算法基础
Foundation of Algorithms

主讲人 徐云

Fall 2019, USTC

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

The slides are from Prof. LeongHonWai@NUS and taoli@FIU

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:

  - INSERT(S, x) — *inserts* element x into S (also write as $S \leftarrow S \cup \{x\}$)
  - MINIMUM(S) — returns element in S with *min* key
  - EXTRACT-MIN(S) — removes and returns element in S with *min* key
  - 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	*J. W. J. Williams*
1972	Leftist Heap	*C. A. Crane*
1978	Binomial Heap	*J. Vuillemin*
1984	Fibonacci Heap	*M. L. Fredman, R. E. Tarjan*
1985	Skew Heap	*D. D. Sleator R. E. Tarjan*
1988	Relaxed Heap	*Driscoll, Gabow Shrairman, Tarjan*

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.
  - LEFT(i) = $2i$ and RIGHT(i) = $2i+1$ and
  - PARENT(i) = $\lfloor i/2 \rfloor$
  - Height of tree $\approx \log n$
- Heap Property: (Each node ≥ 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 (H1, H2) creates and returns a new heap.

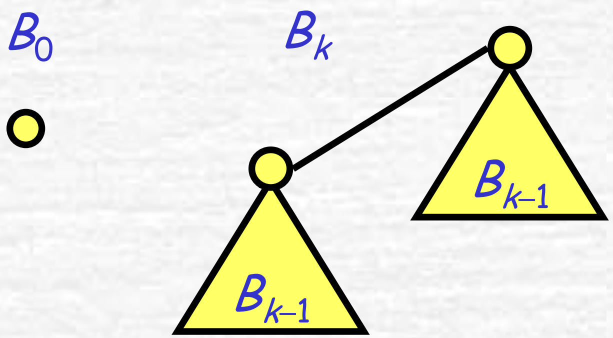
Chapter 19 Binomial Heap (二项堆, in v2)

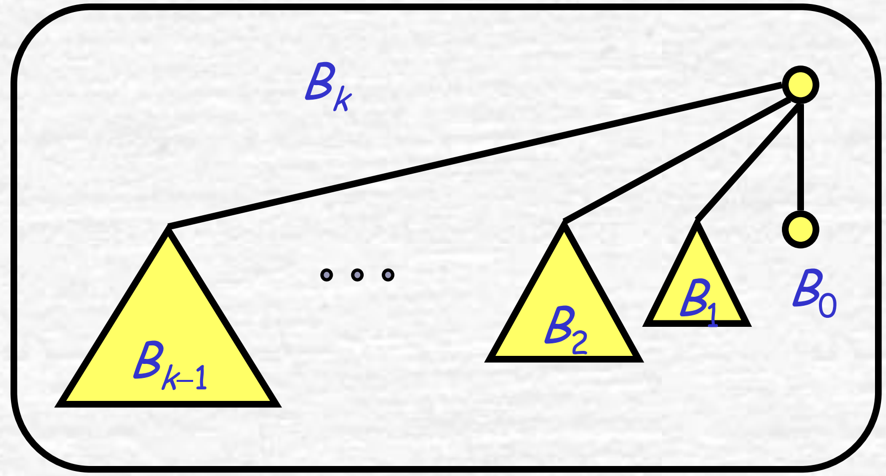
19.2 Binomial Trees and Binomial Heap

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

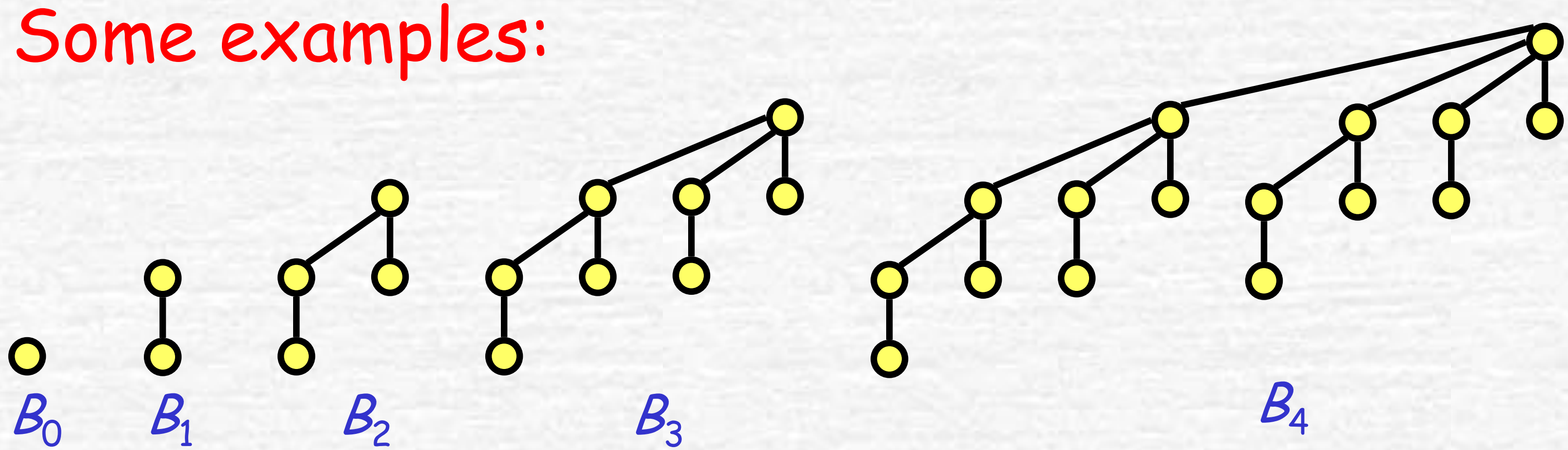
Binomial Trees

Recursive definition:





Some examples:

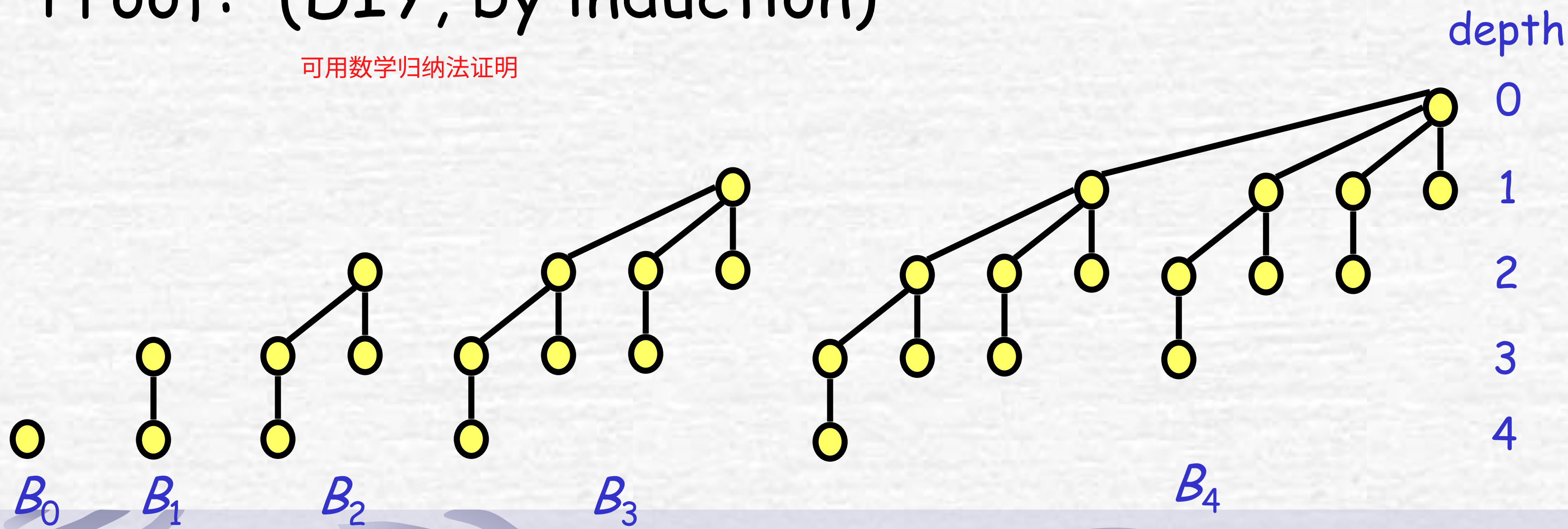


Properties of Binomial Trees

- For a Binomial Tree B_k (of order k)
  1. there are 2^k nodes,
  2. the height of the tree is k, 从0开始记
  3. root has degree k and
  4. deleting the root gives binomial trees $B_0, B_1, \ldots, B_{k-1}$.

Proof: (DIY, by induction)

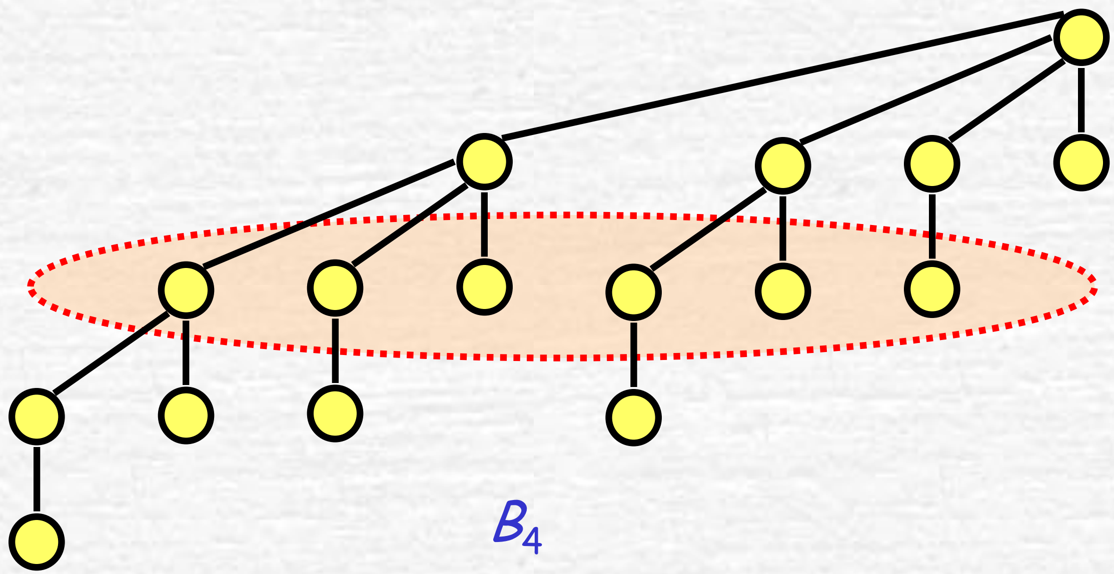
可用数学归纳法证明



Defining Property of Binomial Trees

There are exactly $\binom{k}{i}$ nodes at depth i, for B_k 组合数

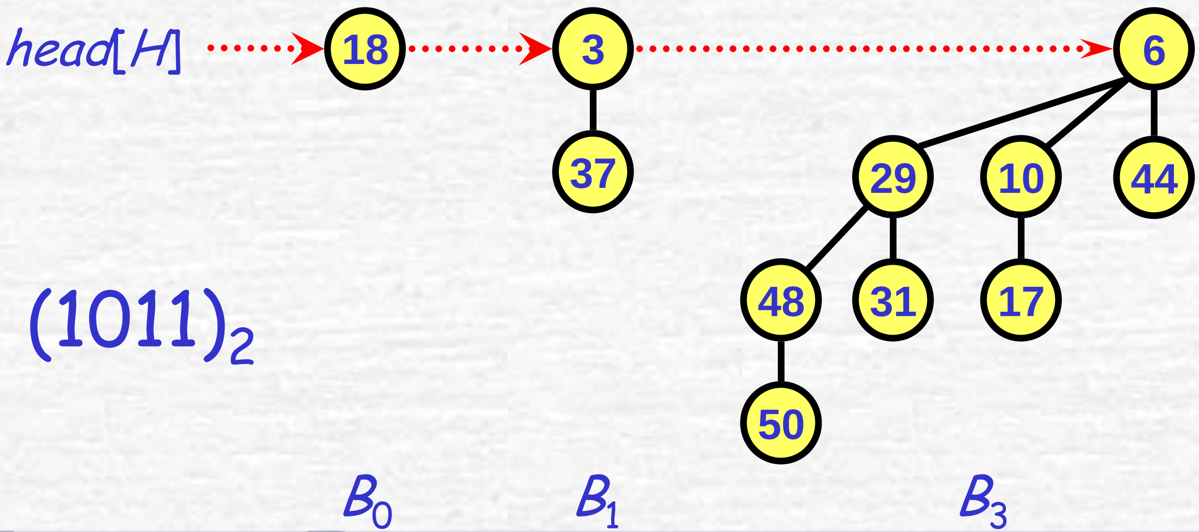
$$(0 \le i \le k)$$



depth	Nodes
0	1
1	4
2	6
3	4
4	1

Binomial Heap (Vuillemin, 1978)

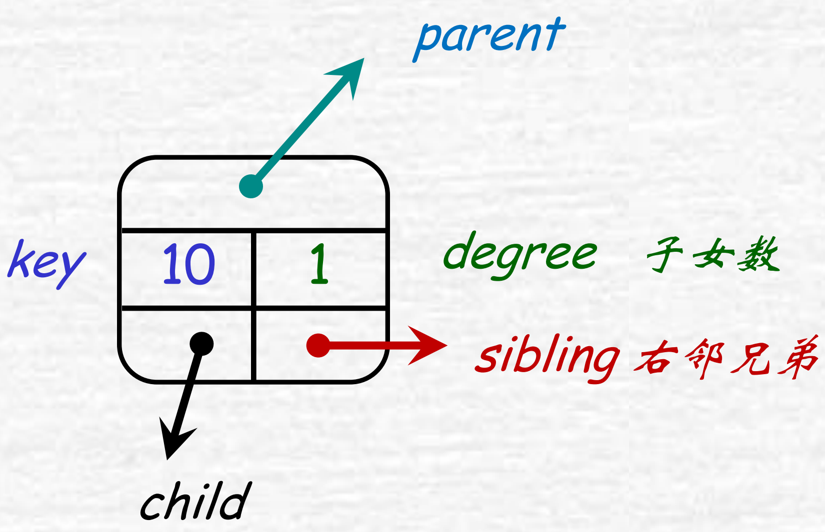
- A sequence of binomial trees that satisfy
  - binomial heap property (each tree B_k is a min-heap)
  - 0 or 1 binomial tree B_k of order k,
- There are at most $\lfloor \log n \rfloor + 1$ binomial trees.
- Eg: A binomial heap H with $n = 11$ nodes.



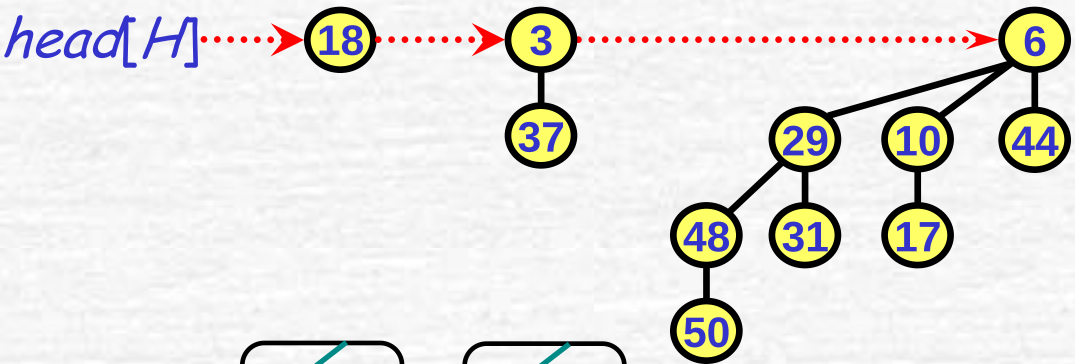
$11 = (1011)_2$

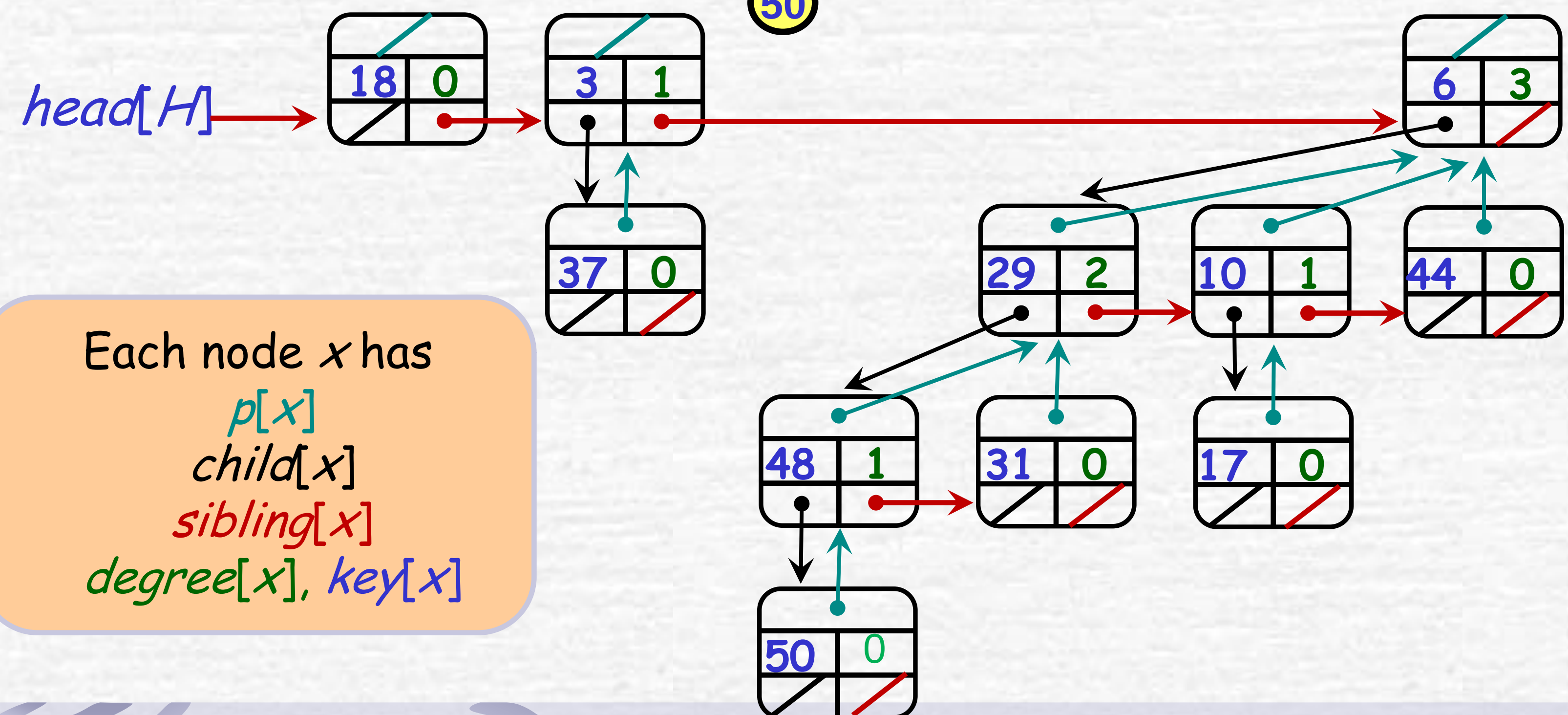
Representing Binomial Heaps (1)

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



Representing Binomial Heaps (2)





Each node *x* has
p[*x*]
child[*x*]
sibling[*x*]
degree[*x*], *key*[*x*]

Chapter 19 Binomial Heap (二项堆, in v2)

19.1 Priority Queue and Union op

19.2 Binomial Trees and Binomial Heap

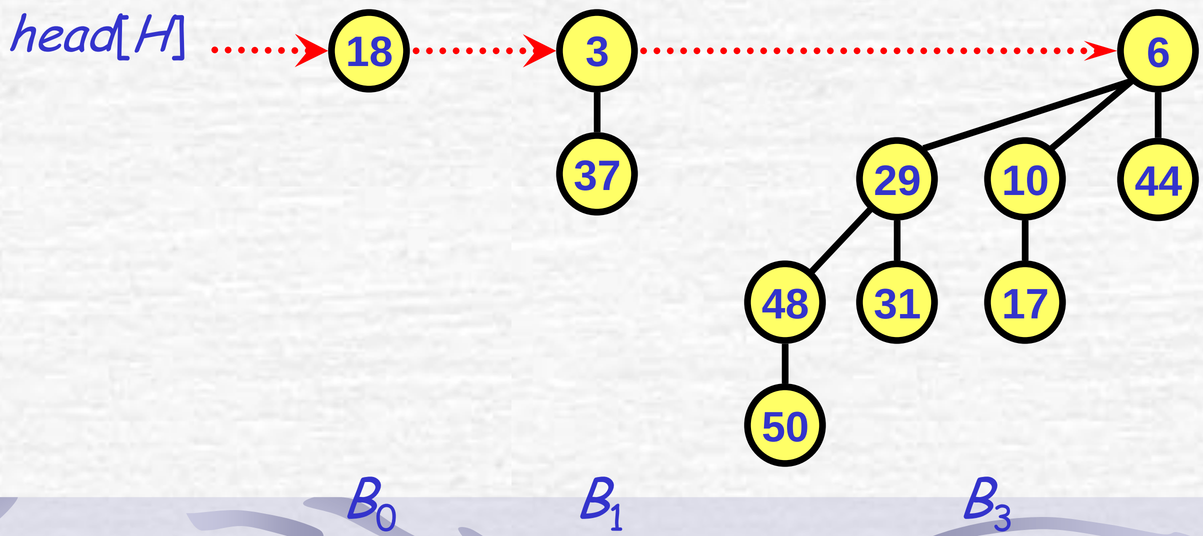
19.3 Operations on a Binomial Heap

19.3 Operations on a Binomial Heap

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

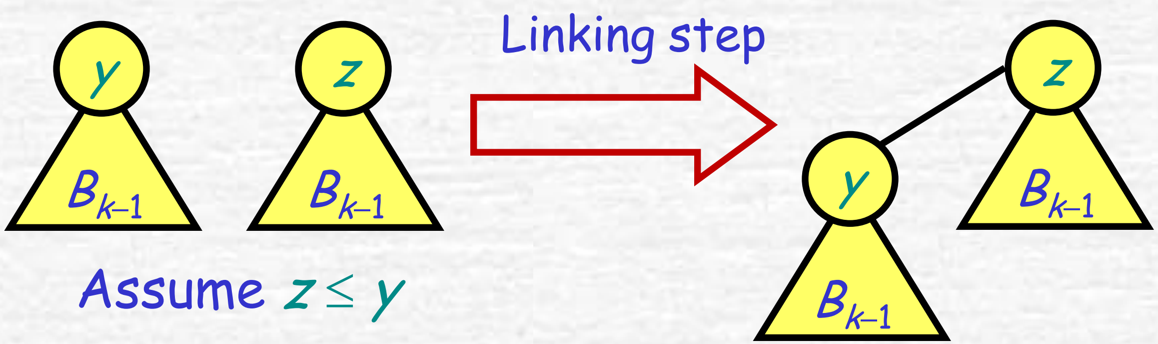
MAKE and MINIMUM

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



Linking Step: Fundamental Op

- BINOMIAL-LINK (y, z)



BINOMIAL-LINK (y, z) ▹ *Assume* $z \le y$

$p[y] \leftarrow z$

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

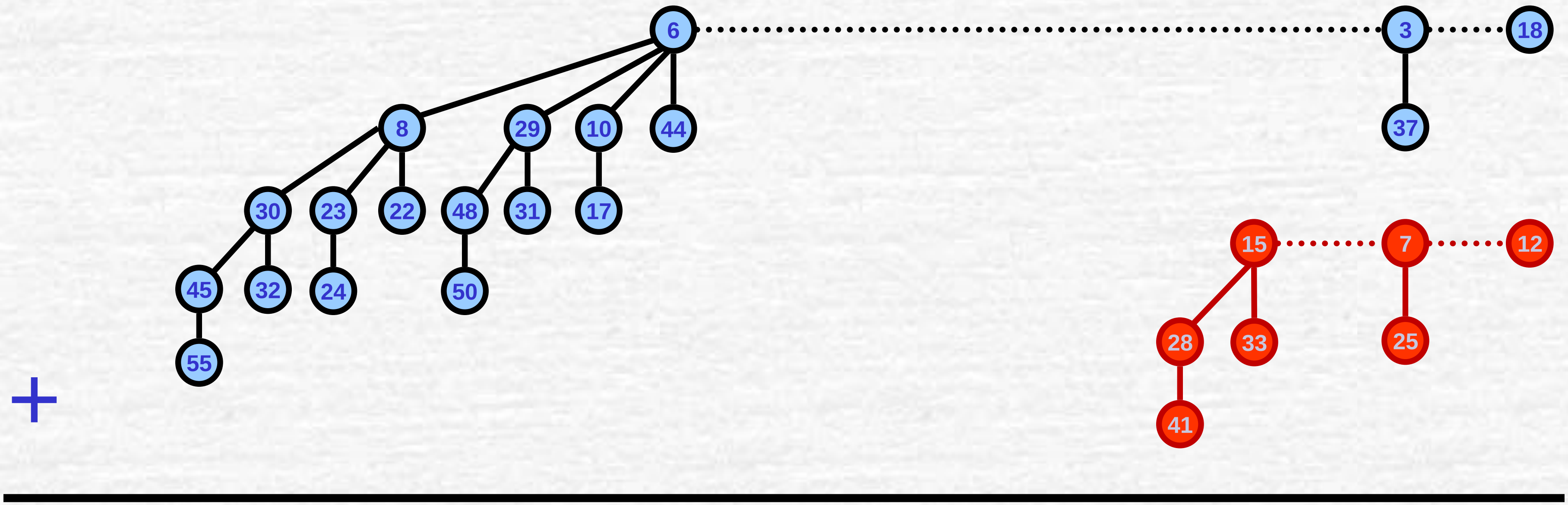
$child[z] \leftarrow y$

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

Constant time $O(1)$

Binomial Heap Union（1）

Let us look at the procedure of an example:



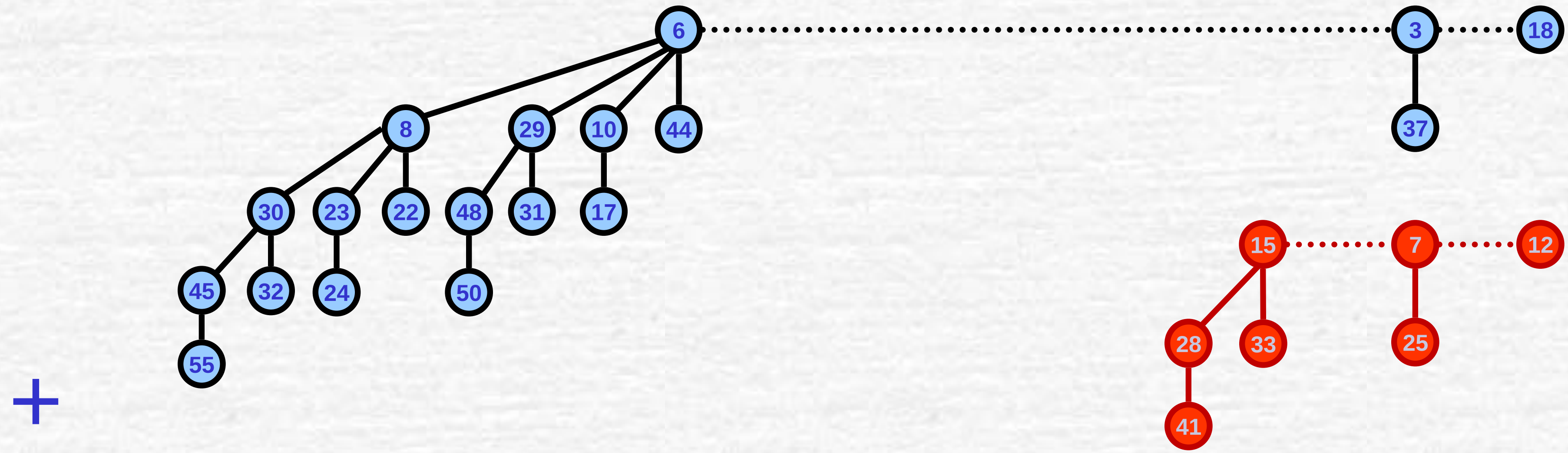
19 + 7 = 26

		1	*1*	*1*	
	1	0	0	1	1
+	0	0	1	1	1
	1	1	0	1	0

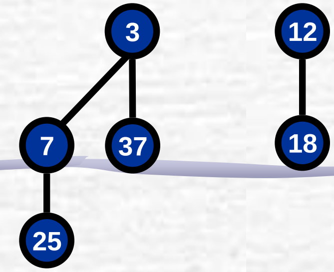
The binomial trees in the Binomial Heap at last: B_1,B_3,B_4

Binomial Heap Union

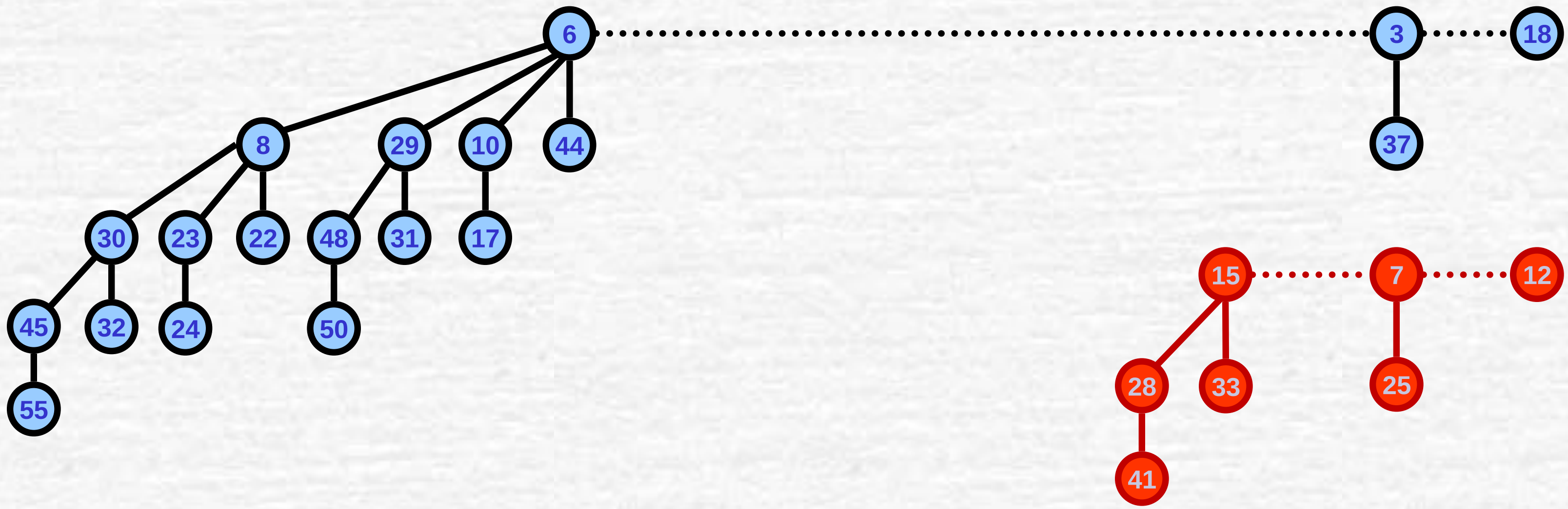
Temporary area:



Stable area:

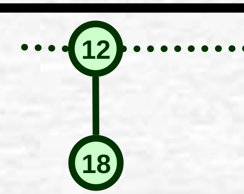


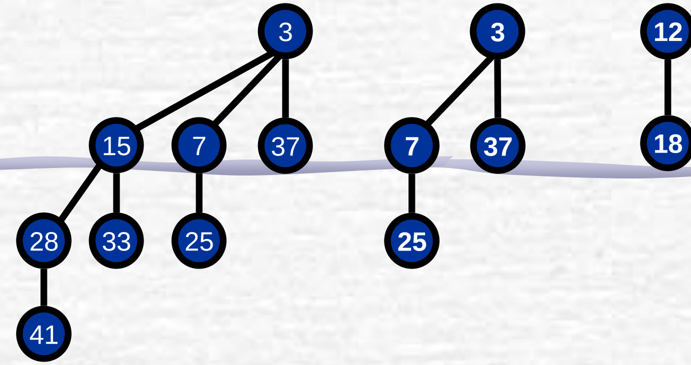
Temporary area:



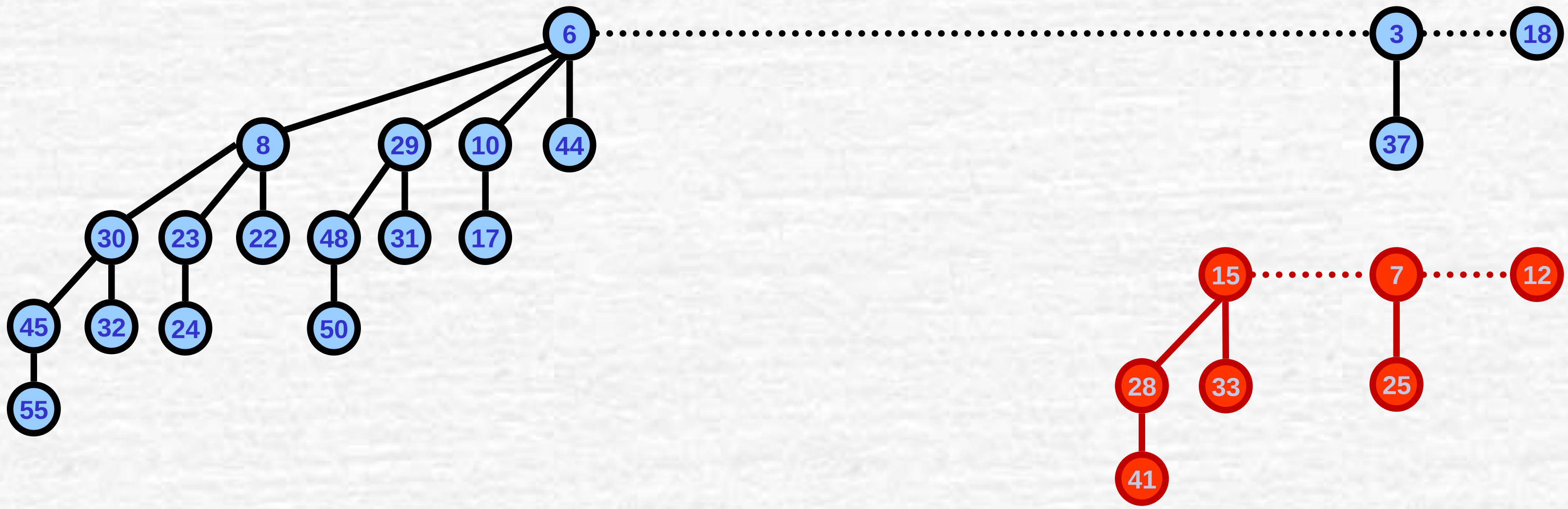
+

Stable area:



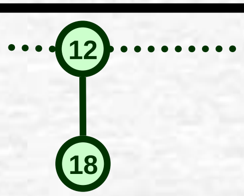


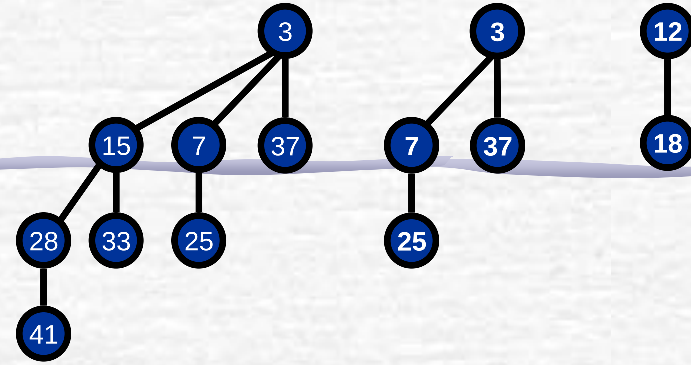
Temporary area:



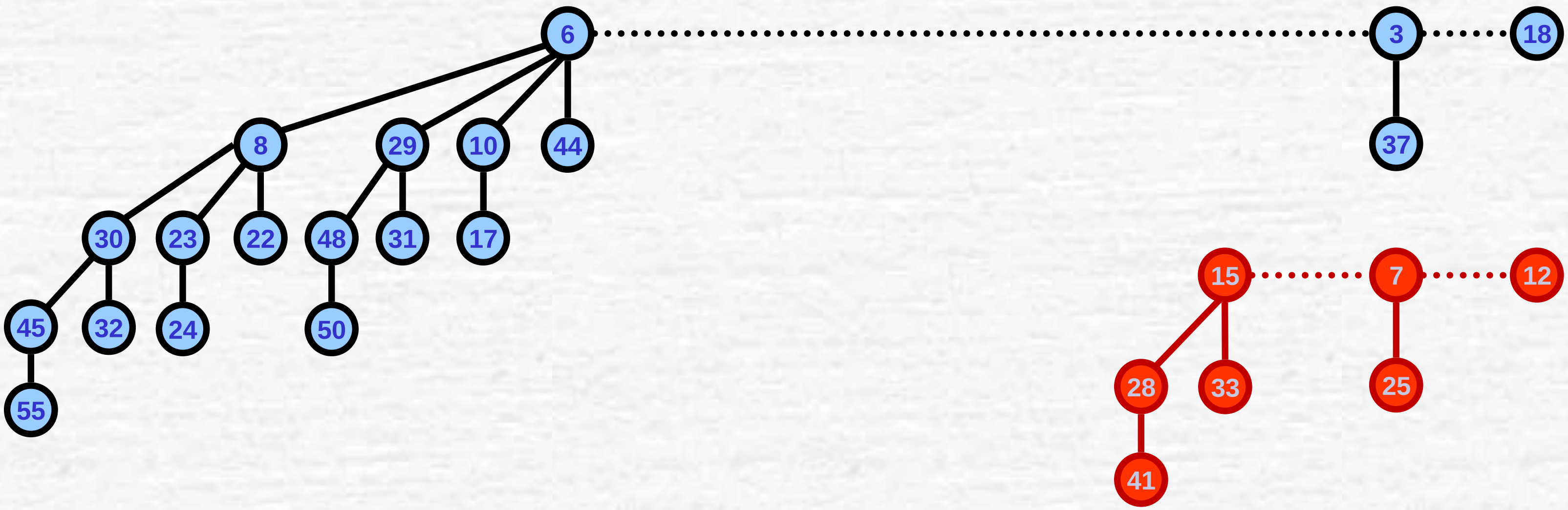
+

Stable area:



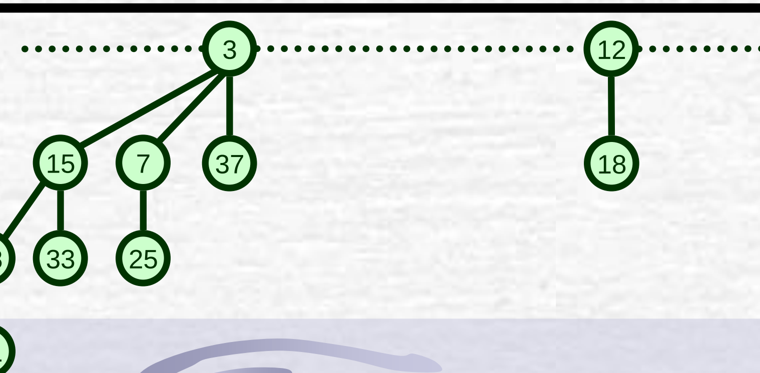


Temporary area:

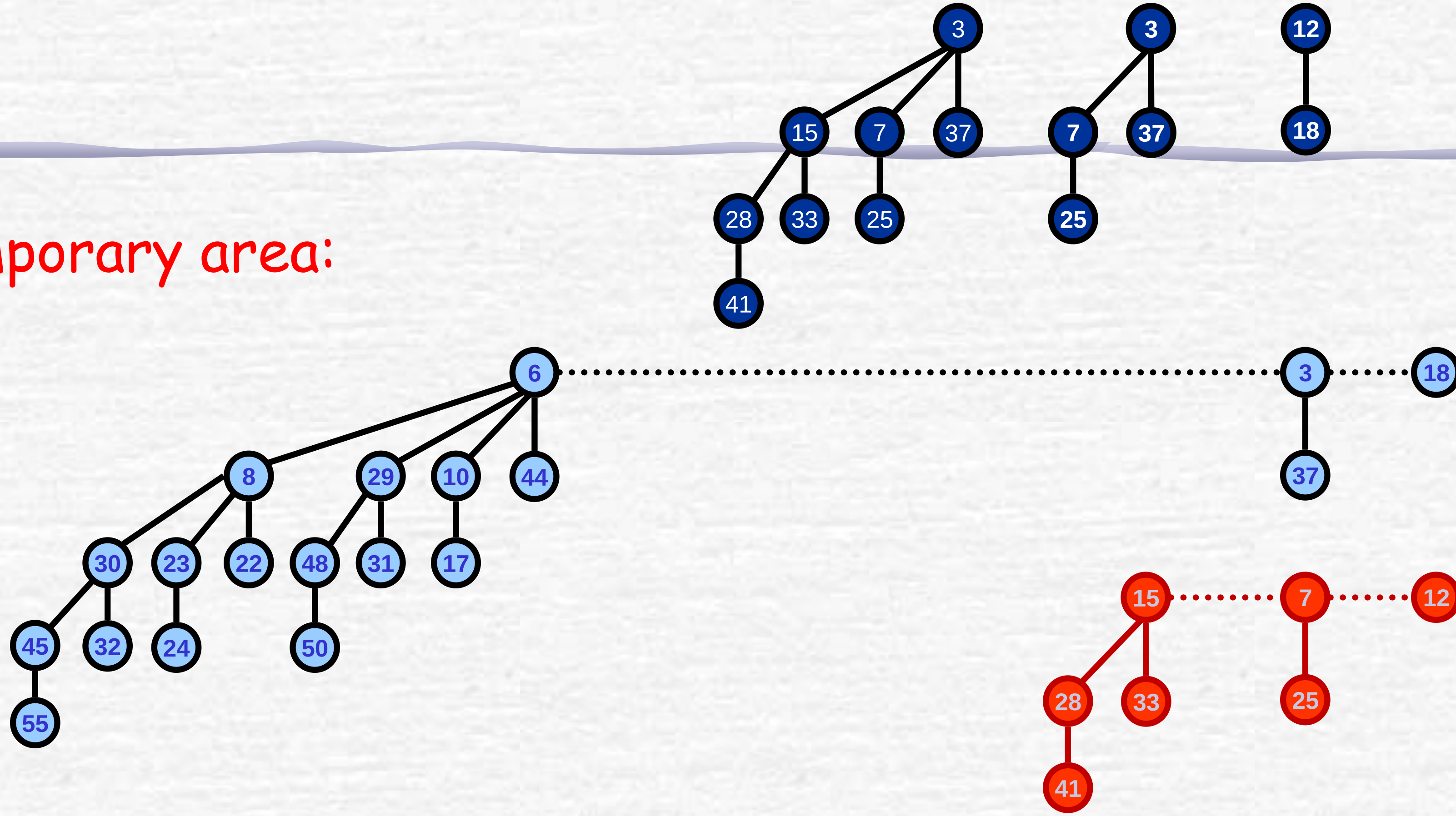


+

Stable area:

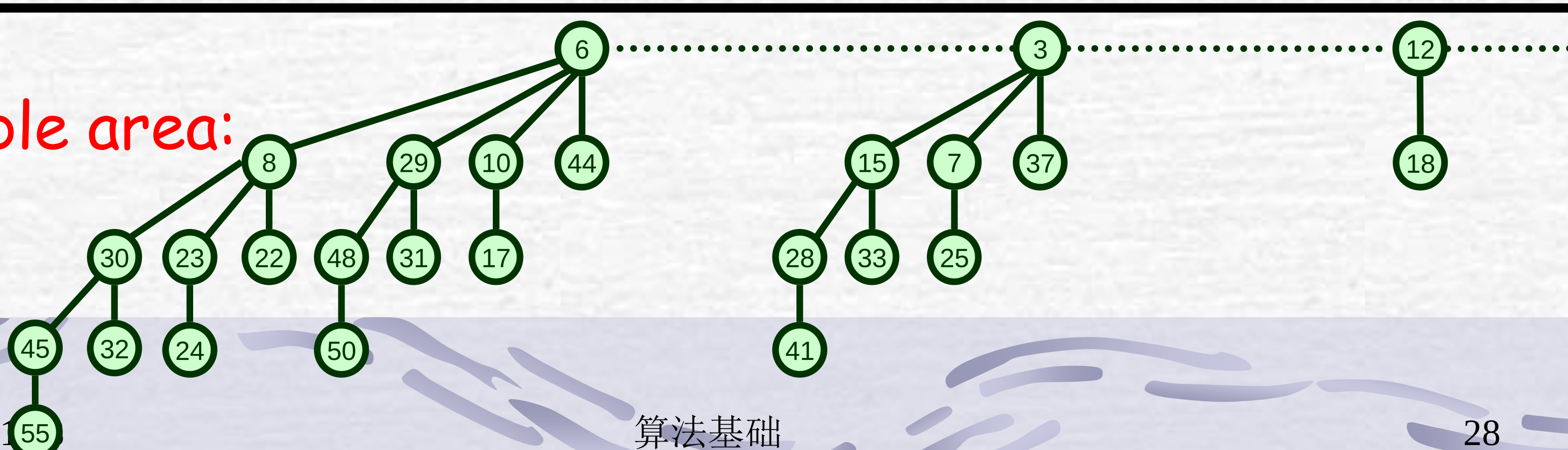


Temporary area:



+

Stable area:



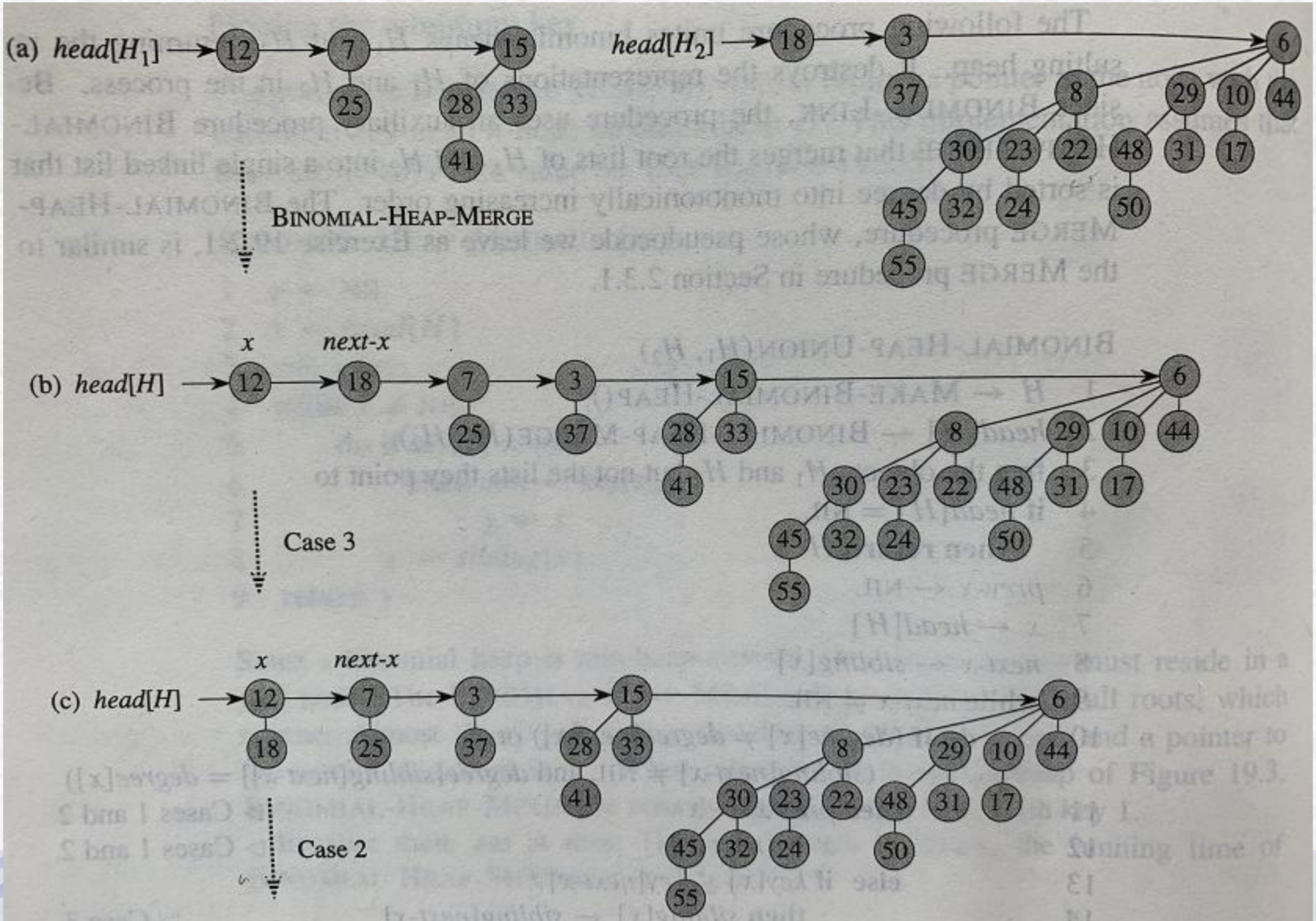
```
BINOMIAL-HEAP-UNION(H1, H2)
H ← MAKE-BINOMIAL-HEAP()
head[H] ← BINOMIAL-HEAP-MERGE(H1, H2)
free the objects H1 and H2 but not the lists they point to
if head[H] = NIL
   then return H
prev-x ← NIL
x ← head[H]
next-x ← sibling[x]
while next-x ≠ NIL
      do if (degree[x] ≠ degree[next-x]) or
               (sibling[next-x] ≠ NIL and degree[sibling[next-x]] = degree[x])
            then prev-x ← x                                   ▷ Cases 1 and 2
                 x ← next-x                                   ▷ Cases 1 and 2
            else if key[x] ≤ key[next-x]
                    then sibling[x] ← sibling[next-x]         ▷ Case 3
                         BINOMIAL-LINK(next-x, x)             ▷ Case 3
            else if prev-x = NIL                              ▷ Case 4
                    then head[H] ← next-x                     ▷ Case 4
                    else sibling[prev-x] ← next-x             ▷ Case 4
                 BINOMIAL-LINK(x, next-x)                     ▷ Case 4
                 x ← next-x                                   ▷ Case 4
         next-x ← sibling[x]
return H
```

Binomial Heap Union（3）

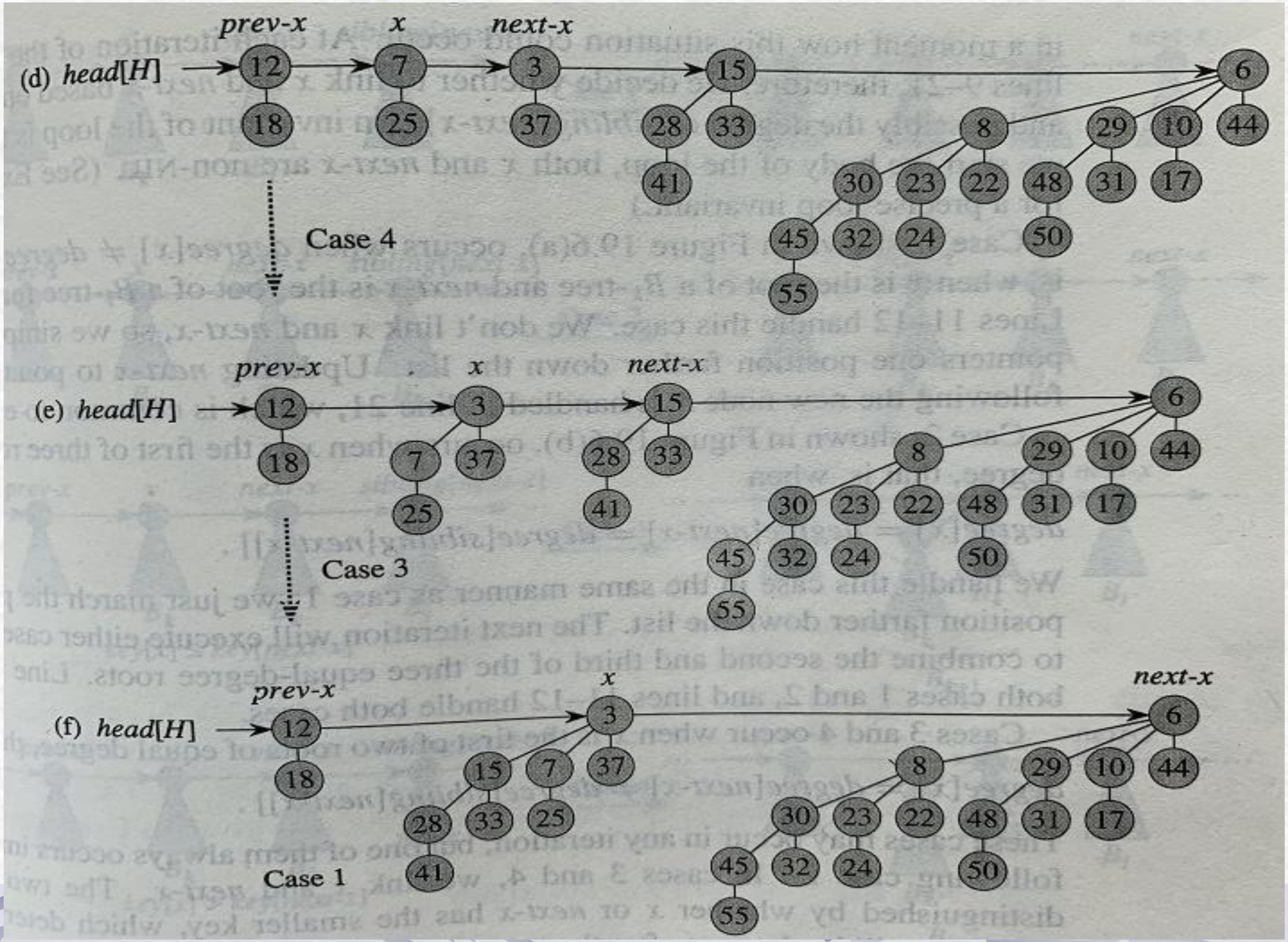
Case classification：

$$
degree[x]\ \begin{array}{l}
\neq degree[\text{next-}x] \quad case\ 1 \\
= degree[\text{next-}x]\ \begin{array}{l}
= degree[sibling[\text{next-}x]] \quad case\ 2 \\
\neq degree[sibling[\text{next-}x]]\ and\ \begin{array}{l}
key[x] \leq key[\text{next-}x] \quad case\ 3 \\
key[x] > key[\text{next-}x] \quad case\ 4
\end{array}
\end{array}
\end{array}
$$

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.: $O(\log n)$ $[n = n_1 + n_2]$

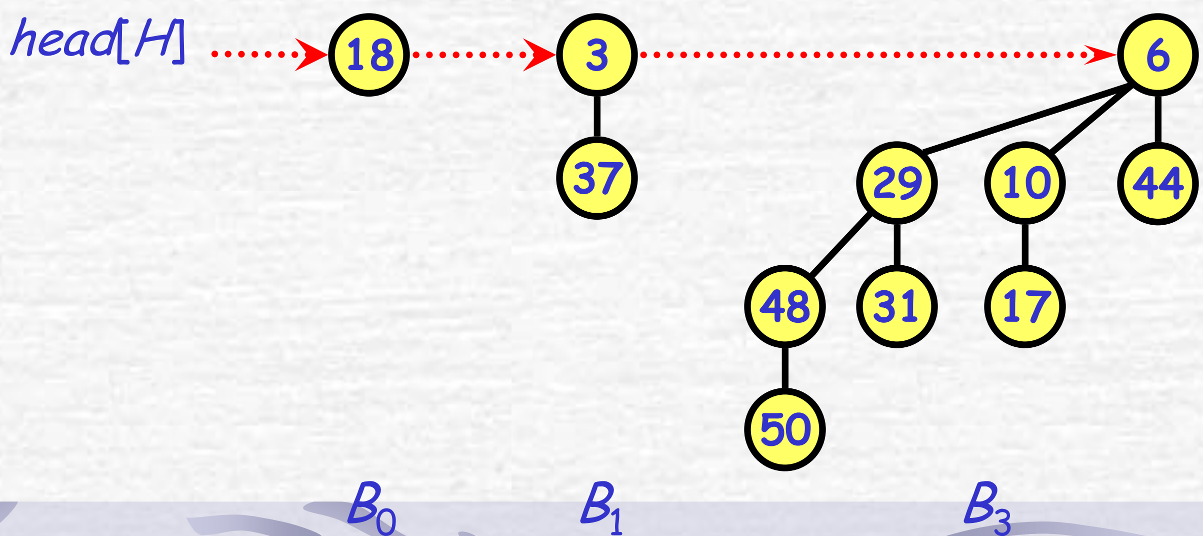
19 + 7 = 26

		1	1	1	
	1	0	0	1	1
+	0	0	1	1	1
	1	1	0	1	0

More Operations (1)

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

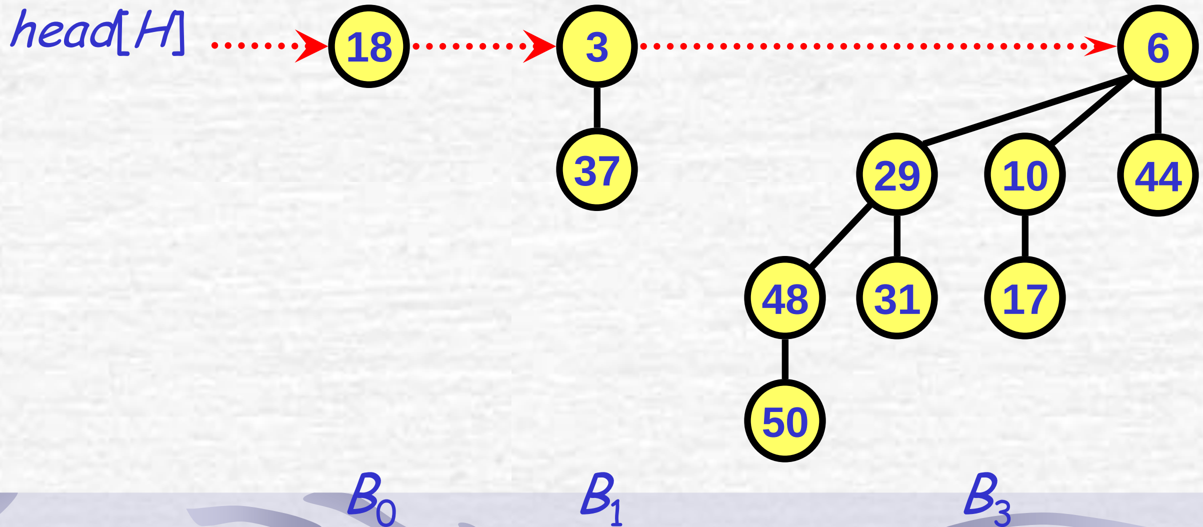
删除最小结点，形成一个新的二项堆，union两个二项堆



More Operations (2)

- BINOMIAL-HEAP-DECREASE (H, x, k) 与整堆操作相似
- BINOMIAL-HEAP-DELETE (H, x)

首先将x置为负无穷，整堆后，再删除最小结点（那个负无穷结点）



Summary

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

End of Ch19