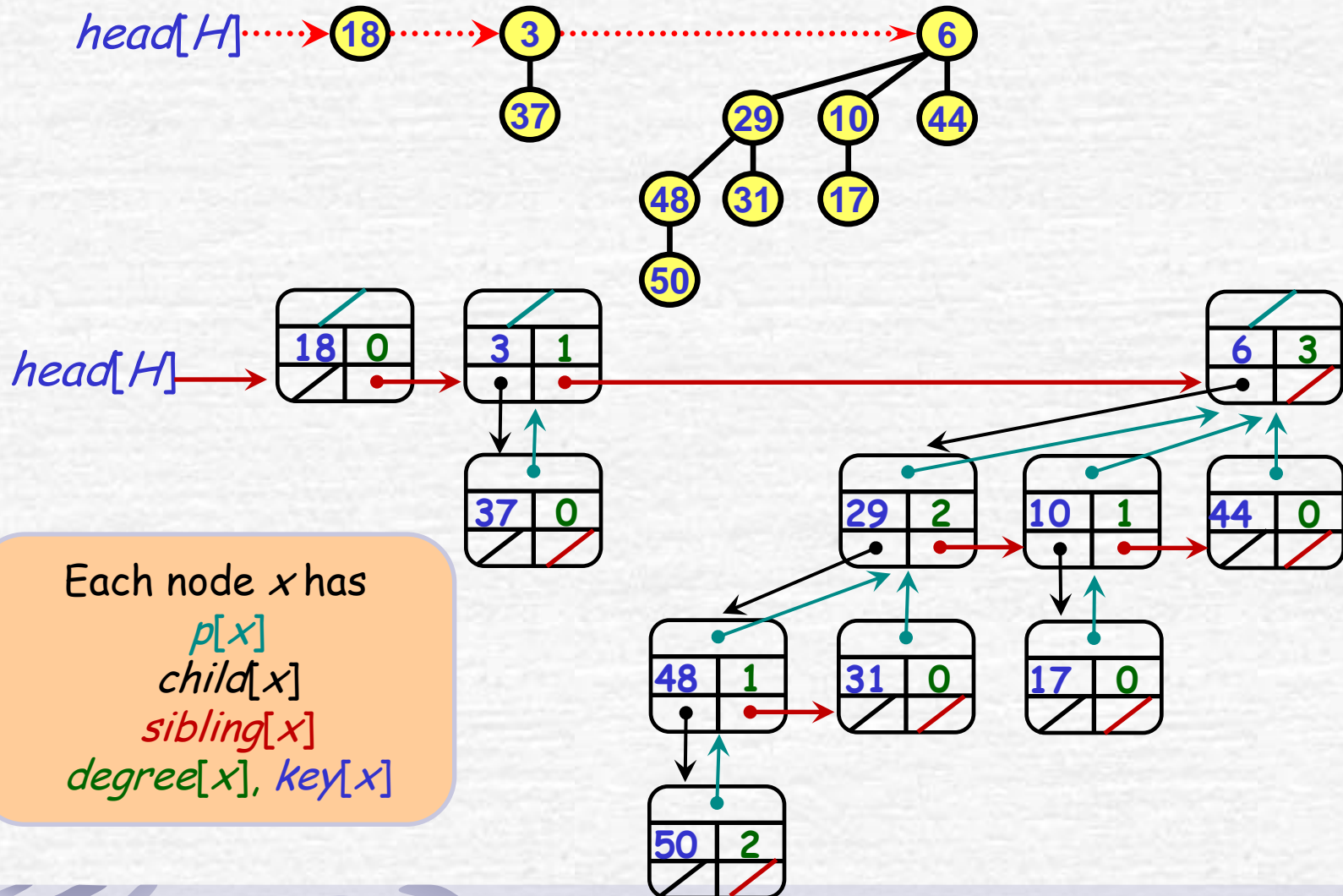


Representing Binomial Heaps (1)

- Each node x stores
 - $key[x]$
 - $degree[x]$
 - $p[x]$
 - $child[x]$
 - $sibling[x]$
- (3 pointers per node)



Representing Binomial Heaps (2)





Chapter 19 Binomial Heap (二项堆, in v2)

19.1 Priority Queue and Union op

19.2 Binomial Trees and Binomial Heap

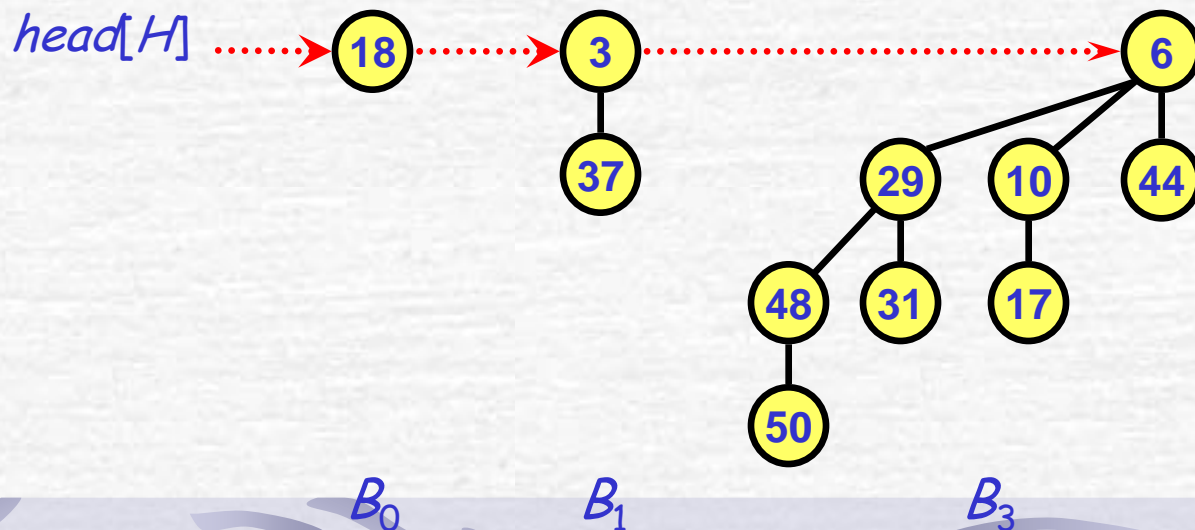
19.3 Operations on a Binomial Heap

19.3 Operations on a Binomial Heap

- MAKE and MINIMUM
- Linking Step: Fundamental Op
- Binomial Heap Union
- More Operations
- Summary

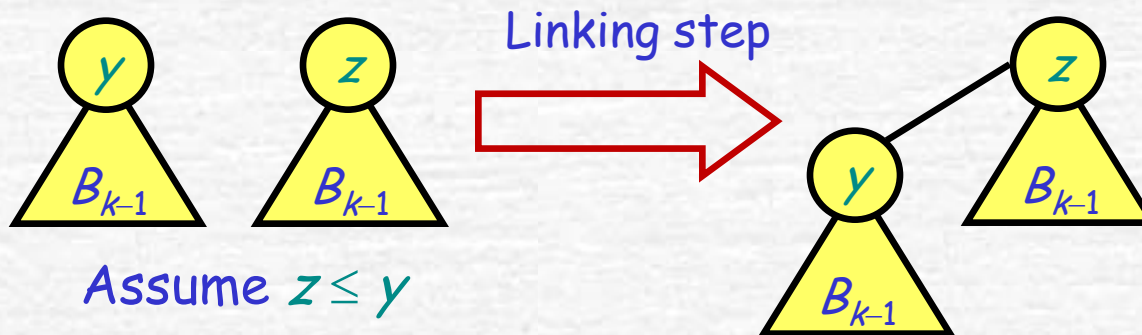
MAKE and MINIMUM

- MAKE-BINOMIAL-HEAP(H)
 - ▣ Allocate object H , make $head[H] = \text{NIL}$. $\Theta(1)$.
- BINOMIAL-HEAP-MINIMUM(H)
 - ▣ Search the root list for minimum. $\mathcal{O}(\log n)$.



Linking Step: Fundamental Op

- BINOMIAL-LINK (y, z)



BINOMIAL-LINK (y, z) ▷ Assume $z \leq y$

$p[y] \leftarrow z$

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

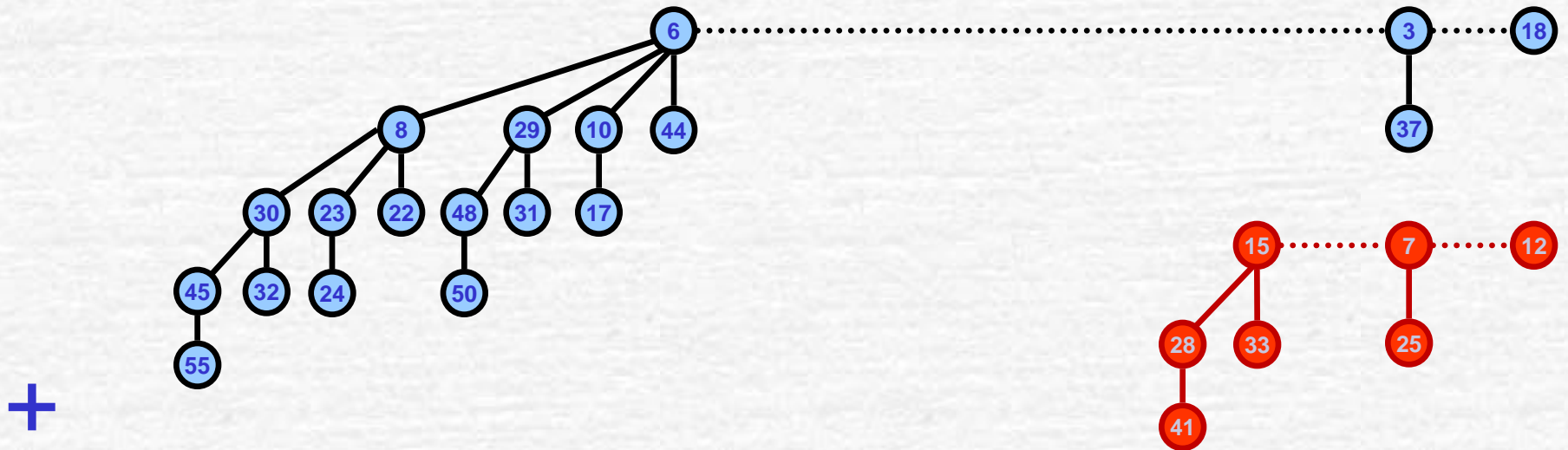
$child[z] \leftarrow y$

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

Constant time $O(1)$

Binomial Heap Union (1)

Let us look at the procedure of an example:



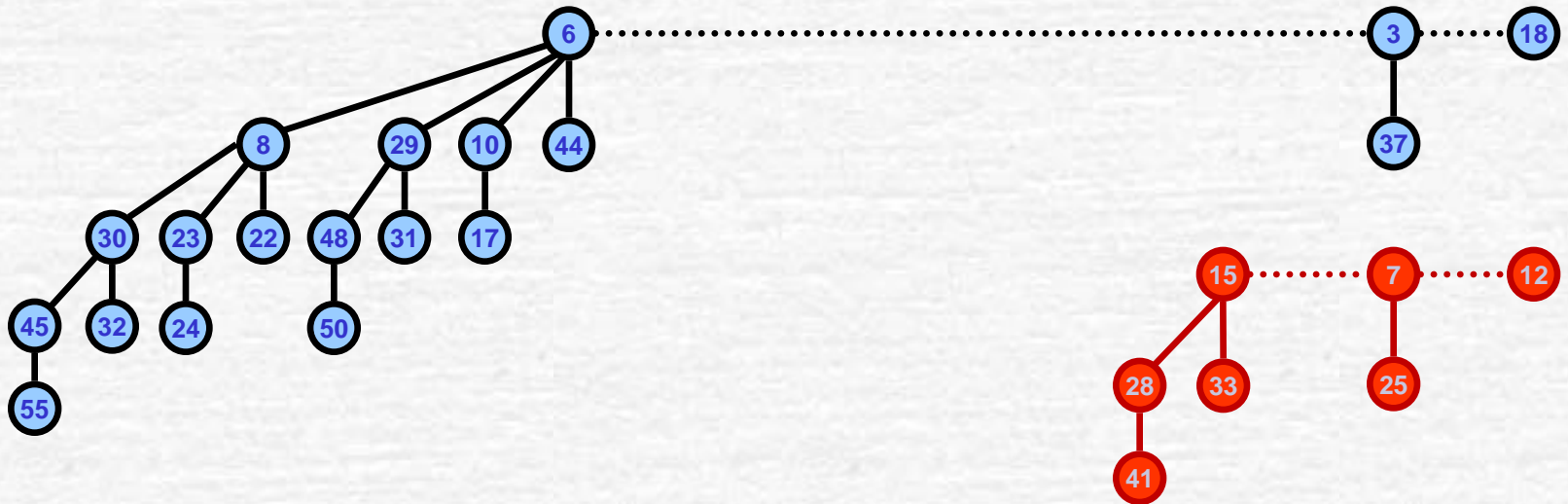
$$19 + 7 = 26$$

The binomial trees in the Binomial Heap at last: B_1, B_3, B_4

$$\begin{array}{r}
 \\
 \\
 \\
 + \\
 \hline
 1 1 1 0
 \end{array}$$

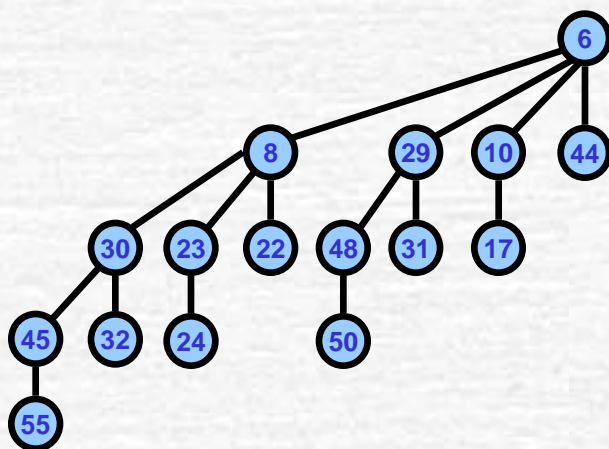
Binomial Heap Union

Temporary area:

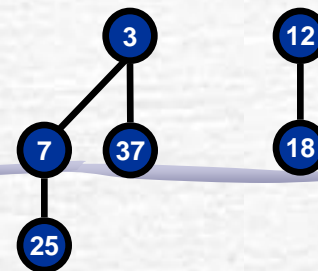


Stable area:

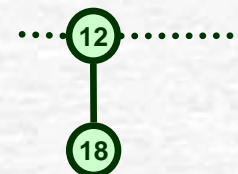
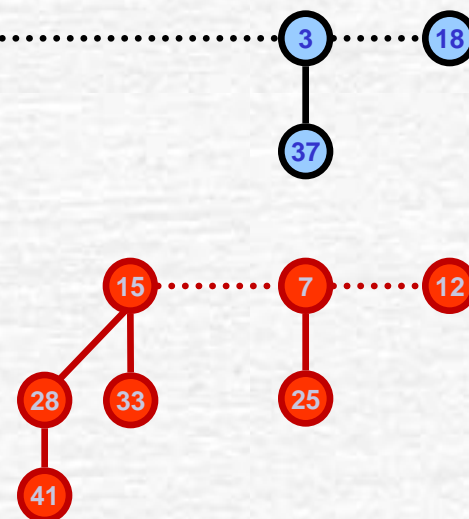
Temporary area:



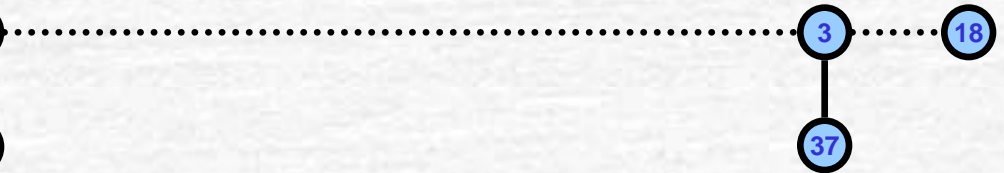
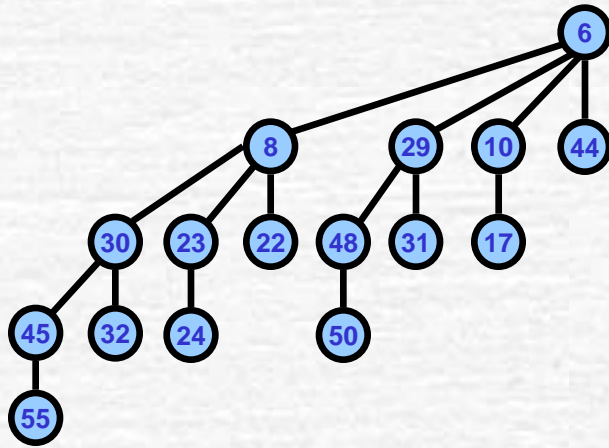
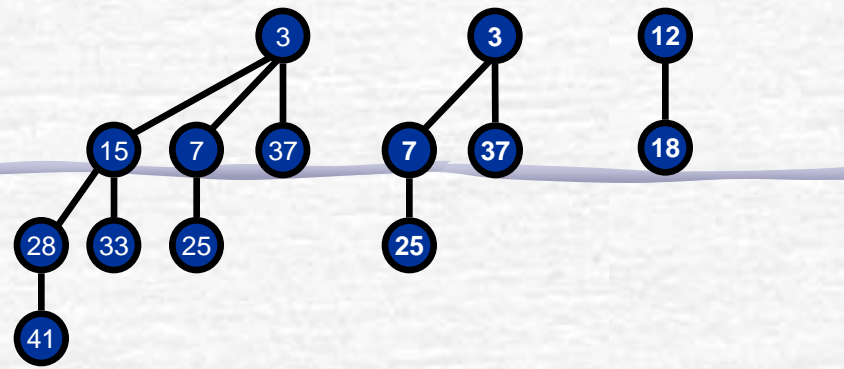
+



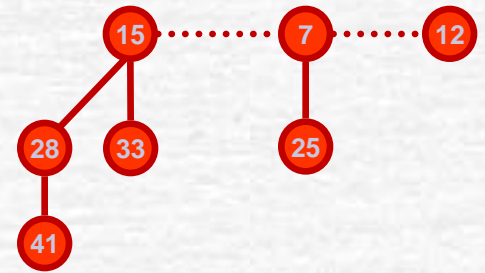
Stable area:



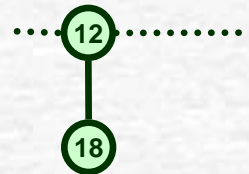
Temporary area:



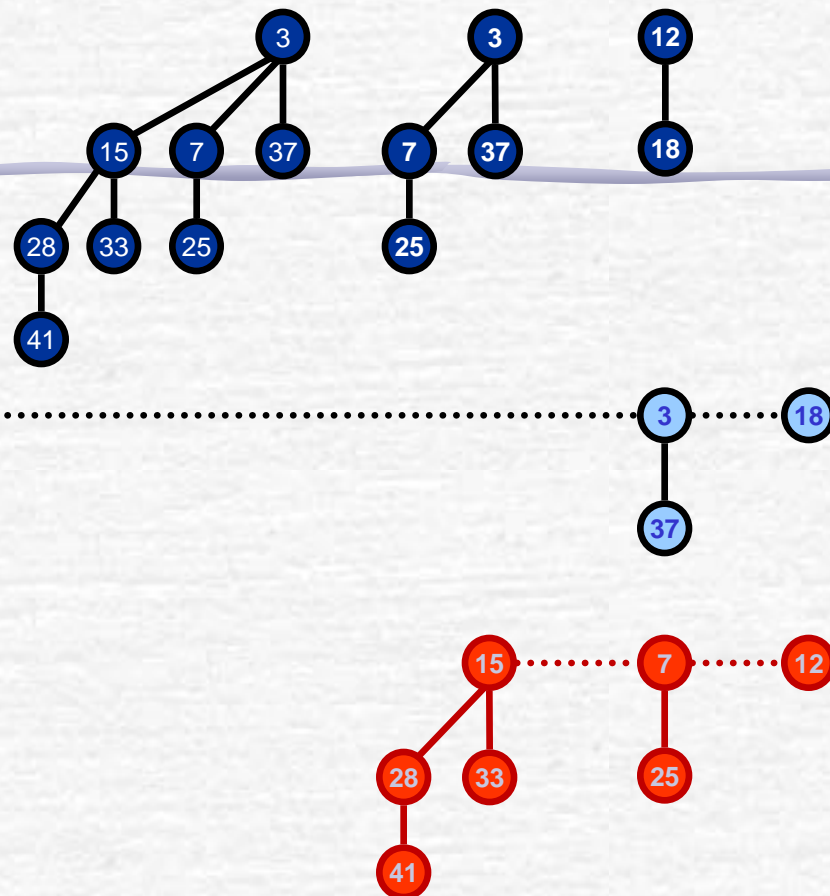
+



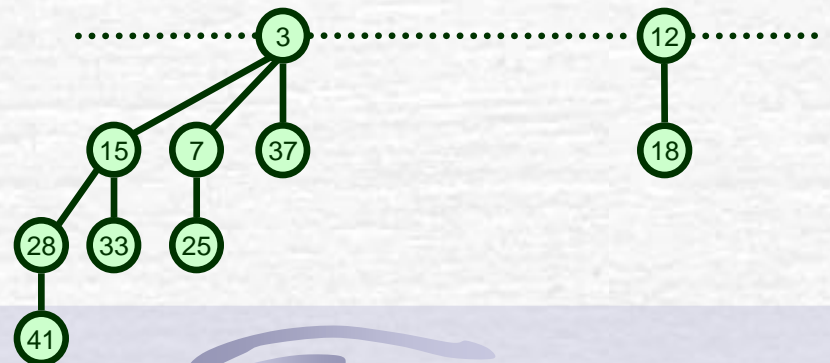
Stable area:



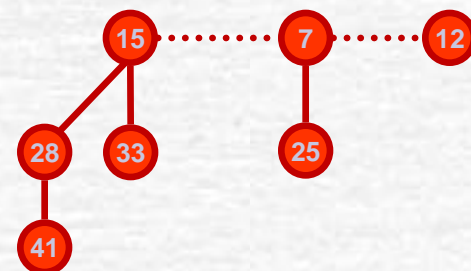
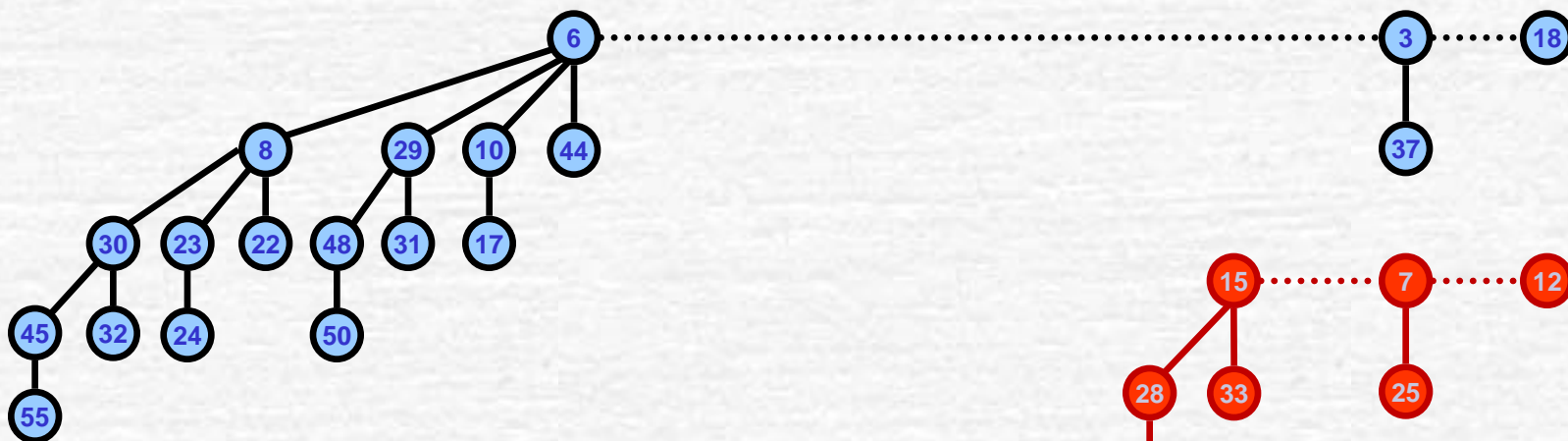
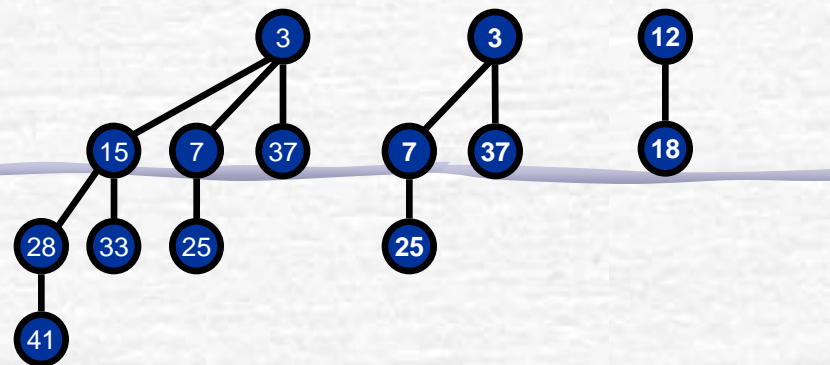
Temporary area:



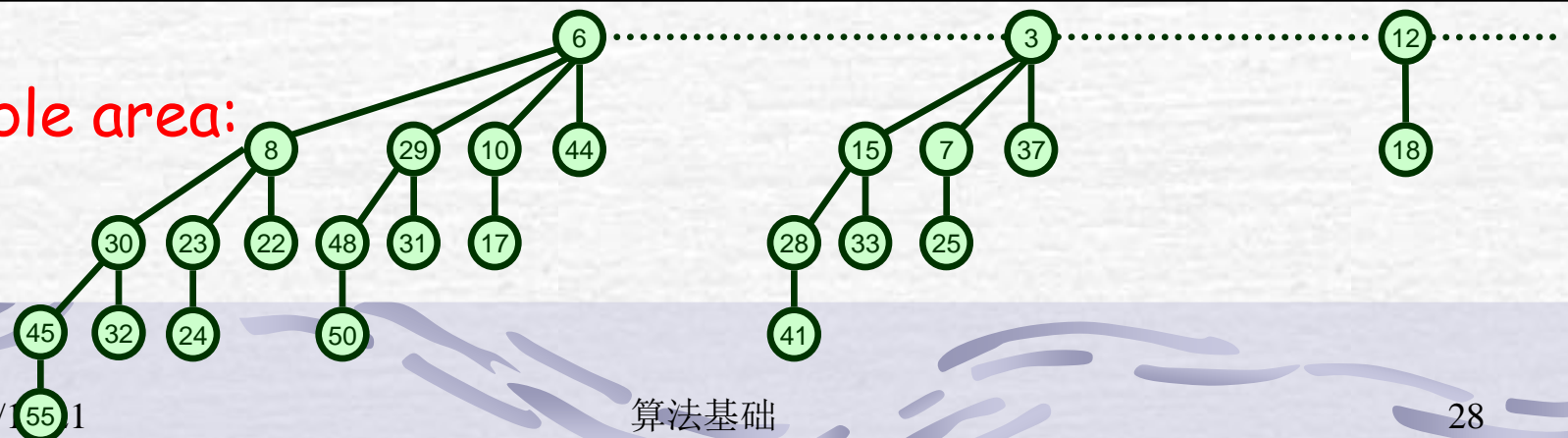
Stable area:



Temporary area:



Stable area:



BINOMIAL-HEAP-UNION(H_1, H_2)

```
1   $H \leftarrow \text{MAKE-BINOMIAL-HEAP}()$ 
2   $\text{head}[H] \leftarrow \text{BINOMIAL-HEAP-MERGE}(H_1, H_2)$ 
3  free the objects  $H_1$  and  $H_2$  but not the lists they point to
4  if  $\text{head}[H] = \text{NIL}$ 
5      then return  $H$ 
6   $\text{prev-}x \leftarrow \text{NIL}$ 
7   $x \leftarrow \text{head}[H]$ 
8   $\text{next-}x \leftarrow \text{sibling}[x]$ 
9  while  $\text{next-}x \neq \text{NIL}$ 
10     do if ( $\text{degree}[x] \neq \text{degree}[\text{next-}x]$ ) or
           ( $\text{sibling}[\text{next-}x] \neq \text{NIL}$  and  $\text{degree}[\text{sibling}[\text{next-}x]] = \text{degree}[x]$ )
11         then  $\text{prev-}x \leftarrow x$                                 ▷ Cases 1 and 2
12              $x \leftarrow \text{next-}x$                                 ▷ Cases 1 and 2
13     else if  $\text{key}[x] \leq \text{key}[\text{next-}x]$ 
14         then  $\text{sibling}[x] \leftarrow \text{sibling}[\text{next-}x]$           ▷ Case 3
15              $\text{BINOMIAL-LINK}(\text{next-}x, x)$                         ▷ Case 3
16     else if  $\text{prev-}x = \text{NIL}$                                      ▷ Case 4
17         then  $\text{head}[H] \leftarrow \text{next-}x$                        ▷ Case 4
18         else  $\text{sibling}[\text{prev-}x] \leftarrow \text{next-}x$              ▷ Case 4
19              $\text{BINOMIAL-LINK}(x, \text{next-}x)$                         ▷ Case 4
20              $x \leftarrow \text{next-}x$                                 ▷ Case 4
21      $\text{next-}x \leftarrow \text{sibling}[x]$ 
22 return  $H$ 
```

Binomial Heap Union (3)

Case classification :

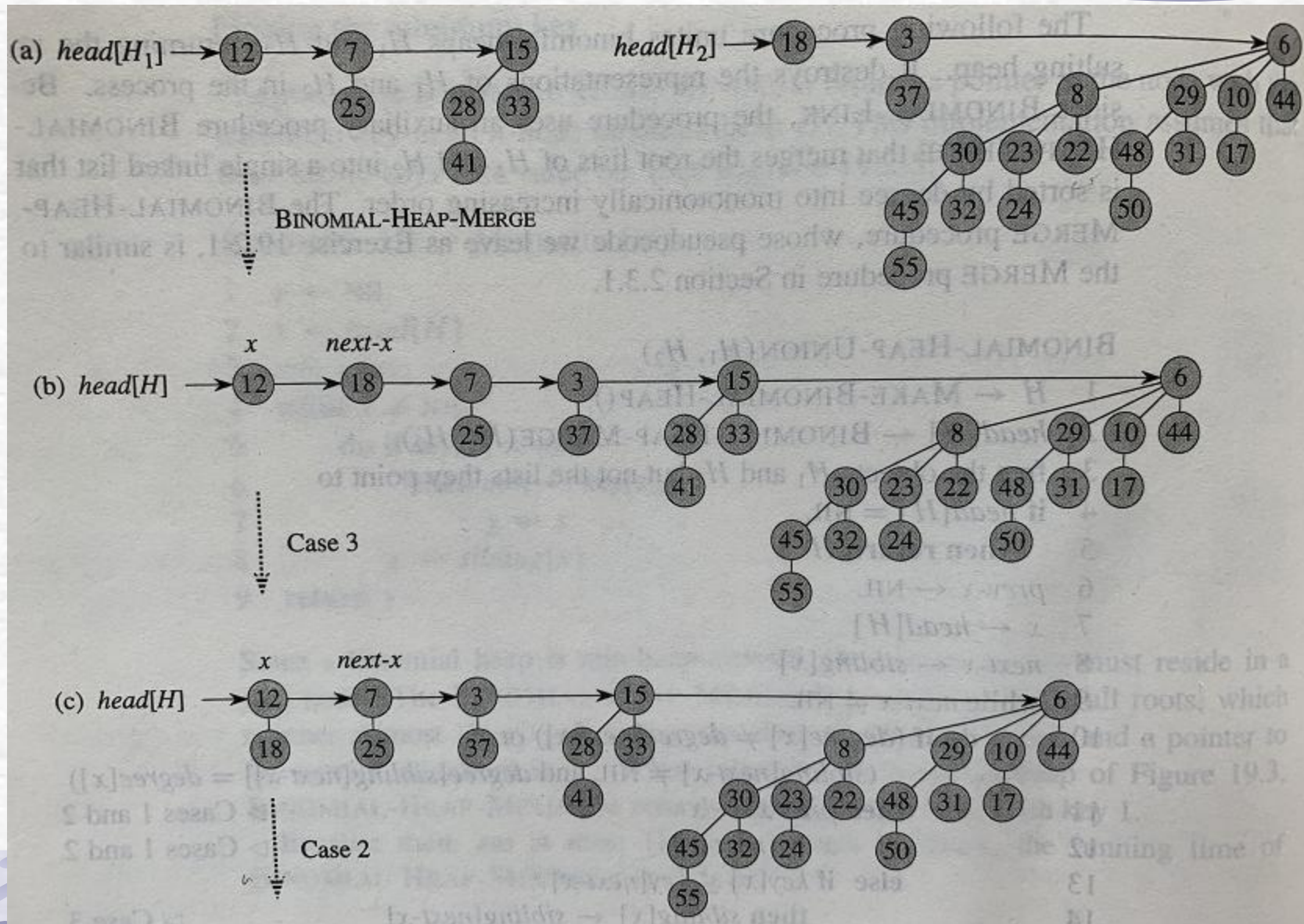
$\neq \text{degree}[\text{next-}x]$ case 1

$\text{degree}[x] = \text{degree}[\text{sibling}[\text{next-}x]]$ case 2

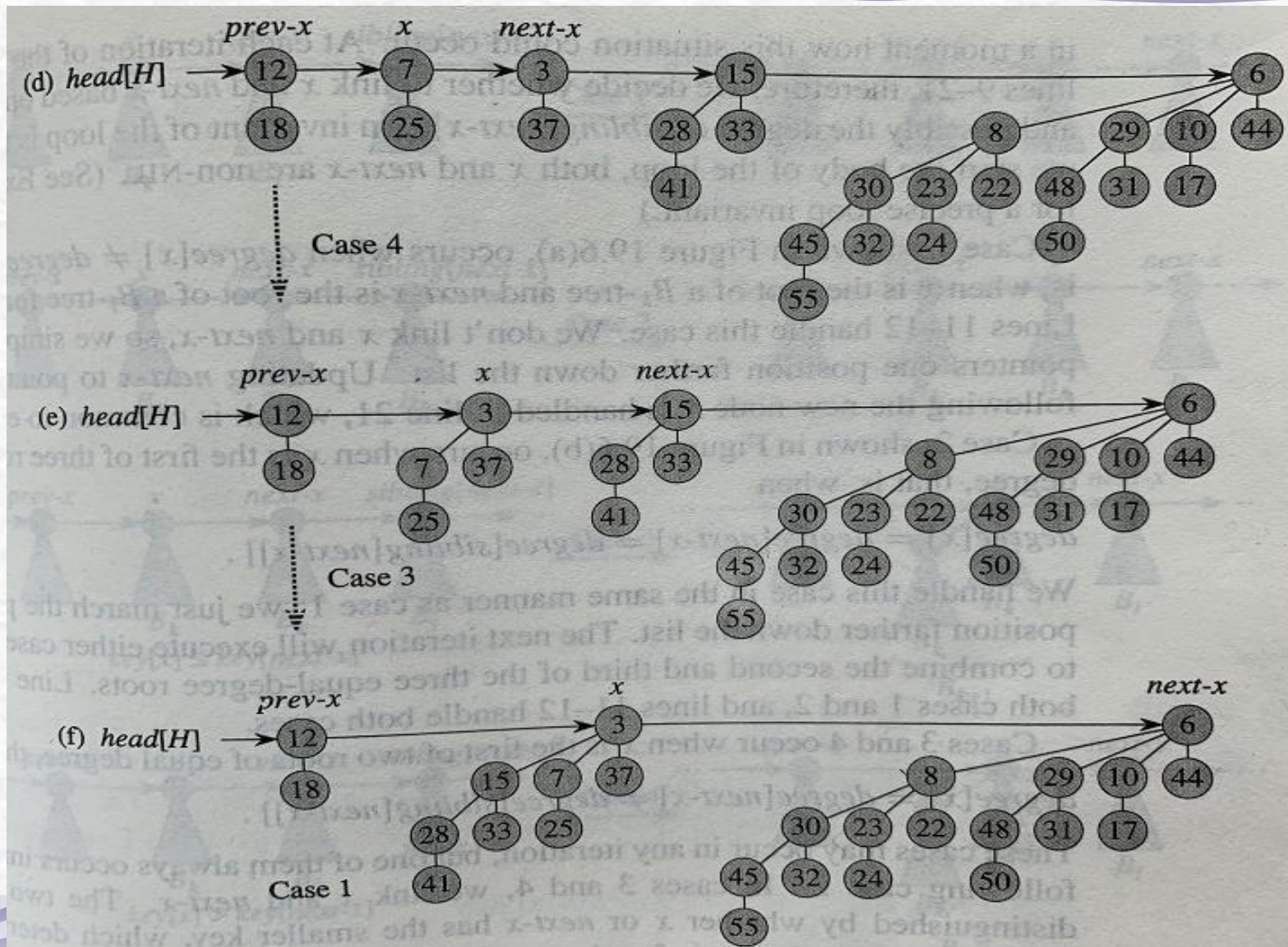
$= \text{degree}[\text{next-}x]$ $\text{key}[x] \leq \text{key}[\text{next-}x]$ case 3

$\neq \text{degree}[\text{sibling}[\text{next-}x]]$ and $\text{key}[x] > \text{key}[\text{next-}x]$ case 4

Binomial Heap Union (4)



Binomial Heap Union (5)



Binomial Heap Union (6)

- MAKE-BINOMIAL-HEAP-UNION (H_1, H_2):
 - ▣ Create a heap H that is the union of two heaps H_1 and H_2
 - ▣ Analogous to binary addition of n_1 and n_2
- Running time.: $O(\log n)$ [$n = n_1 + n_2$]

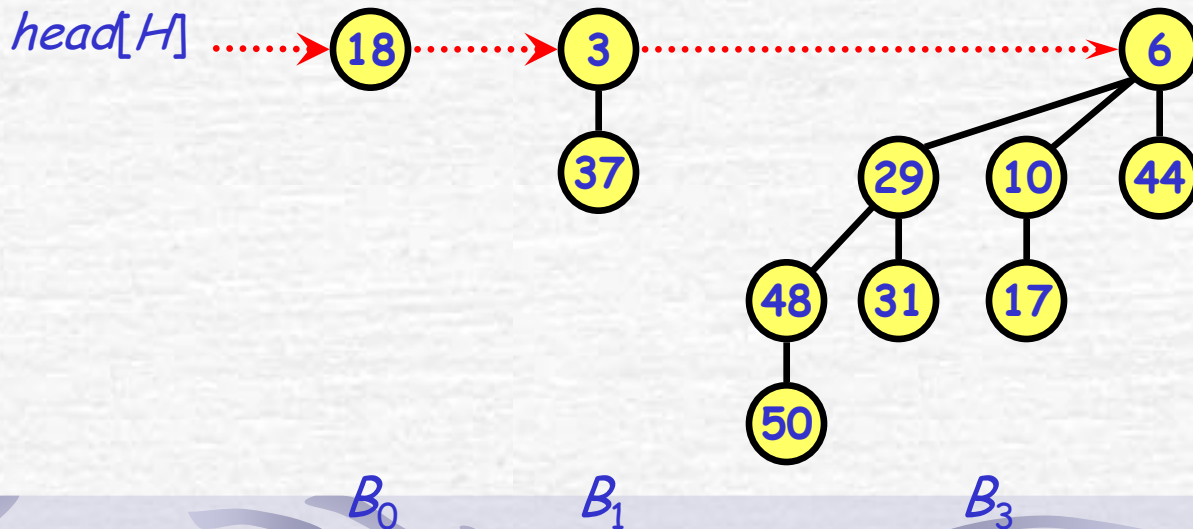
$$19 + 7 = 26$$

$$\begin{array}{r}
 \\
 \\
 \\
 + \\
 \hline

 \end{array}$$

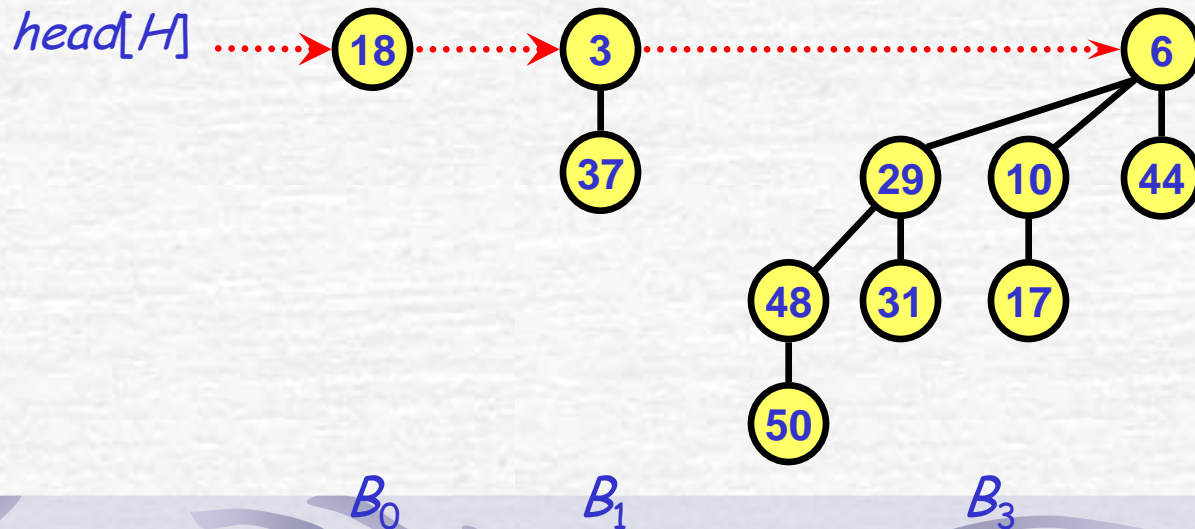
More Operations (1)

- **BINOMIAL-HEAP-INSERT(H, x)**
 - ▣ Create a one-item (x) binomial heap H_1 and then union H and H_1 . $\mathcal{O}(\lg n)$.
- **BINOMIAL-HEAP-EXTRACT-MIN(H)**
 - ▣ Find minimum, remove root, then union. $\mathcal{O}(\lg n)$.



More Operations (2)

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



Summary

- $\text{MINIMUM}(H)$ $O(\lg n)$
- $\text{UNION}(H_1, H_2)$ $O(\lg n)$
- $\text{INSERT}(H, x)$ $O(\lg n)$
- $\text{EXTRACT-MIN}(H)$ $O(\lg n)$
- $\text{DECREASE-KEY}(H, x, k)$ $O(\lg n)$
- $\text{DELETE}(H, x)$ $O(\lg n)$



End of Ch19