Fibonacci Heaps

(B-Heaps serve as Special Cases of F-Heaps)

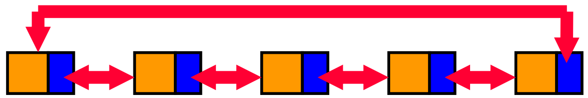
Min Fibonacci Heap

- Collection of min trees.
- The min trees need NOT be Binomial trees.
  - 沒有一定
  - 但，也可以是!

不支援search!!!

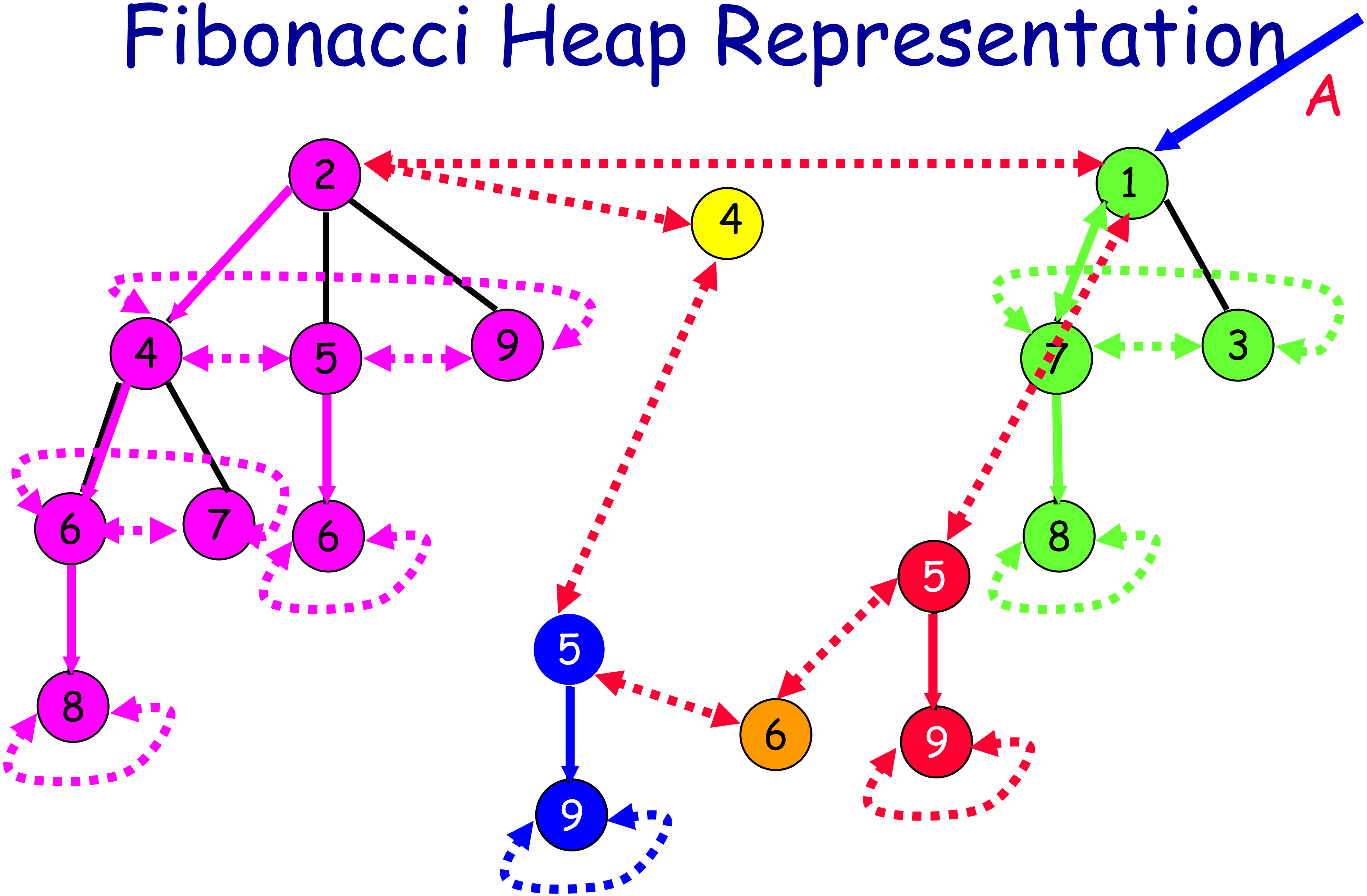
Node Structure

- Data
- Child
- Left and Right Sibling
  - Used for circular doubly linked list of siblings.



- Parent
  - Pointer to parent node.
- ChildCut flag
  - True if node has lost a child since it became a child of its current parent.
  - Set to false if node goes to the top-level.
  - Undefined for a root node.

Fibonacci Heap Representation

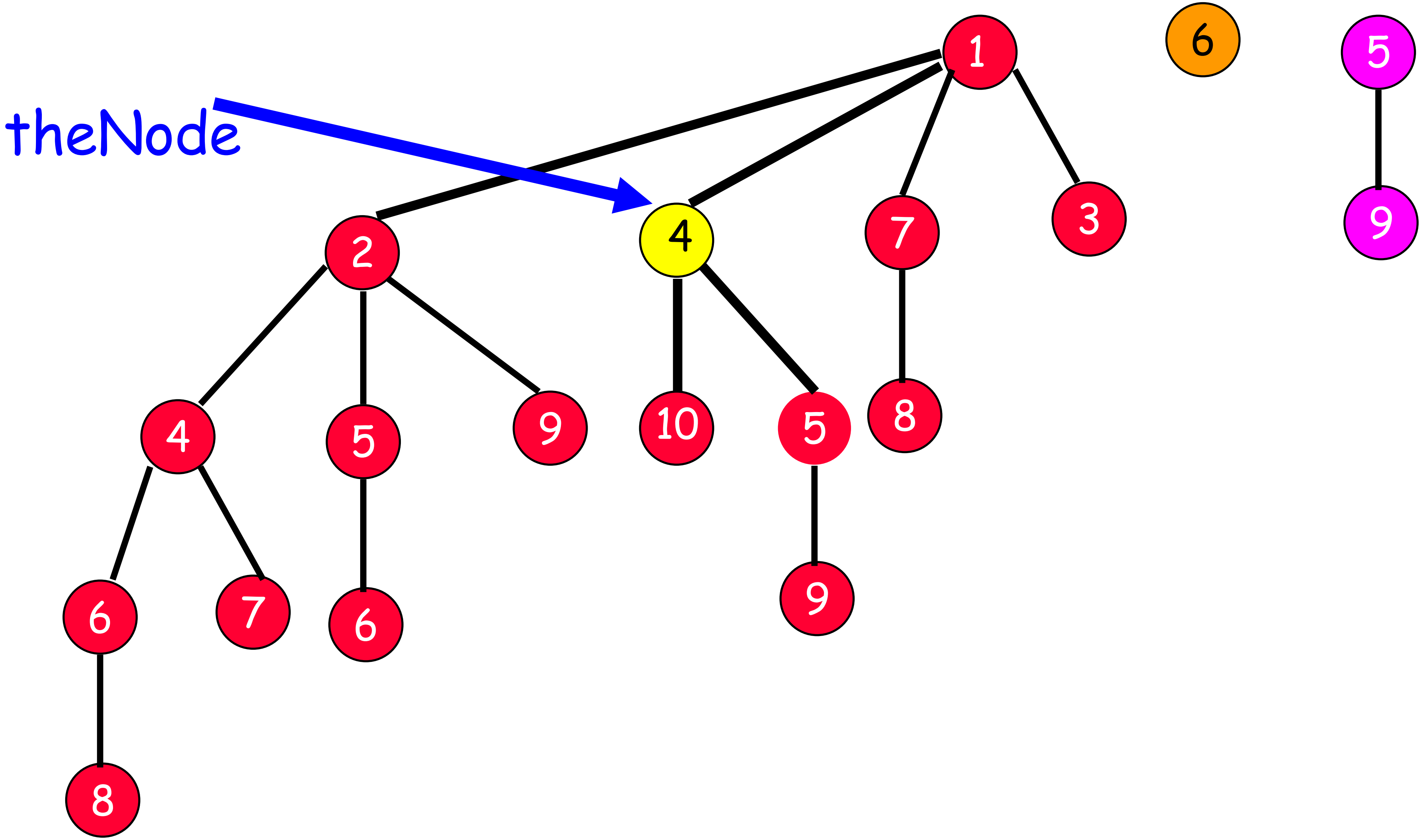


Parent and ChildCut fields not shown.

Delete(theNode)

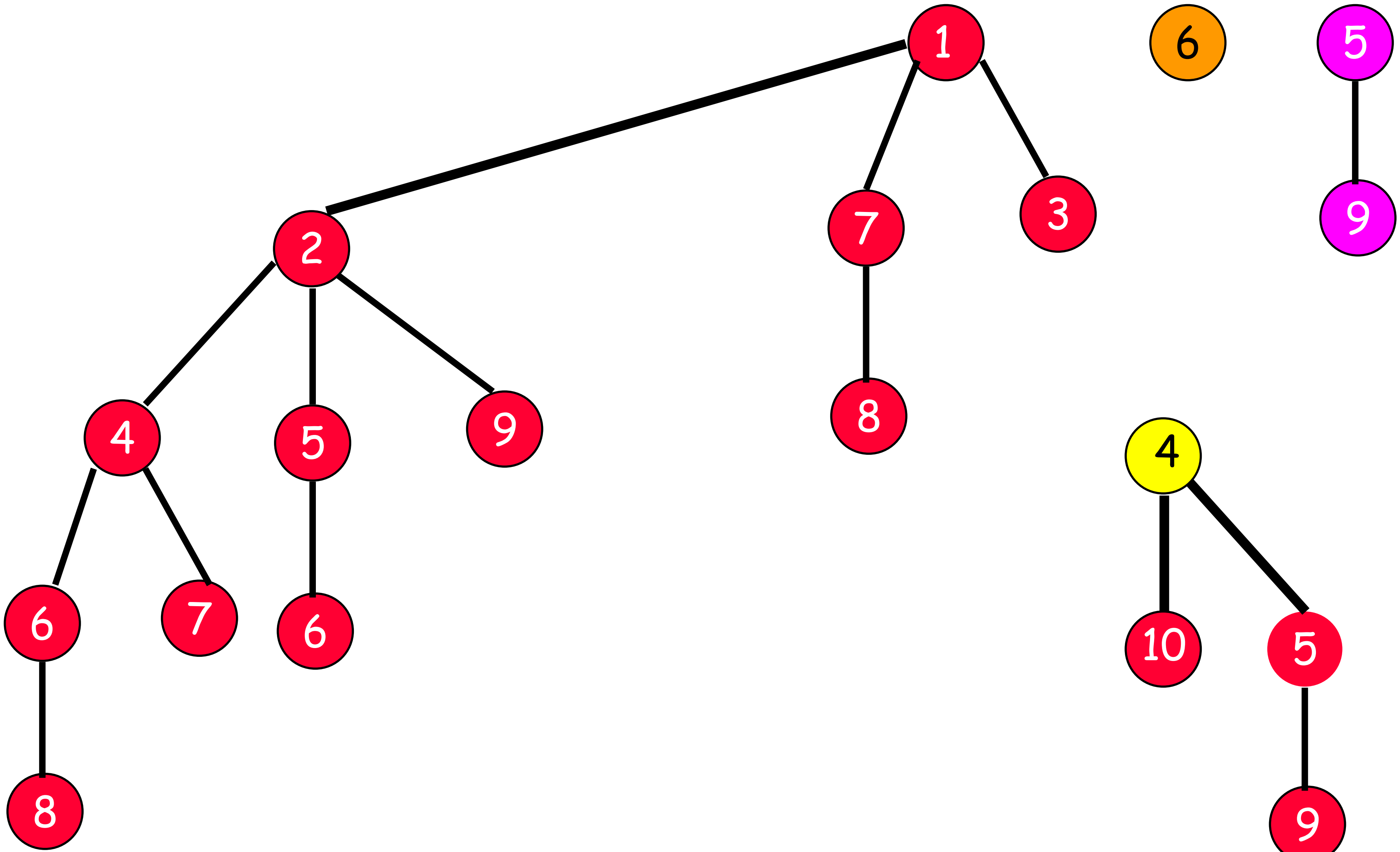
- theNode points to the Fibonacci heap node that contains the element that is to be deleted.
  - 假設有個方式 (是甚麼方式?) 可以找到要被delete的 theNode
- theNode points to min element => do a delete min.
  - In this case, complexity is the same as that for delete min.

Delete(theNode)



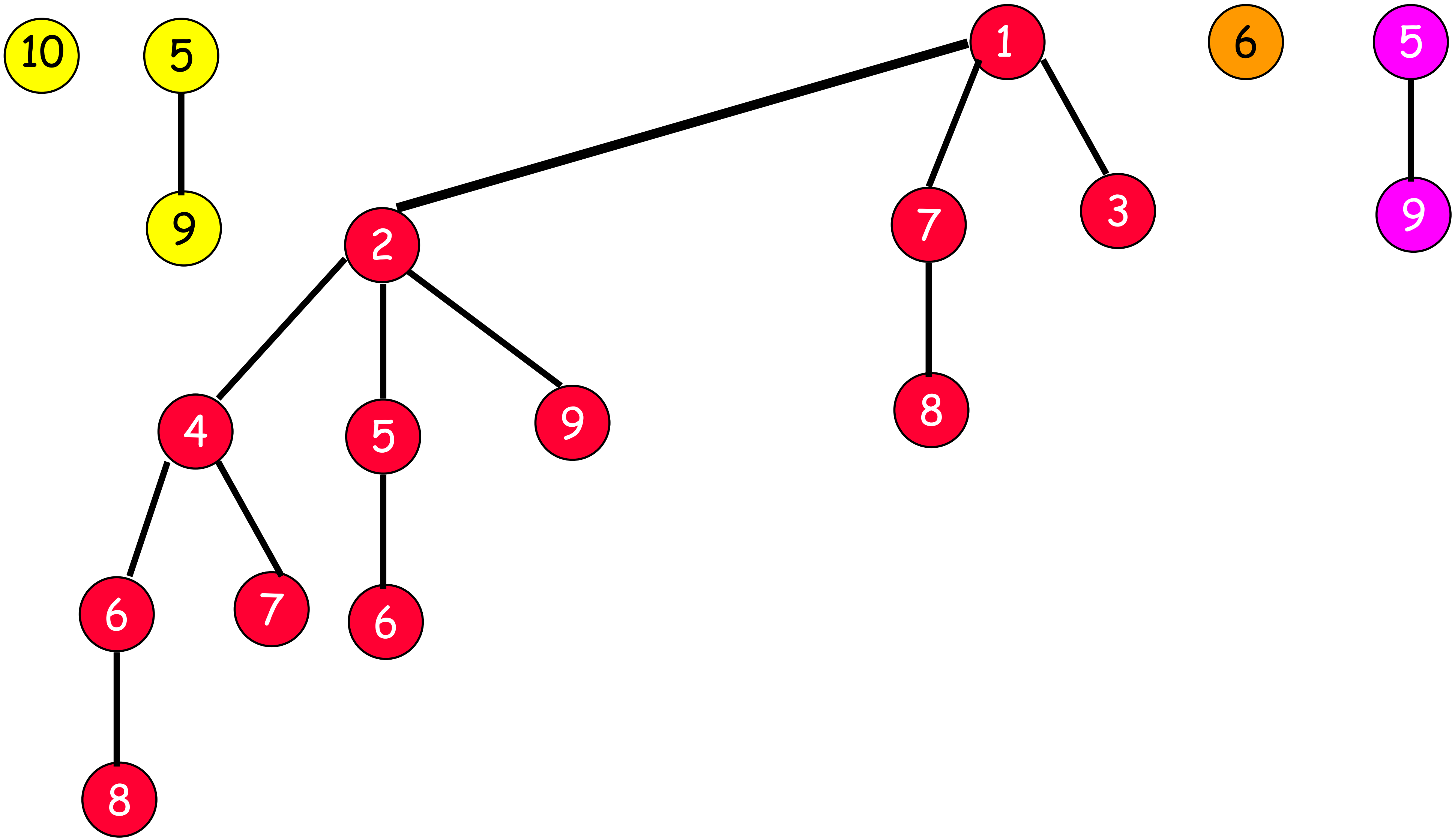
Remove theNode from its doubly linked sibling list.

Delete(theNode)

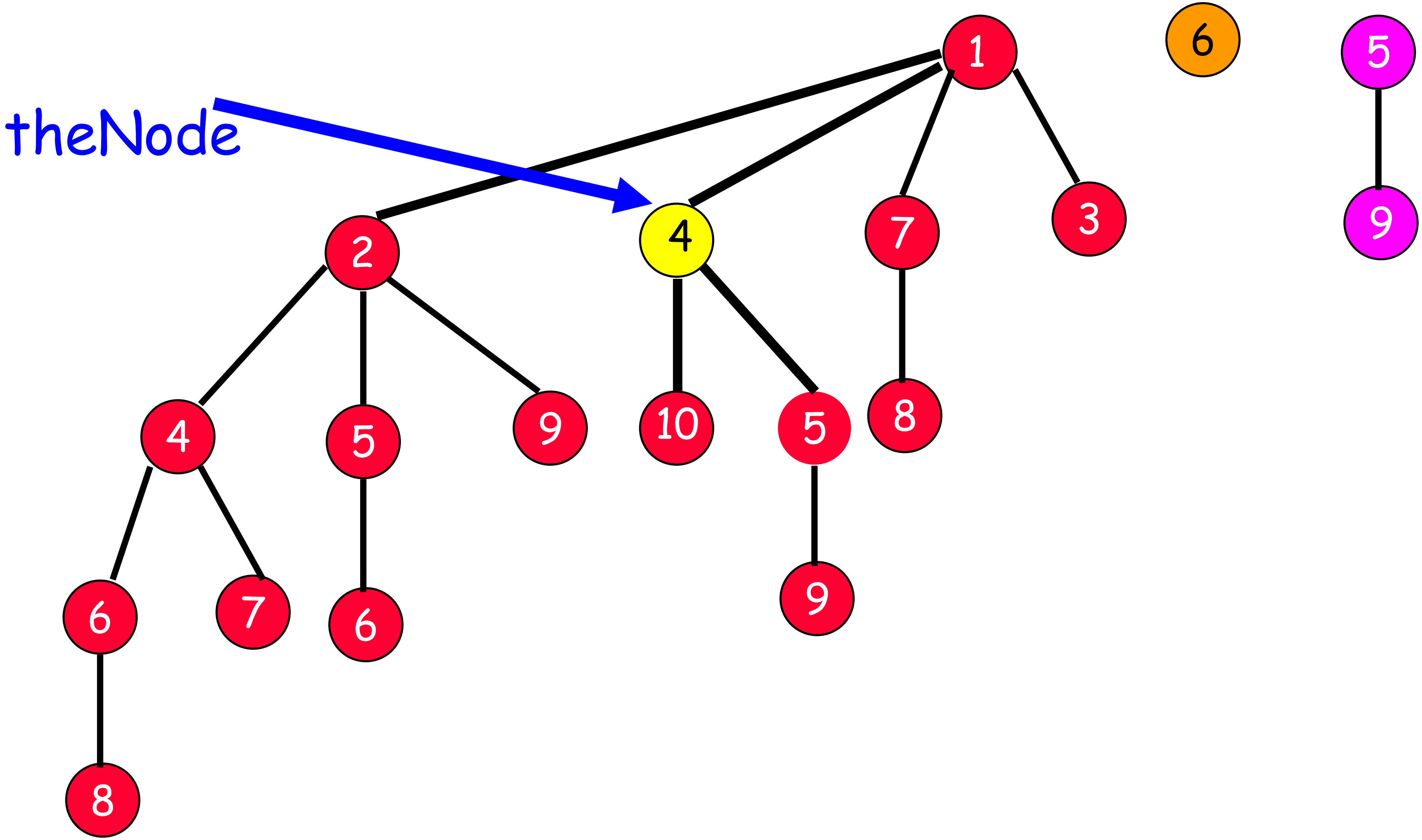


Combine top-level list and children of theNode.

Delete(theNode)

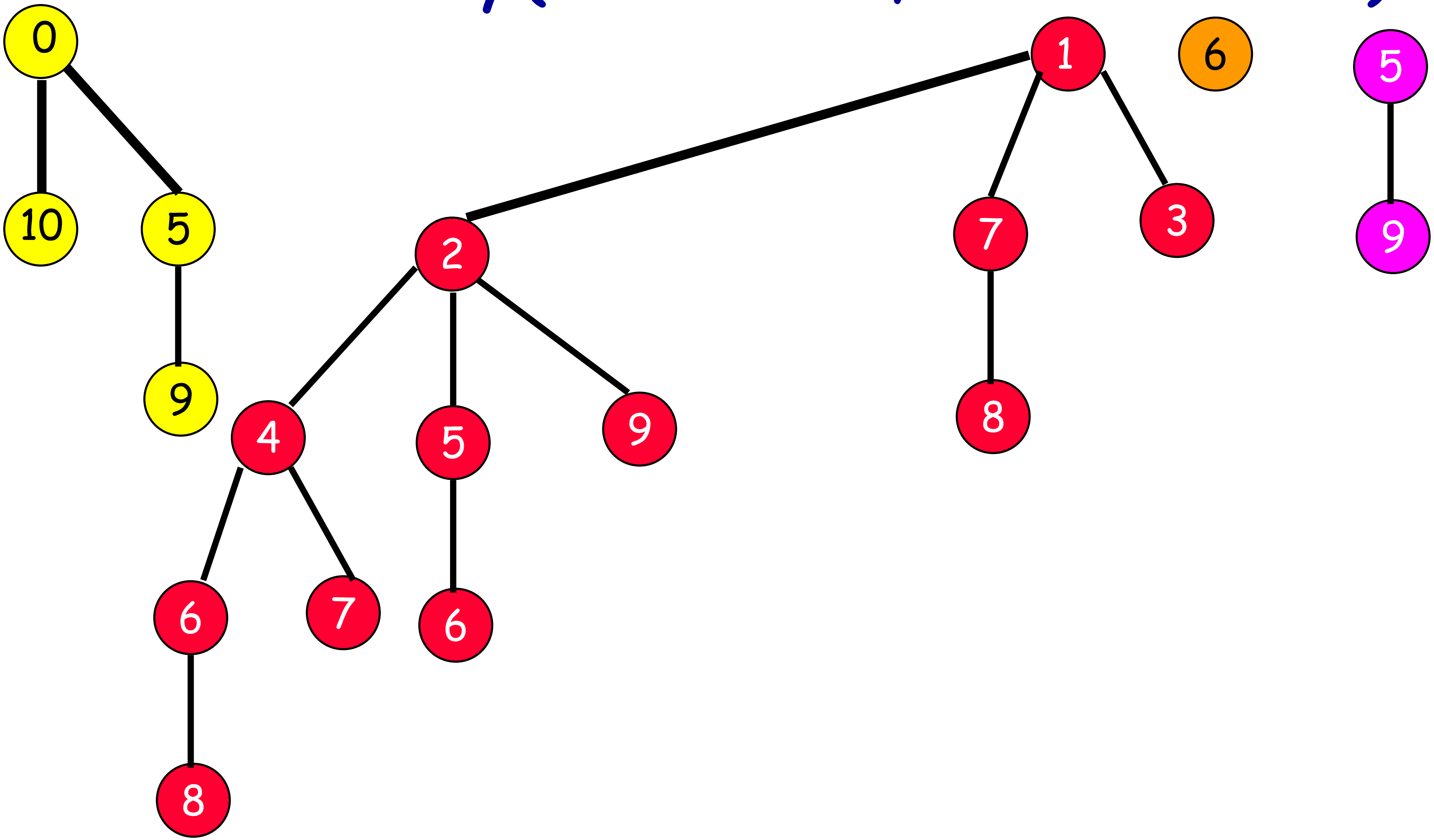


DecreaseKey(theNode, theAmount)



- If theNode is not a root and new key < parent key, remove subtree rooted at theNode from its doubly linked sibling list.
- Insert into top-level list.

DecreaseKey(theNode, theAmount)

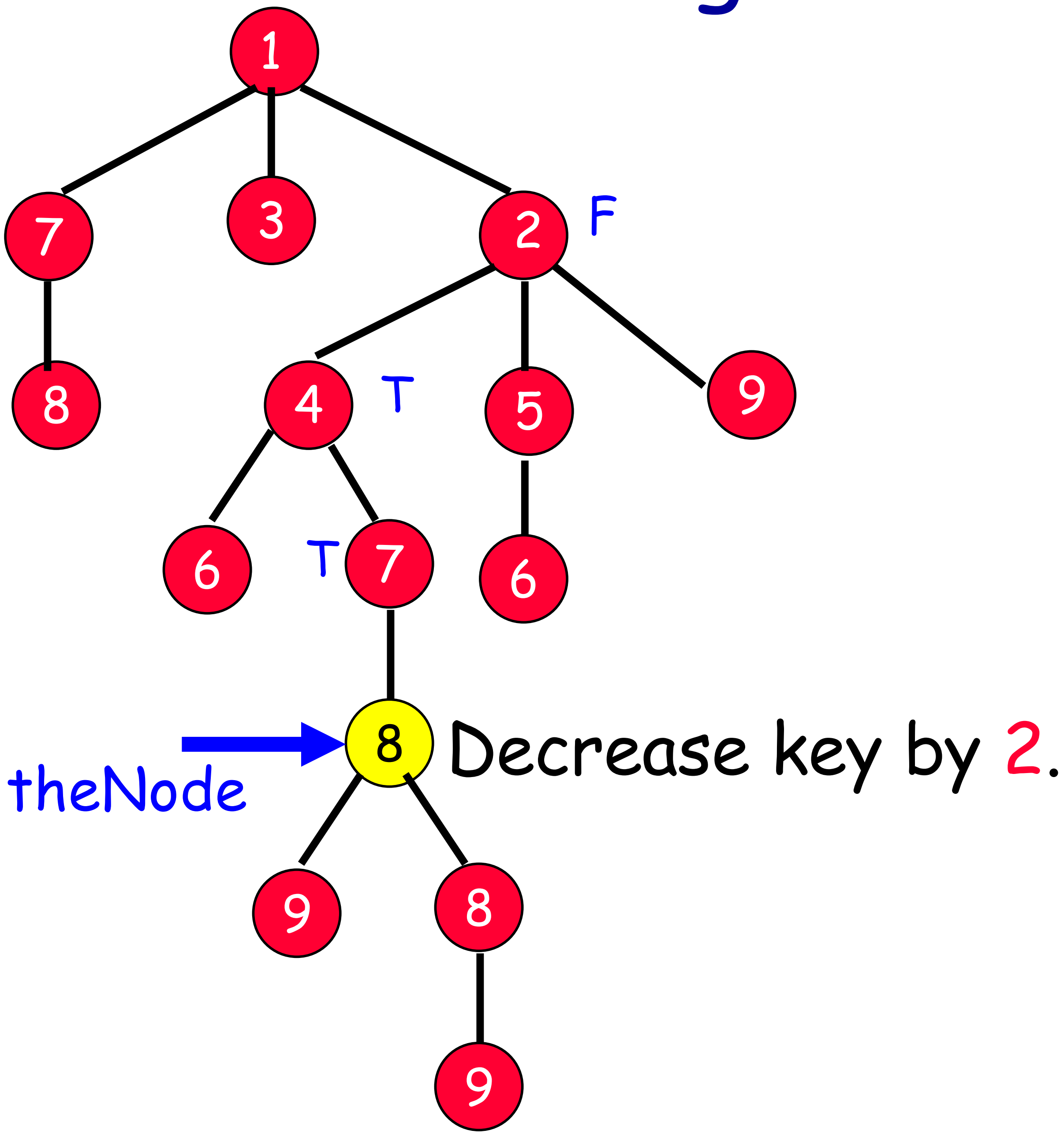


Cascading Cut

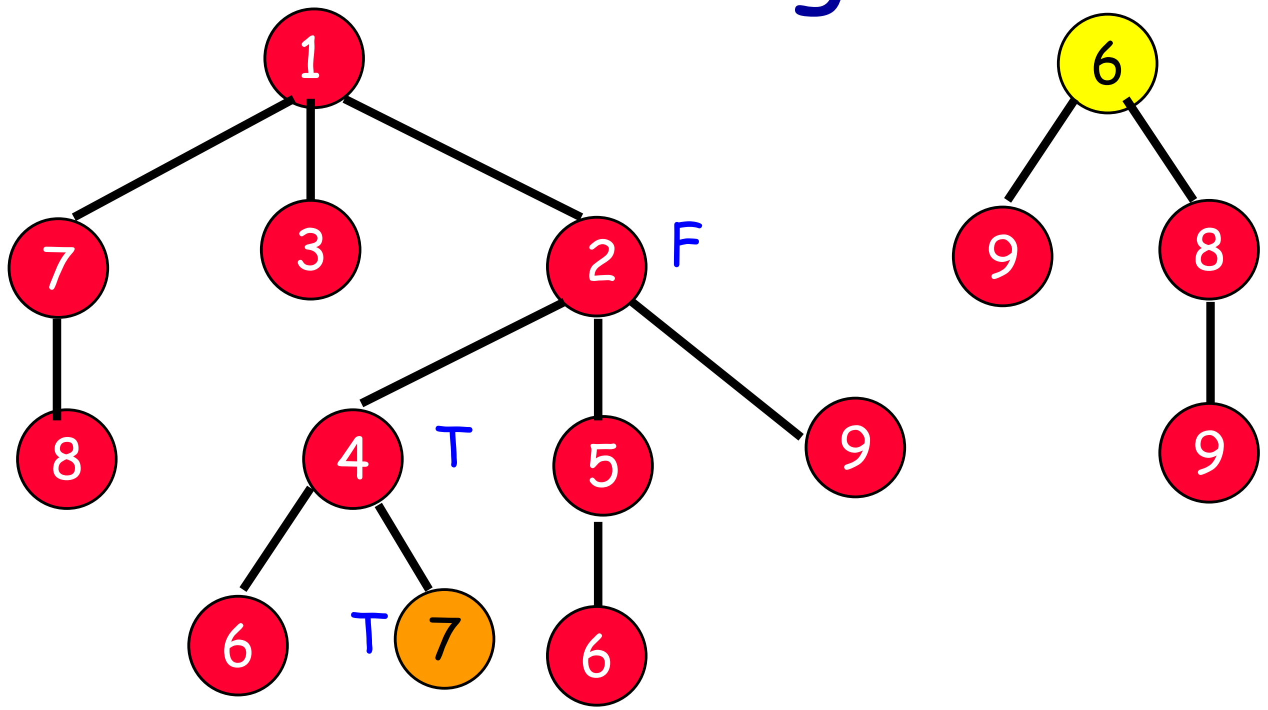
- When theNode is cut out of its sibling list in a remove or decrease key operation, follow path from parent of theNode to the root.
- Encountered nodes (other than root) with ChildCut = true are cut from their sibling lists and inserted into top-level list.
- Stop at first node with ChildCut = false.
  - For this node, set ChildCut = true.

- Note: ChildCut becomes "false" if being merged due to delete min (or delete)
  - The ChildCut is associated with the root of some F-heap in the top-level list (這個root是當時被cut出來的，當然其整個子樹也都跟著被帶出來)

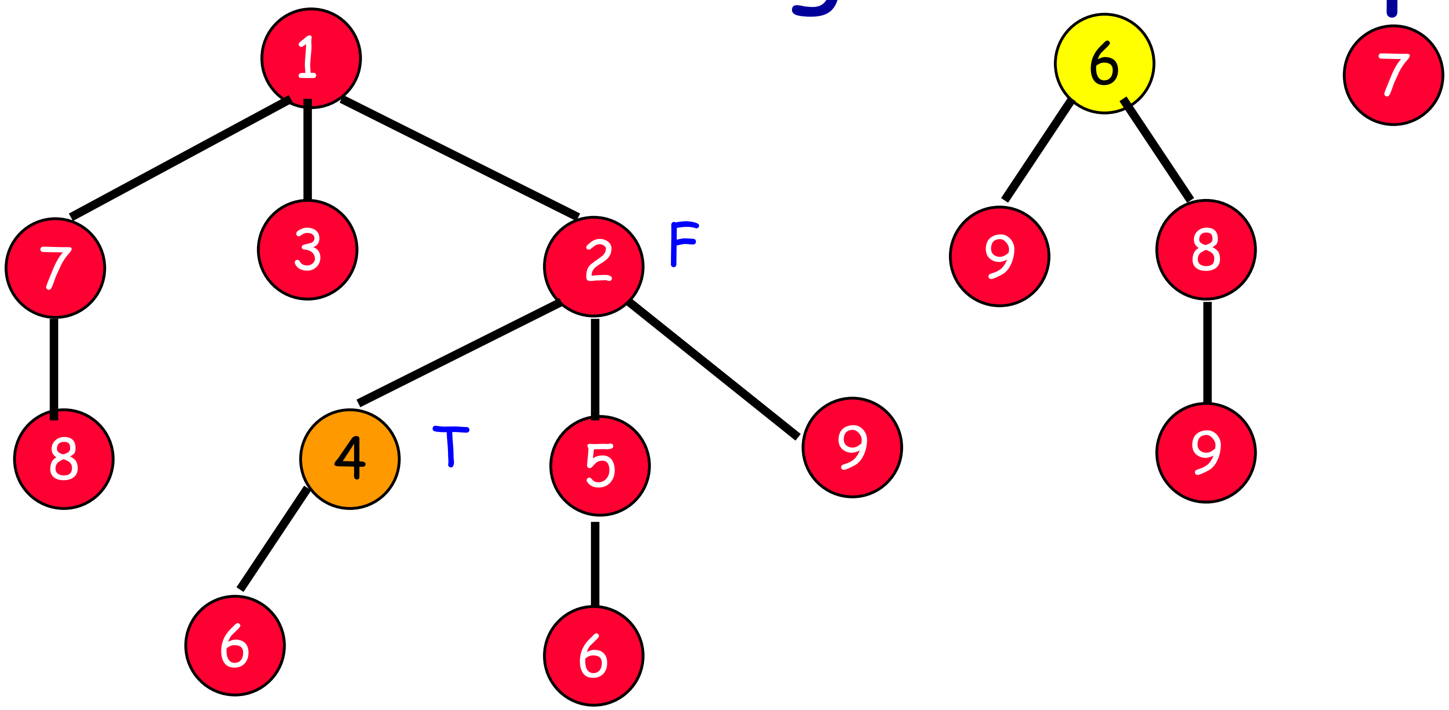
Cascading Cut Example



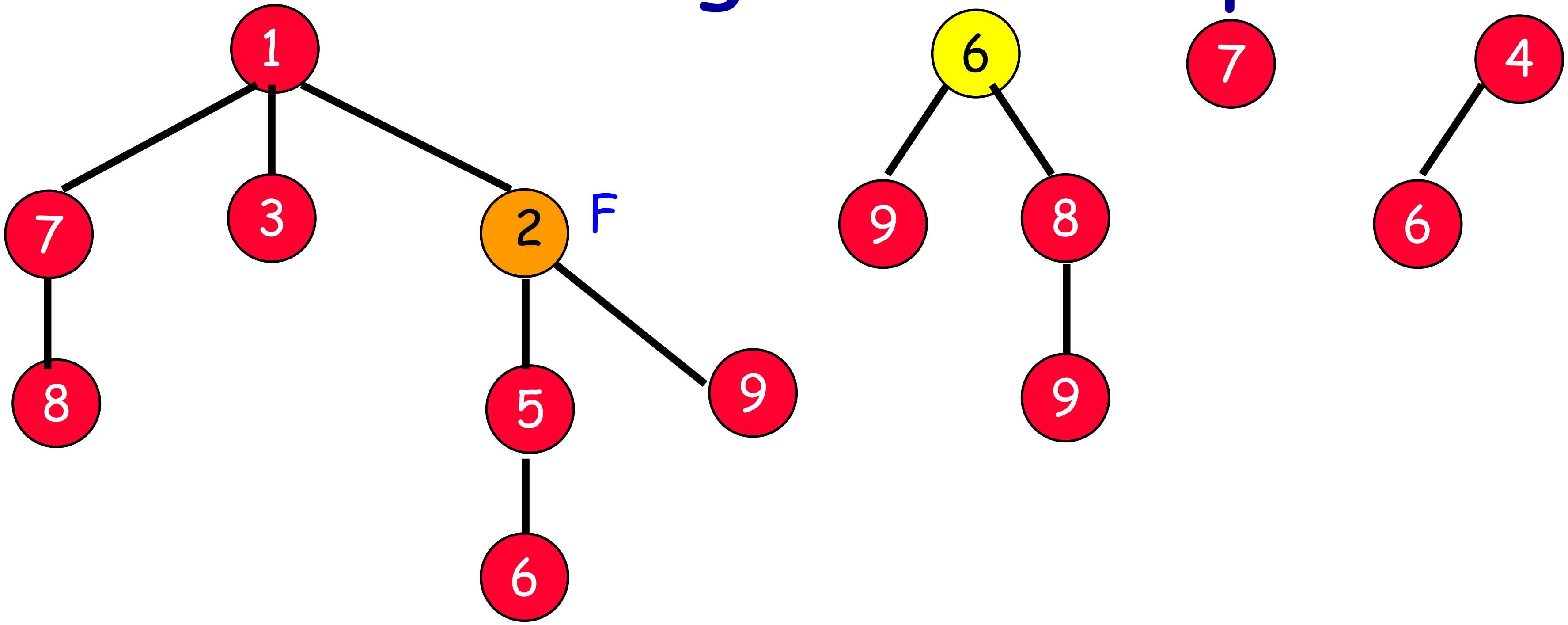
Cascading Cut Example



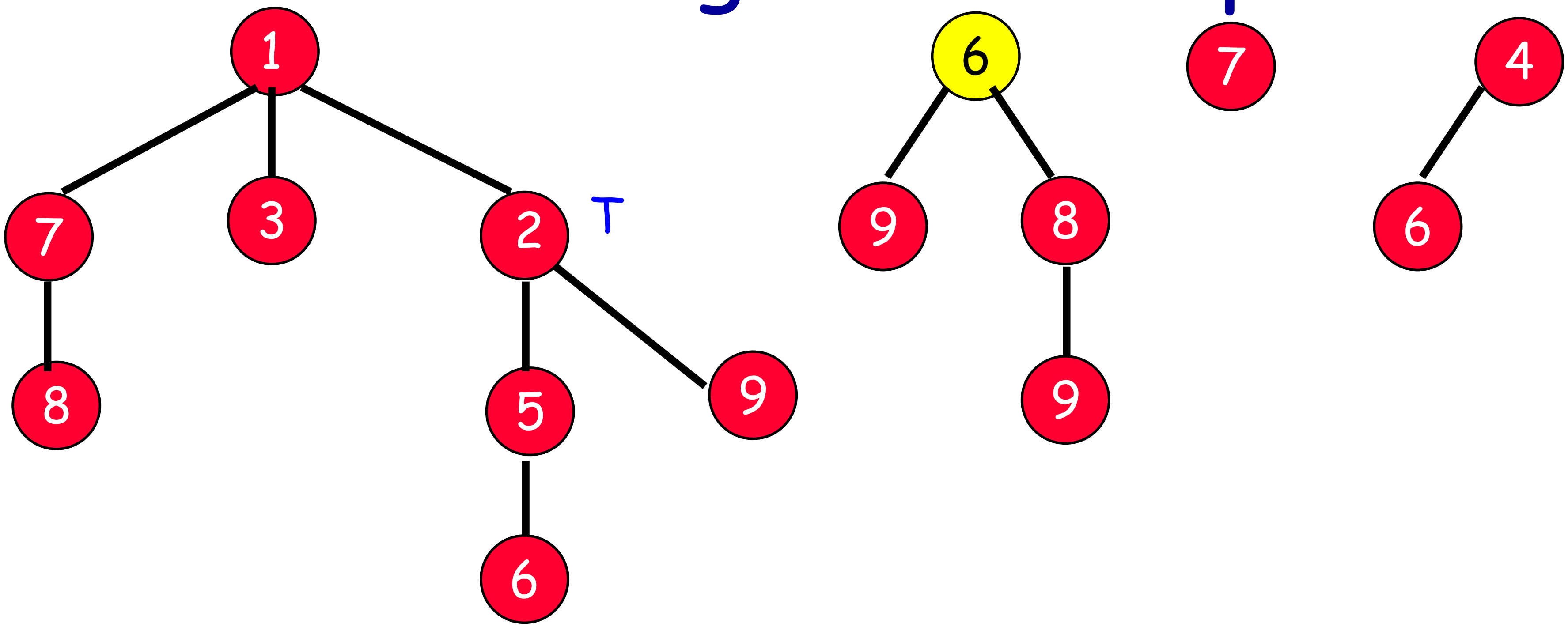
Cascading Cut Example



Cascading Cut Example

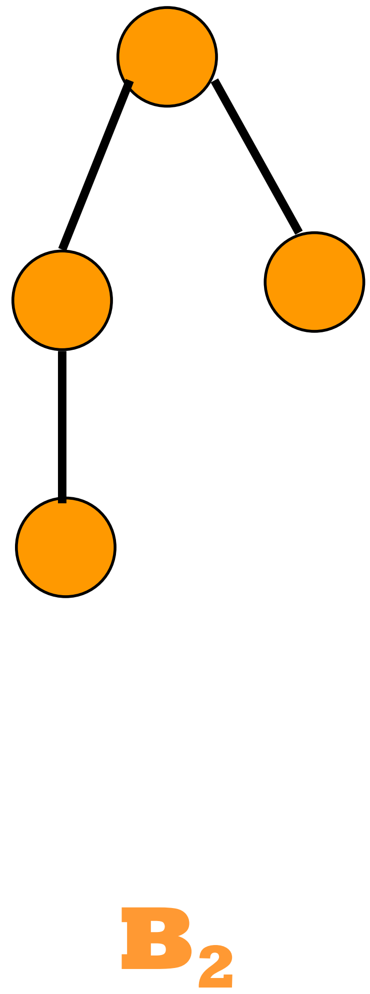


Cascading Cut Example

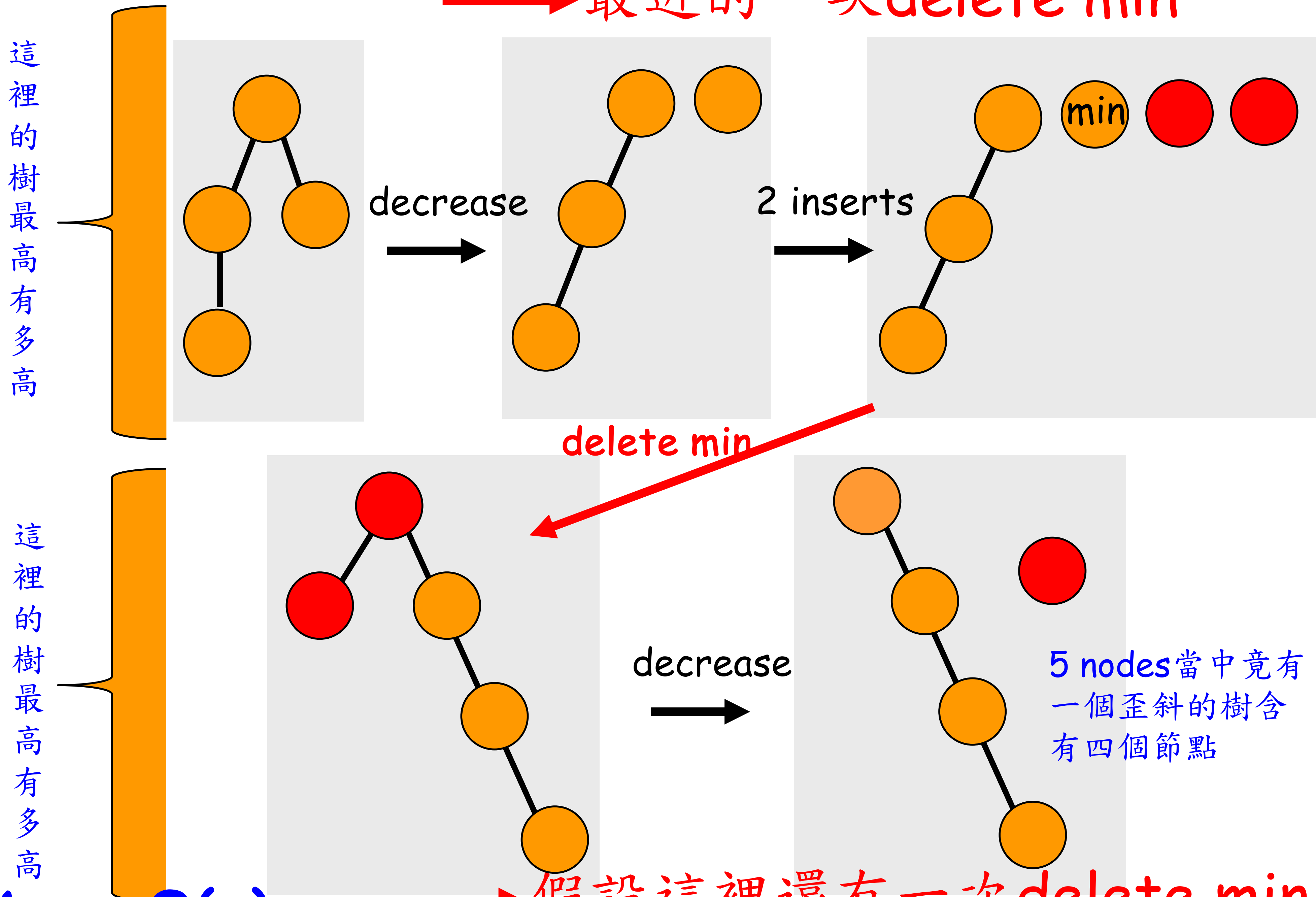


- Actual complexity of cascading cut is O(h) = O(n). //Why?
- 或者，正比於insert+decrease op's的個數 (自從上一次delete min或delete後) // amortized cost analytic的基礎

- Do ops to create a B2 (height = 3).
- Now do a decrease key to get a chain of height (length) 3 and a B0.
- Do 2 inserts, a remove min, and a decrease key to get a chain of height 4 and B0.
- We can increase height by 1 each time we do 2 inserts, a remove min, and a decrease key.
- So, height is O(n), where n is number of operations (or number of elements in tree).



最近的一次delete min



假設這裡還有一次delete min

Ans.: O(n)

底下討論要花一點時間…

- 為甚麼Fibonacci?
- 為甚麼一個internal node被cut**兩個**children (為甚麼不是一個，三個，…) 就也要跟著被cut?
- 這一切都是因為要維持如下質量：

k的degree的F-heap需要至少有(some base number)$^{\Theta(k)}$的節點個數，i.e., 指數數量的節點個數; here, $\Theta(k)=ak+b$ (where a=1, b=-2)

補充: O(k)是指小於等於ak+b這個等級的數量，而Θ(k)是指等於ak+b，這裡k可以任意大，但a,b是兩個常數

Fibonacci Number

- Fibonacci number: definition

$$F_0 = 0, \quad F_1 = 1,$$

and

$$F_n = F_{n-1} + F_{n-2}$$

for $n > 1$.

- Examples

F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}
0	1	1	2	3	5	8	13	21	34	55	89	144

- Closed from:

關鍵項

$$F_n = \frac{\varphi^n - \psi^n}{\varphi - \psi} = \frac{\varphi^n - \psi^n}{\sqrt{5}},$$

$-0.61803\,39887\ldots$

where

黃金比例

$$\varphi = \frac{1+\sqrt{5}}{2} \approx 1.61803\,39887\ldots$$

大約F_3項之後以黃金比例成長

F_0=0, F_1=1, F_2=1, F_3=2

Fn近似F_3*1.618^{n-3}，當n大於3

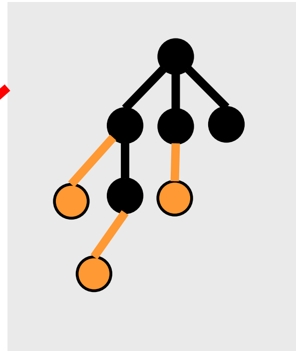
Fn指數成長

Why We Name it "Fibonacci" Heap?

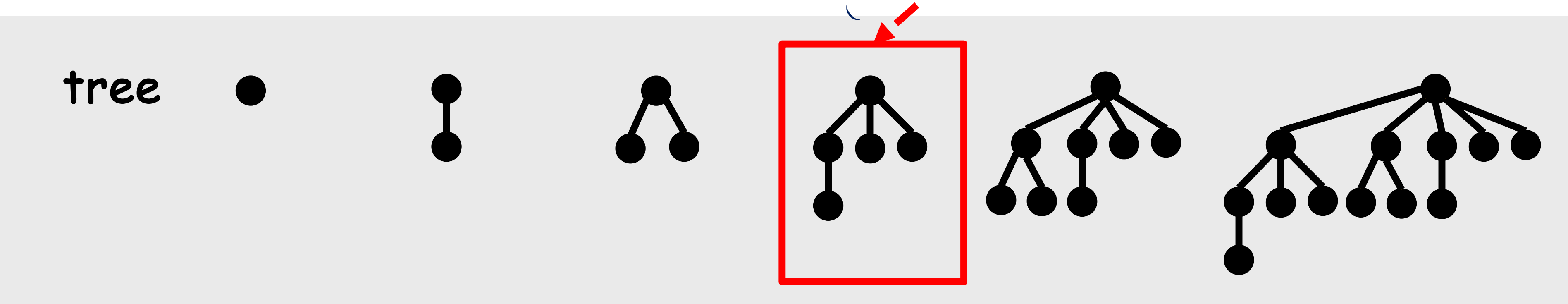
問：考慮一b-heap，當其經歷若干個delete (any), decrease key之後，則該b-heap (degree不變下) 所形成的min tree，**至少**會有幾個節點?

答：經歷上述操作，考慮degree分別為0, 1, 2, 3,...的min trees如下

例如：degree為3的b-heap (B_3), 執行上述動作前為

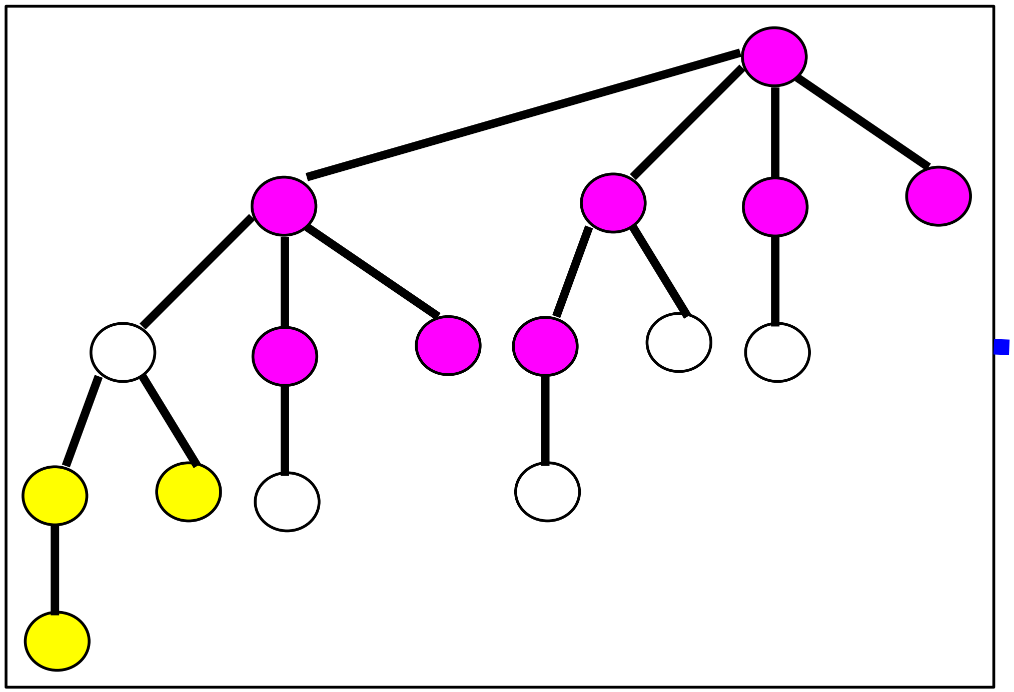


屬於這個家族
(“最多”移除三個黃色節點)

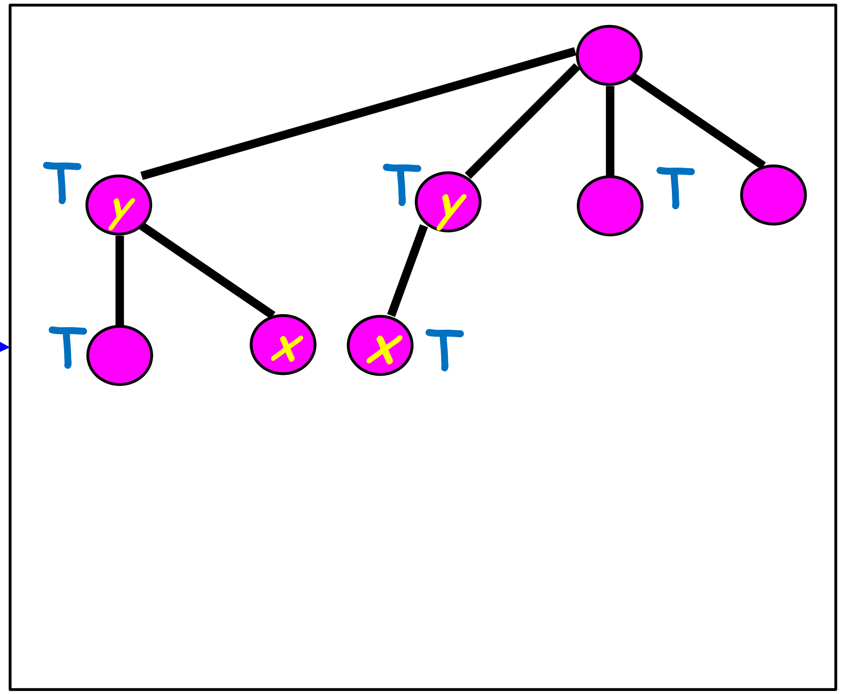


節點數	1	2	3	5	8	13
	F_2	F_3	F_4	F_5	F_6	F_7

再看一個F-Heap (starting from B4)
被移出若干個節點後的樣子...



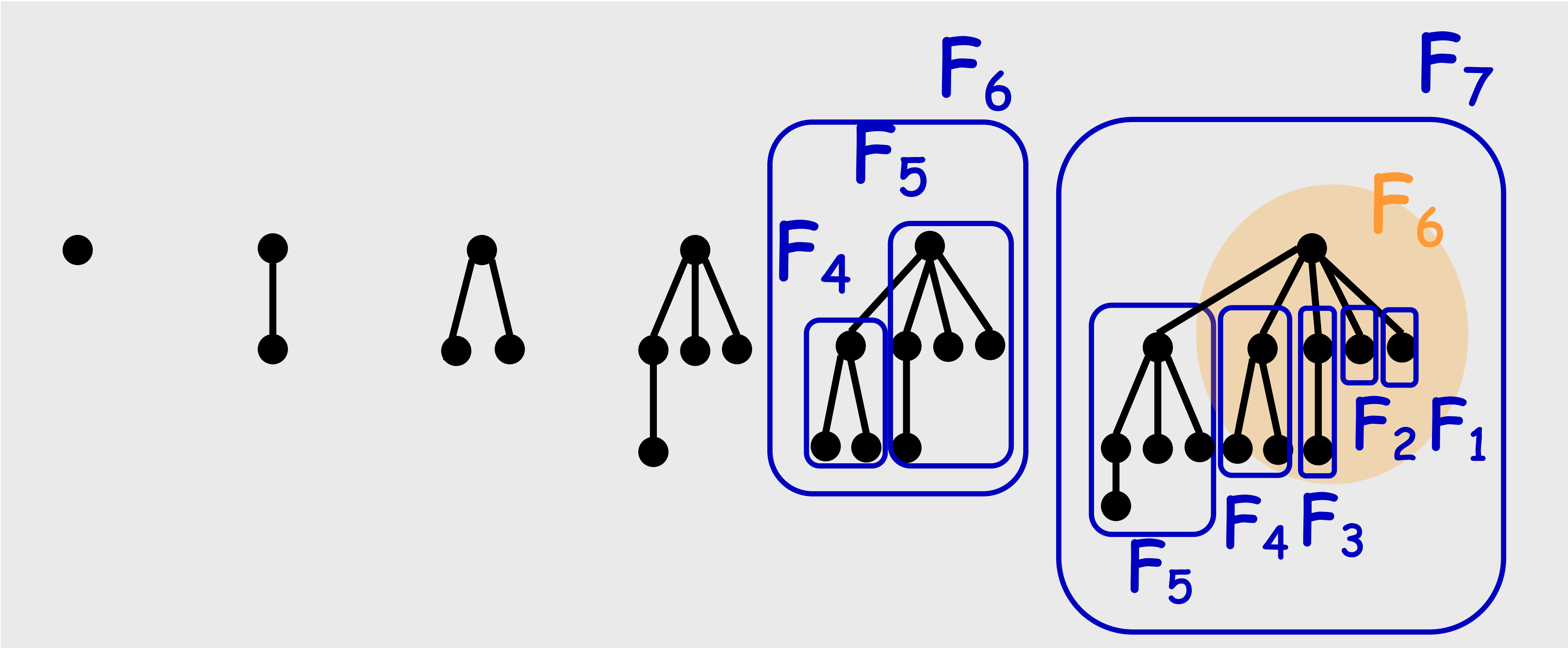
$\mathbf{B_4}$



$\mathbf{F_6}$

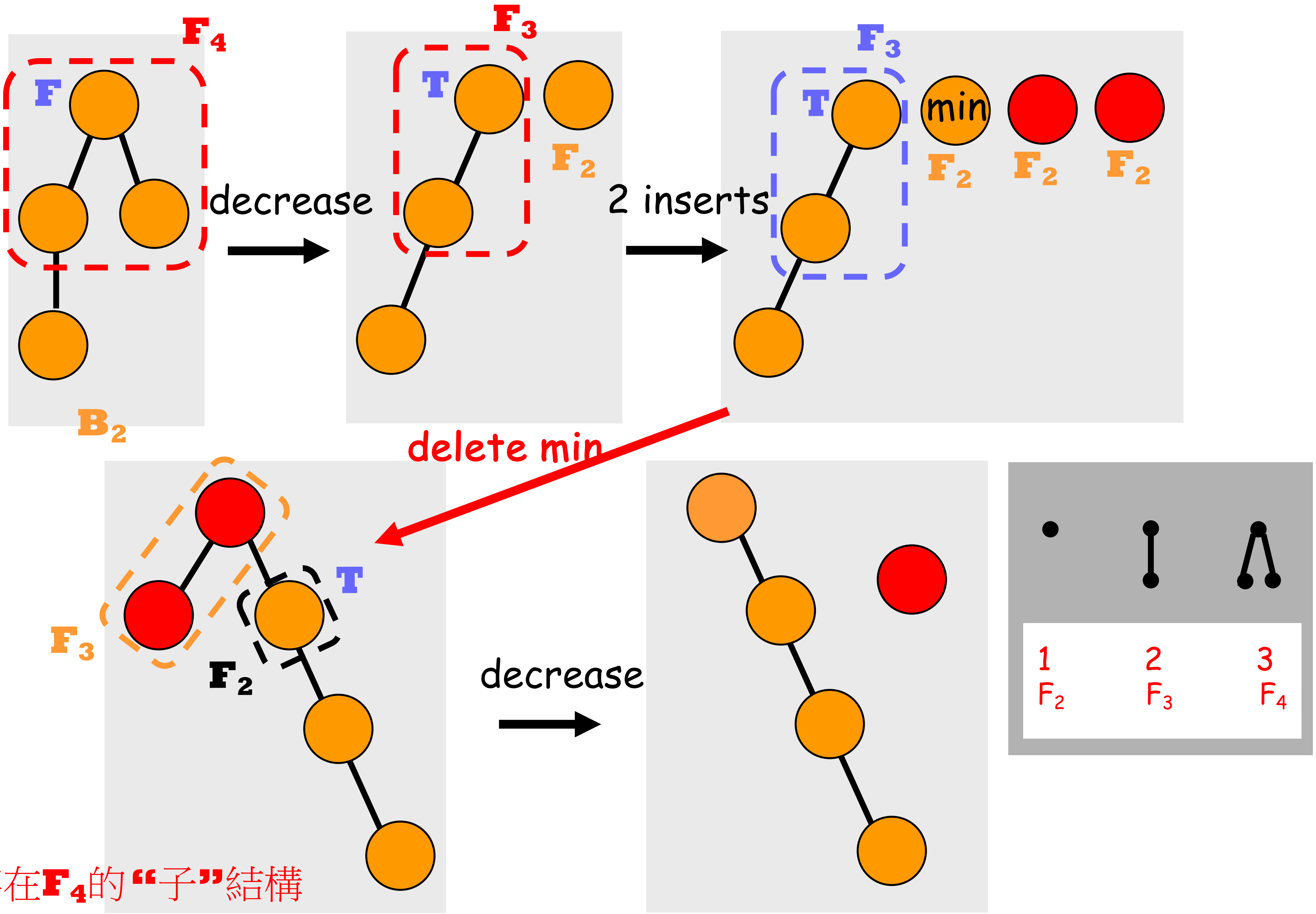
問：考慮一b-heap，當其經歷若干個delete (any), decrease key之後，則該b-heap (degree不變下) 所形成的min tree，至少會有幾個節點?

答：Fibonacci number



- Fibonacci numbers的特性:
  - $F_6=F_5+F_4$ (by definition)
  - $F_7=F_5+F_4+F_3+F_2+F_1+1$ (自行證明，目測見圖右)

近似B-Heap在節點“數量”上的質量



存在$\mathbf{F_4}$的“子”結構

Max Degree?

F_n近似F_3*1.618^{n-3}，當n大於3

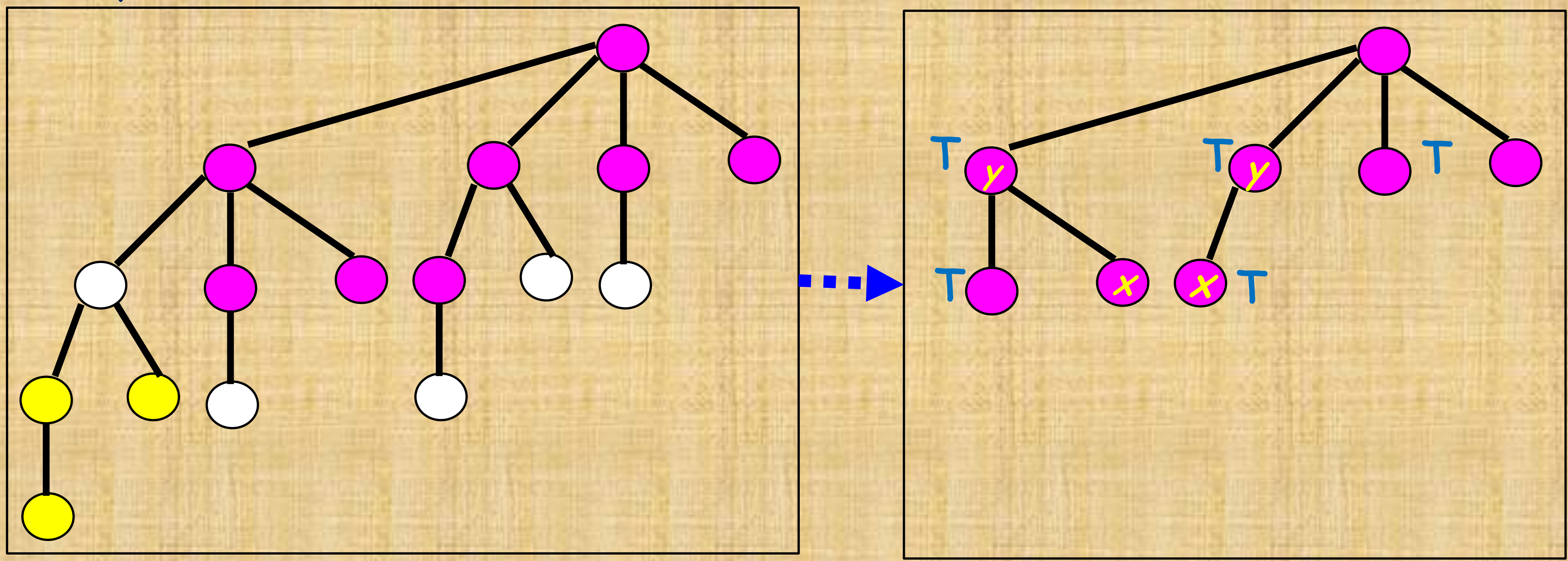
- …
- F_6的degree=4
- F_7的degree=5
- …

n=max degree最大多少?

  - O(log系統總節點的個數)

More on Cascading Cut

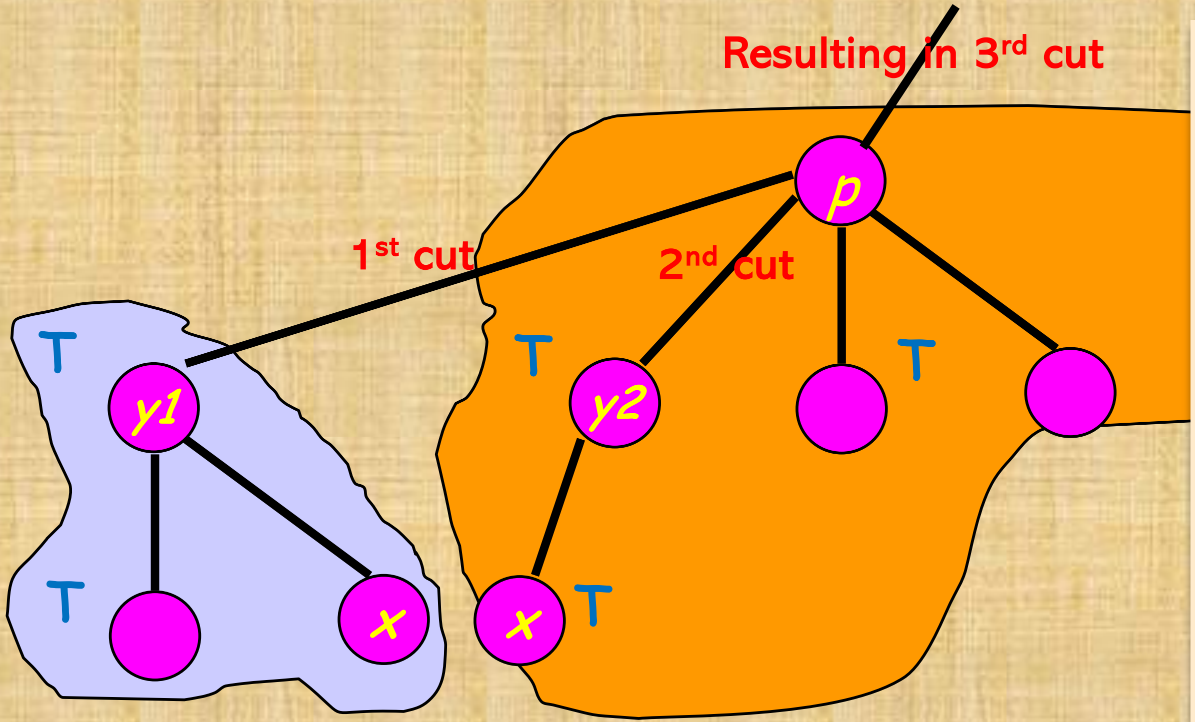
- F-Heap (starting from B4) 被移出若干個節點後的樣子...(下圖左)
- 下圖右，除了root以外的節點，**只要**當中任一個再被移出一次，則原來的F_6馬上變F_5  這個家族至少會有的節點數，例如從F_6移出x，則y也會相應移出，得到了至少F_5會有的節點數
- **目的：Cascading cut試圖控制每一個生成在top-level中$F_?$的節點數的質量，如下圖右，要不F_6家族要不F_5家族**，被移出的子樹也會是某一$F_?$家族



B_4 F_6

More on Cascading Cut (cont'd)

- When a parent p removes one of its child y1 (a subtree rooted at x), the subtree y2 of p are also cut if any p's descendant subtree (say, x) is further eliminated



為什麼cut兩次subtrees (y1 and y2) 後，就cut the entire p's subtree? "因為萬一最大的兩個p's subtrees被cut…則p剩餘的節點數將不足原來的1/2!"

Recall: $F_i \approx 1.618 F_{i-1}$ (for any i)
p的節點數$F_{k+1}=F_k+F_{k-1}$
y1的節點數F_{k-1}
y2的節點數F_{k-2}
y1+y2的節點數$F_k = F_{k-1}+F_{k-2}$

Consolidation due to Delete Min

- Recall: merge 2 Bi's with identical degree in B-heap, iteratively
- How about consolidation in F-heap?
  - Say, merge F_5 and F_4, resulting in F_6
    - Refer to "Fibonacci number def."
  - In fact, you may also merge 2 F_5's, resulting in something larger than F_6 (in terms of # of nodes)
    - Prior merging algorithm remains to work

Fibonacci Heaps
(B-Heaps are Special Cases of F-Heaps)

if consolidation

	Actual	Amortized
Insert	O(1)	O(1)
Delete min (or max)	O(n)	O(log n)
Meld	O(1)	O(1)
Delete any	O(n)	O(log n) // 同 delete min
Decrease key (or increase)	O(n) // due to cascading cut	O(1) // see P. 19 // 攤到n個meld上