

紅黑樹

Michael Tsai

2010/12/31 ← 2010最後一堂資料結構課

Happy New Year

- 作業六(最後一個作業, 耶)上線
- 期末考前due (跟助教會再討論)
- 期末考方式討論(close book, 2 A4雙面大抄)
- 範圍: 全部(第一堂課到最後一堂課)



紅黑樹

- 可以幹嘛?
- 是棵平衡的樹: 保證從root到某個leaf的simple path一定不會超過從root到任何一條其他這樣的path的兩倍長
- 大概平衡 \rightarrow operation可以都在 $O(\log n)$ 內完成. 耶.
- 那些operation?
 1. 找
 2. 插入某element
 3. 殺掉某element
 4. 找最大or找最小element

紅黑樹

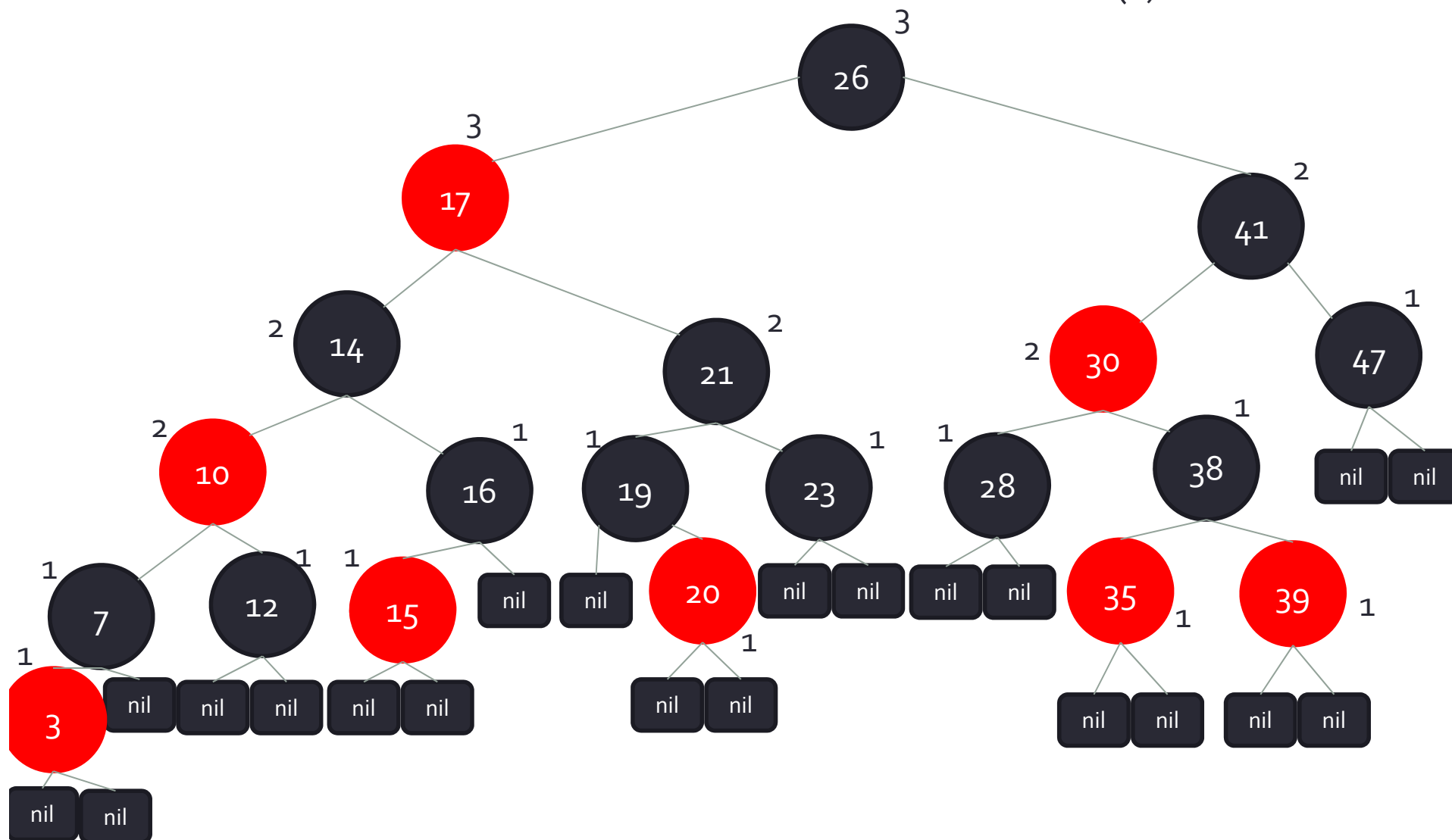
- 每個node都分配一個顏色: 紅或黑
- 使用extended binary tree: 沒有children的地方都補上external node, 又叫nil
- 規則們:
 - 1. 每個node不是黑就是紅
 - 2. root是黑色的
 - 3. 每個leaf (external node, or nil)都是黑色
 - 4. 如果一個node是紅的, 則它的children都是黑的
 - 5. 對每個node來說, 從它到他的所有子孫葉子node的路徑上含有一樣數目的黑色node

黑高度

- Black height: $bh(x)$ = 從 x 到任何一個它的子孫葉子node遇到的black node個數 (因為都一樣, 所以可以是任何一個)
- 不包含node x 自己
- external node或nil(葉子node)的black height為0

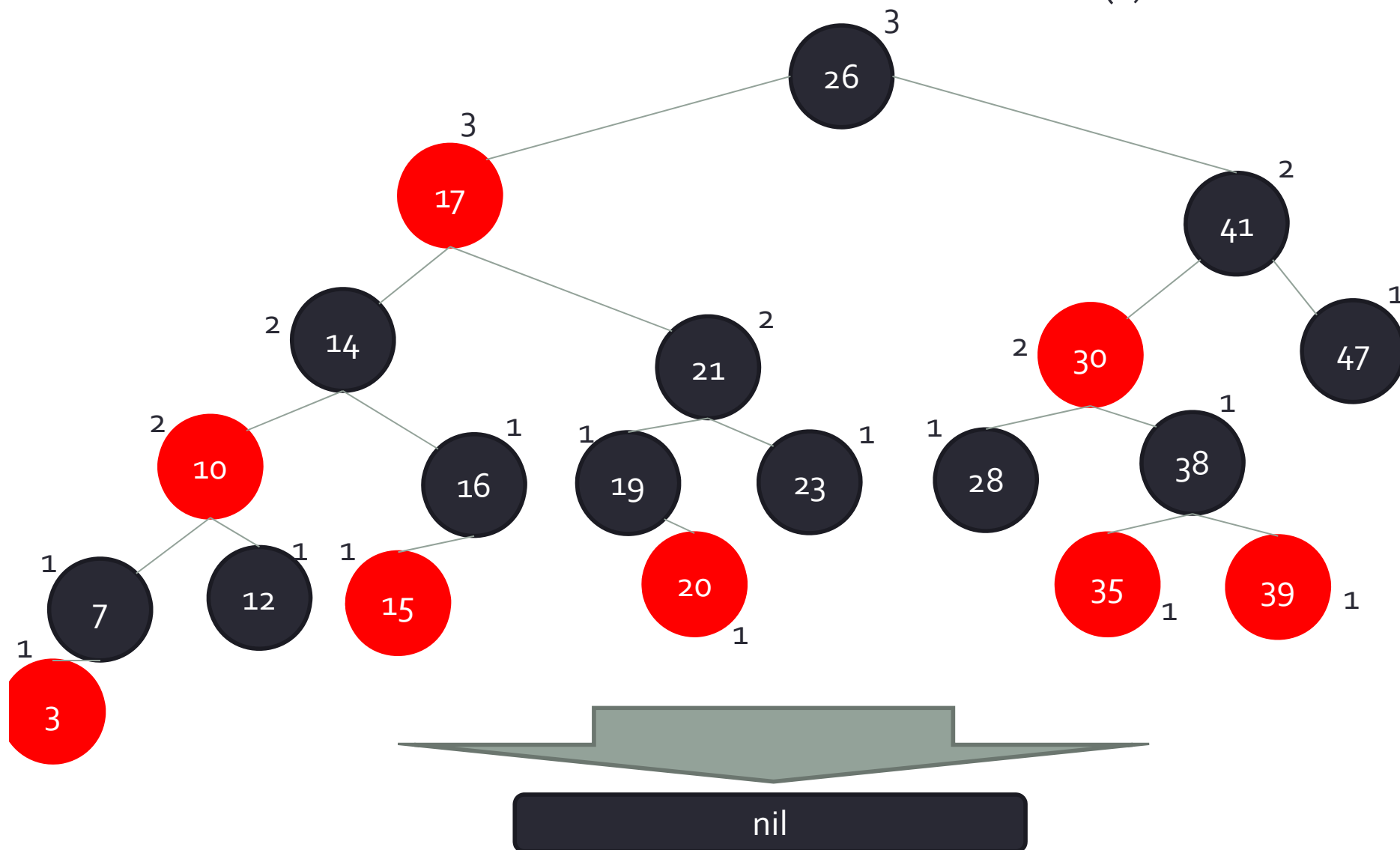
巨大的紅黑樹例子

bh(x)們



巨大的紅黑樹例子

bh(x)們



來點證明

- **定理:** 一個有 n 個node的紅黑樹, 最高為 $2 \log(n+1)$
- 第一步驟: 證明node x 底下的subtree最少有 $2^{bh(x)} - 1$ 個internal node
- 歸納法證明:
 - 1. 當 x 的height為0, 則 x 是個葉子. $bh(x)=0$. $2^0 - 1 = 0$. 的確subtree有0個internal node.
 - 2. 假設當 x 的height為正整數, 且是一個有兩個children的internal node. 每個小孩的black height為 $bh(x)$ or $bh(x)-1$. (看該小孩是黑是紅)
 - 假設小孩的subtree都至少有 $2^{bh(x)-1} - 1$ 個internal node
 - 3. 則 x 底下的subtree應該有
 - $(2^{bh(x)-1}-1) + (2^{bh(x)-1} - 1) + 1 = 2^{bh(x)} - 1$ (成功了!)

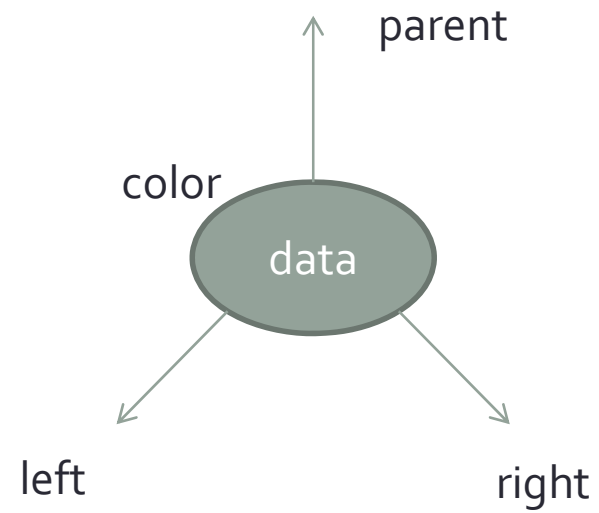
來點證明

- “4. 如果一個node是紅的, 則它的children都是黑的”
- 另外一種解釋:
- 從任一node到leaf, 至少一半以上的node是黑的
- 假設 h 是tree的高度, 則 $bh(root) \geq \frac{h}{2}$
- “node x 底下的subtree最少有 $2^{bh(x)} - 1$ 個internal node”
- 第二步驟: root底下至少有多少個node?
- root底下最少有
- $2^{bh(x)} - 1 \geq 2^{\frac{h}{2}} - 1$ 個node.
- 假設node個數為 n , 則 $n \geq 2^{\frac{h}{2}} - 1$
- $\rightarrow h \leq 2 \log(n + 1)$. 耶.

以上證明...

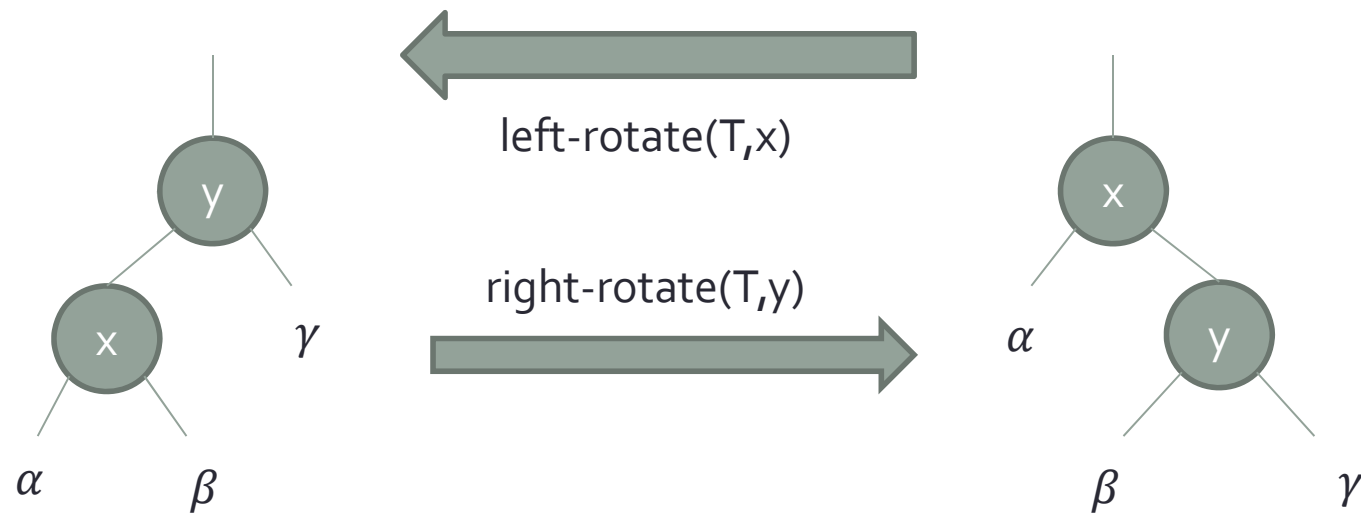
- 說明為什麼
- 1. 找
- 4. 找最大or找最小element
- 等operation可以在 $O(\log n)$ 內完成
- 那麼insert和delete呢?
- 問題: 插入或殺掉之後, 可能不滿足紅黑樹的條件

Representation




Rotate

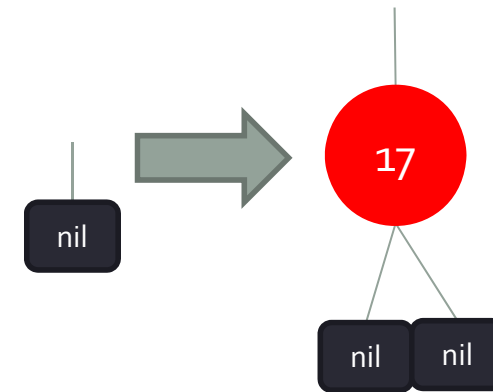
- 等一下會用到的



Insertion

- 首先, 用原本的binary search tree插入的方法
- insert(z) 
- 不同的地方:
 1. z的兩個children都指到nil node
 2. z為紅色
 3. 我們最後要處理不符合紅黑樹規則的部分
- 怎麼處理?

會違反那些規則呢?

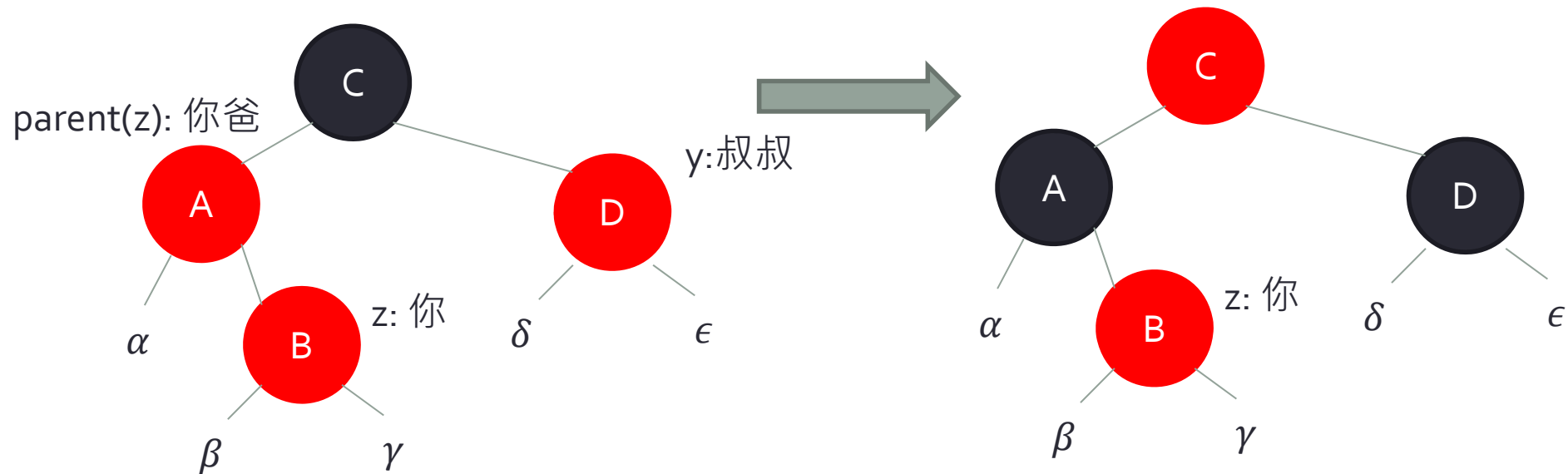


- 規則們:
- ~~1. 每個node不是黑就是紅~~
- 2. root是黑色的
- ~~3. 每個leaf (external node, or nil)都是黑色~~
- 4. 如果一個node是紅的, 則它的children都是黑的
- ~~5. 對每個node來說, 從它到他的所有子孫葉子node的路徑上含有一樣數目的黑色node~~
- 5. 不會違反因為z是紅的, z取代掉一個nil, 而z的兩個children都是nil
- 如果違反2, 則z是root, 整棵樹只有z: 很容易處理
- 來看違反4的情形: z's parent is also red

情形一：你的叔叔是紅的

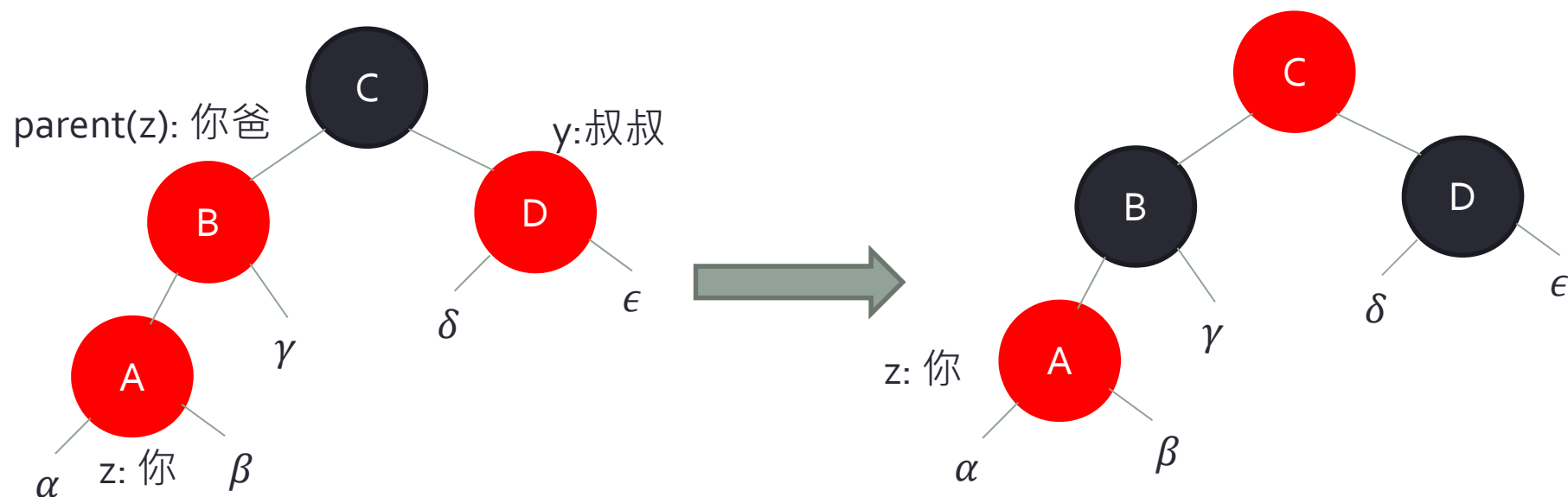
繼續看新z爸爸是不是紅的

你是你爸右邊的小孩



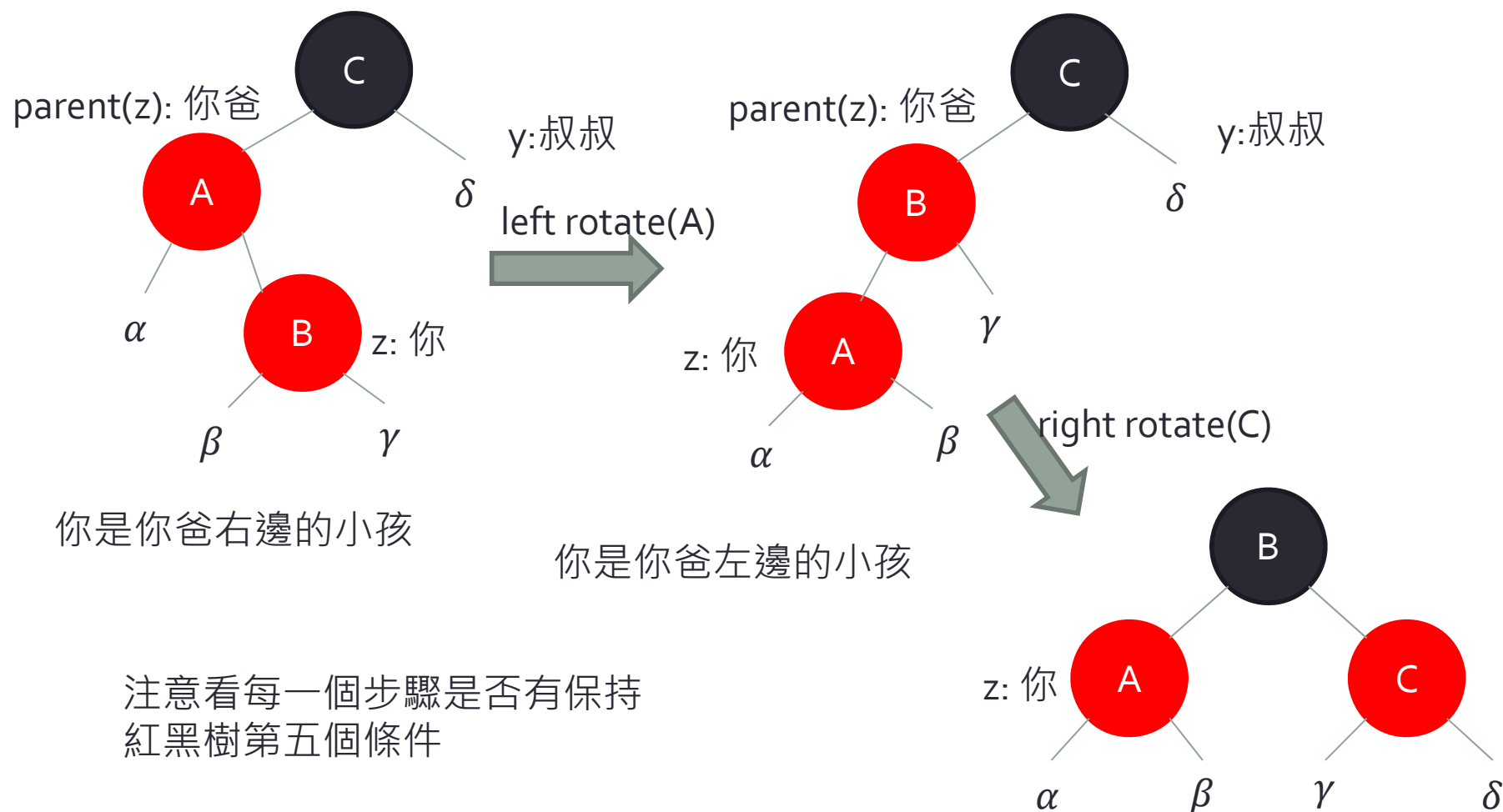
情形一：你的叔叔是紅的

你是你爸左邊的小孩



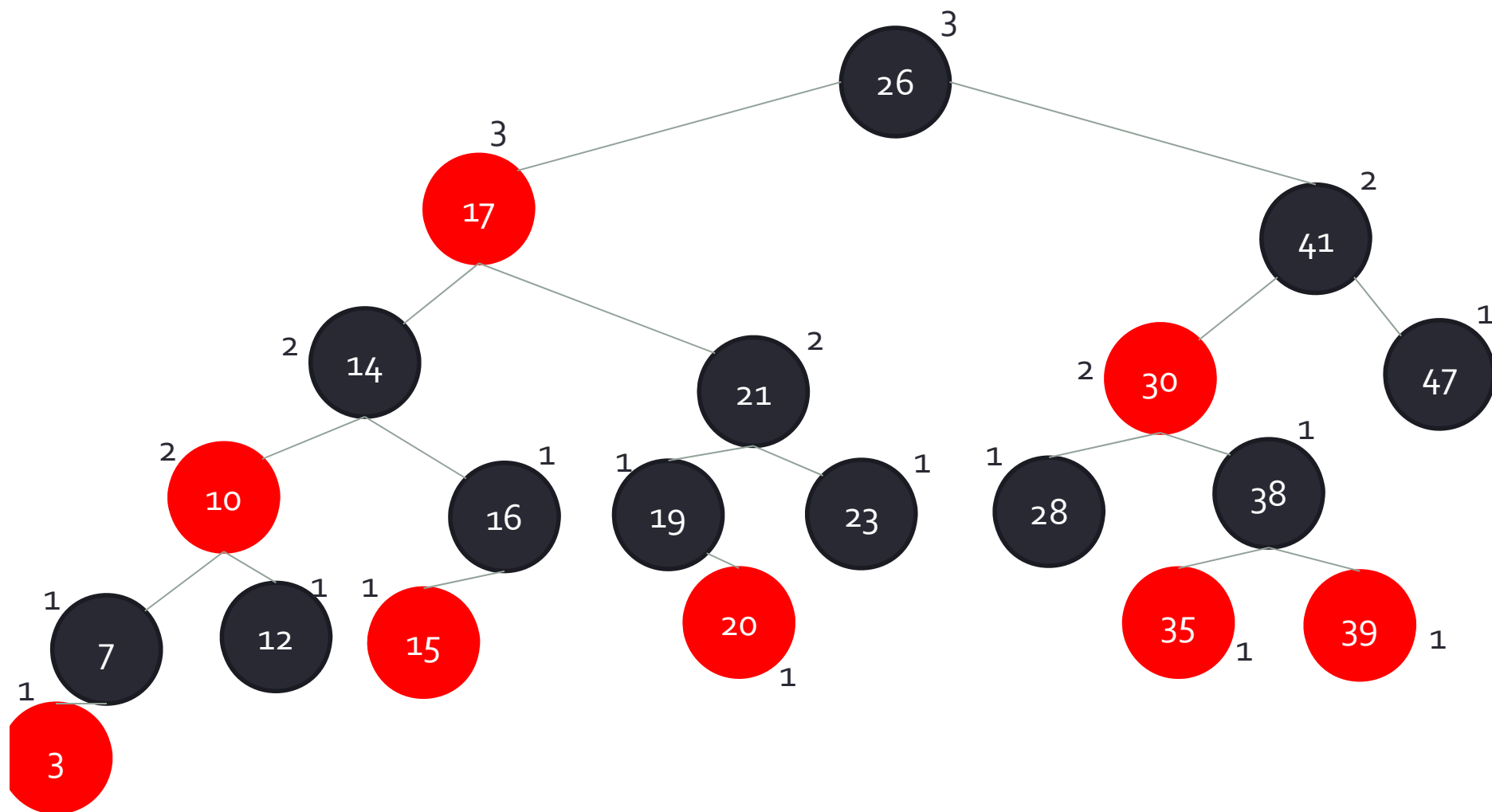
注意看每一個步驟是否有保持
紅黑樹第五個條件

情形二與三：你的叔叔是黑的



需要繼續往上層看嗎? 不用! 因為B還是黑的!

黑板練習時間



1. 加入40?

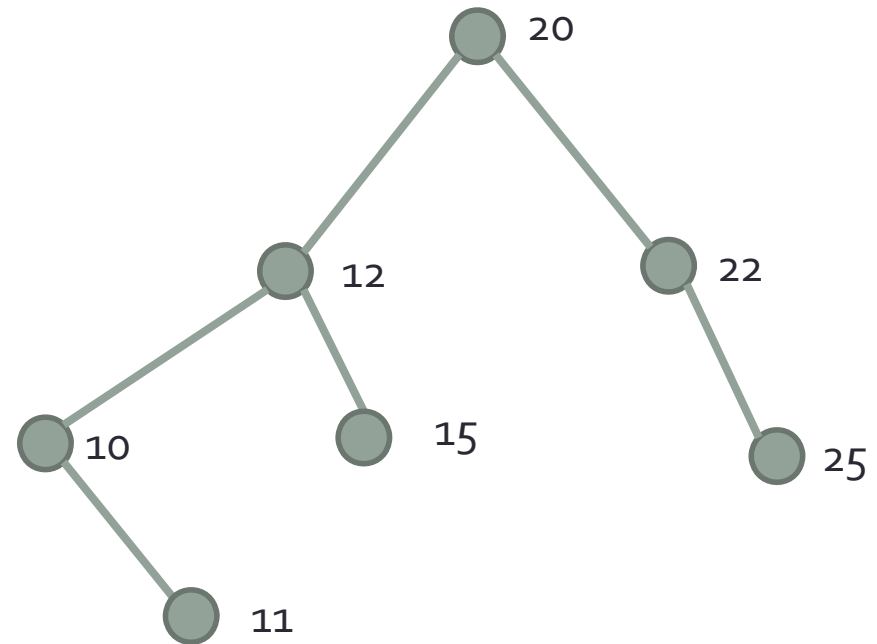
2. 加入4?

要花多少時間呢?

- 原本正常的binary tree insert要花 $O(\log n)$ 的時間
- 因為高度最高為 $2 \log(n+1)$
- 那麼花費在調整的時間呢?
- 最糟的狀況? 情形一——一直重複發生, 每次z往上移兩層
- 執行時間最糟要花跟高度成正比的時間
- 也是 $O(\log n)$
- 那如果發生情形二or三呢?
- 執行一次即完成. $O(1)$. (所以比 $O(\log n)$ 小)
- 另外, 最多rotate只需要執行兩次. (不會再發生第二次情形二or三)

如何刪掉一個node? (binary search tree 複習)

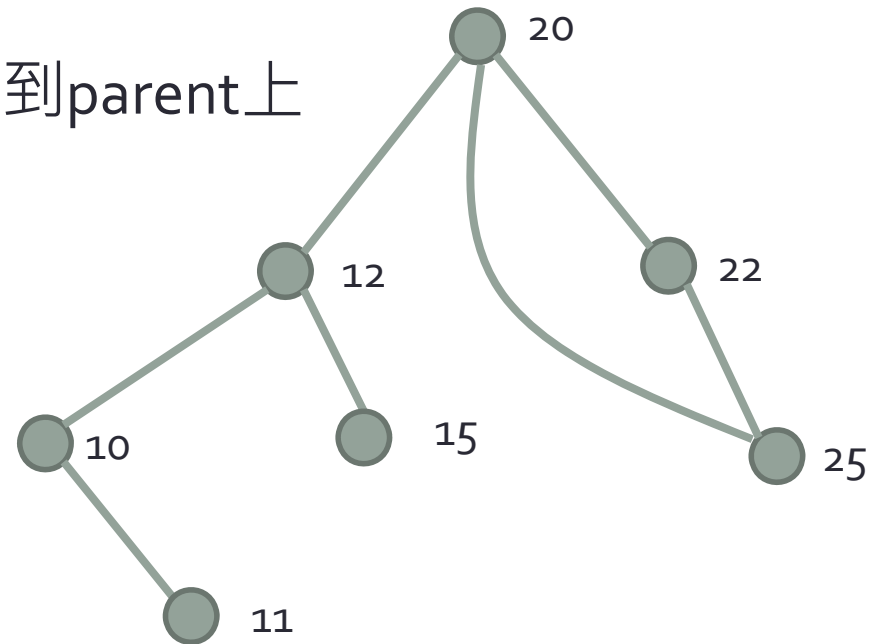
- 首先要先找到那個node
- 接著, 有各種不同情形:
- 如果沒有子(deg=0)
- 直接拿掉



如何刪掉一個node? (binary search tree 複習)

- 如果只有一隻手 (degree=1)
- 則把那個唯一的child拿上來接到parent上

- 例如: 拿掉22

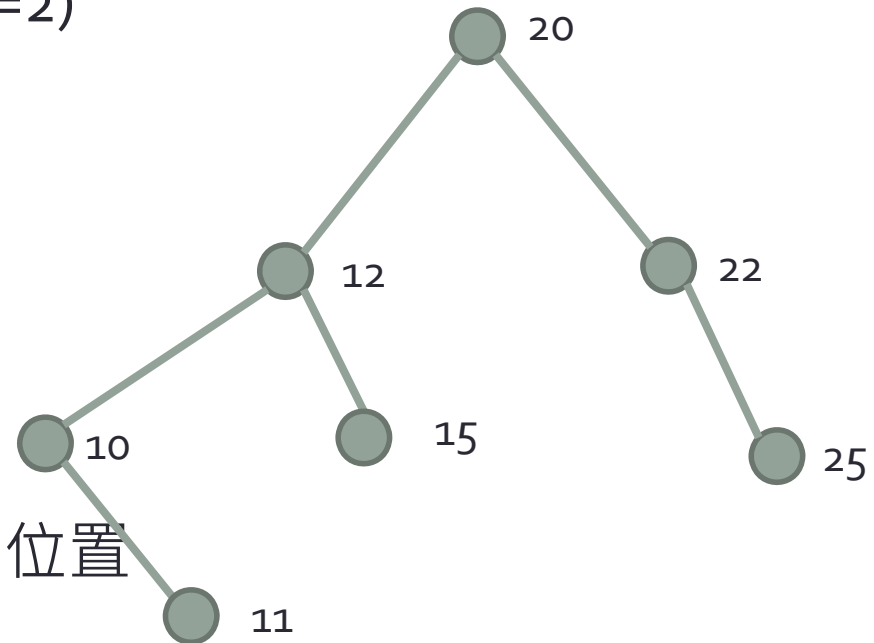


- 問題: 要怎麼接?
- 怎麼記得parent是誰?

如何刪掉一個node? (binary search tree 複習)

- 如果兩手都有東西呢? (degree=2)

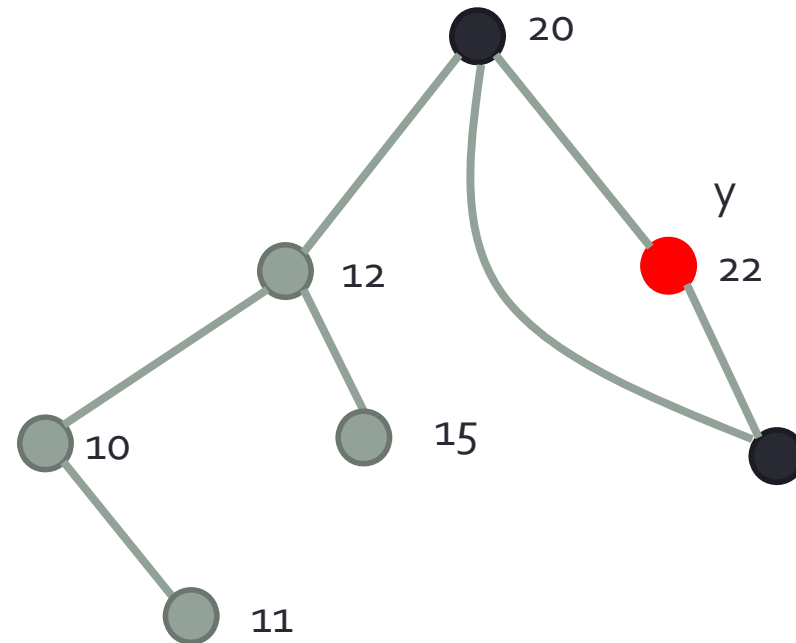
- 例如刪掉12
- 找左邊child底下最大的
- (或者是右邊child底下最小的)



- 刪掉它並把它移到原本刪掉的位置
- 問題: 那如果那個最大的底下還有child呢?
- 直接拿上來(最多只會有左邊一隻手)
- 這時被移上去的node顏色改變成新位置原本node的顏色

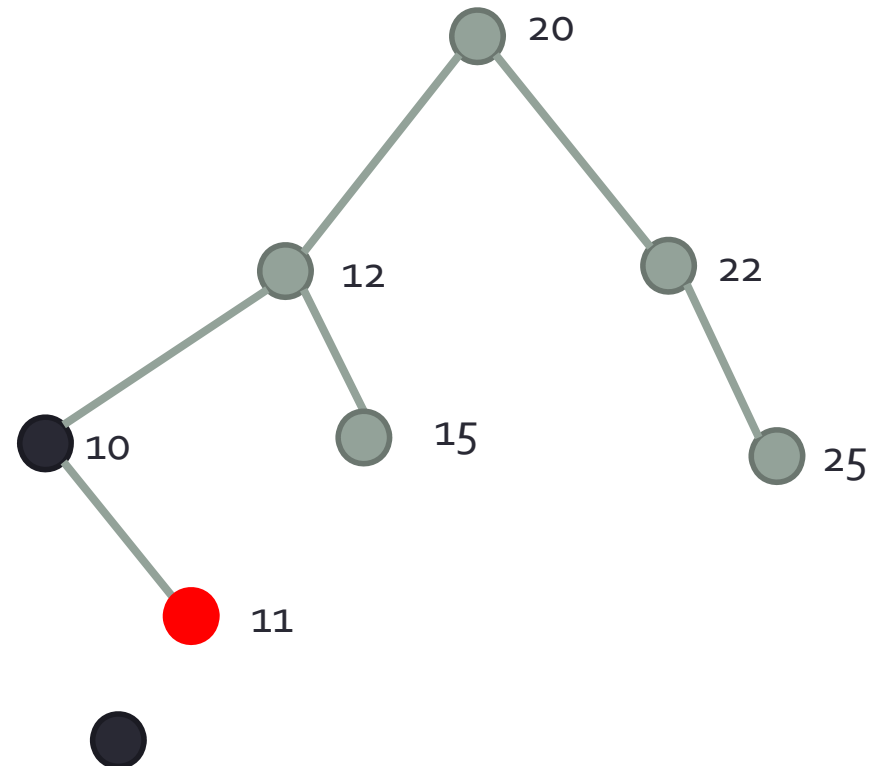
拿掉一個node，什麼時候會違反規則？

- 假設前述三種情形中，
- 移動(兩手都有)或是刪除(一隻手或沒有手)的node為y
- 當y為紅色(原本的颜色)，移動或刪除會造成違反規則嗎？
 1. black height不會改變，因為y是紅色node
 2. 會不會造成兩個紅色node是相鄰的呢？(父子)
 - A. 如果y是被刪掉的，因為它是紅的，所以它的上下層都是黑的
- 不會有問題



拿掉一個node，什麼時候會違反規則？

- B. 如果y是被移動的, 假設y有children也是黑色的, 不會造成問題
- 3. y如果是紅色, 它不會是root. 所以也不會有造成違反root是黑色的規定
- 綜合以上三點, 只有當y為黑色時才會造成問題, 需要調整



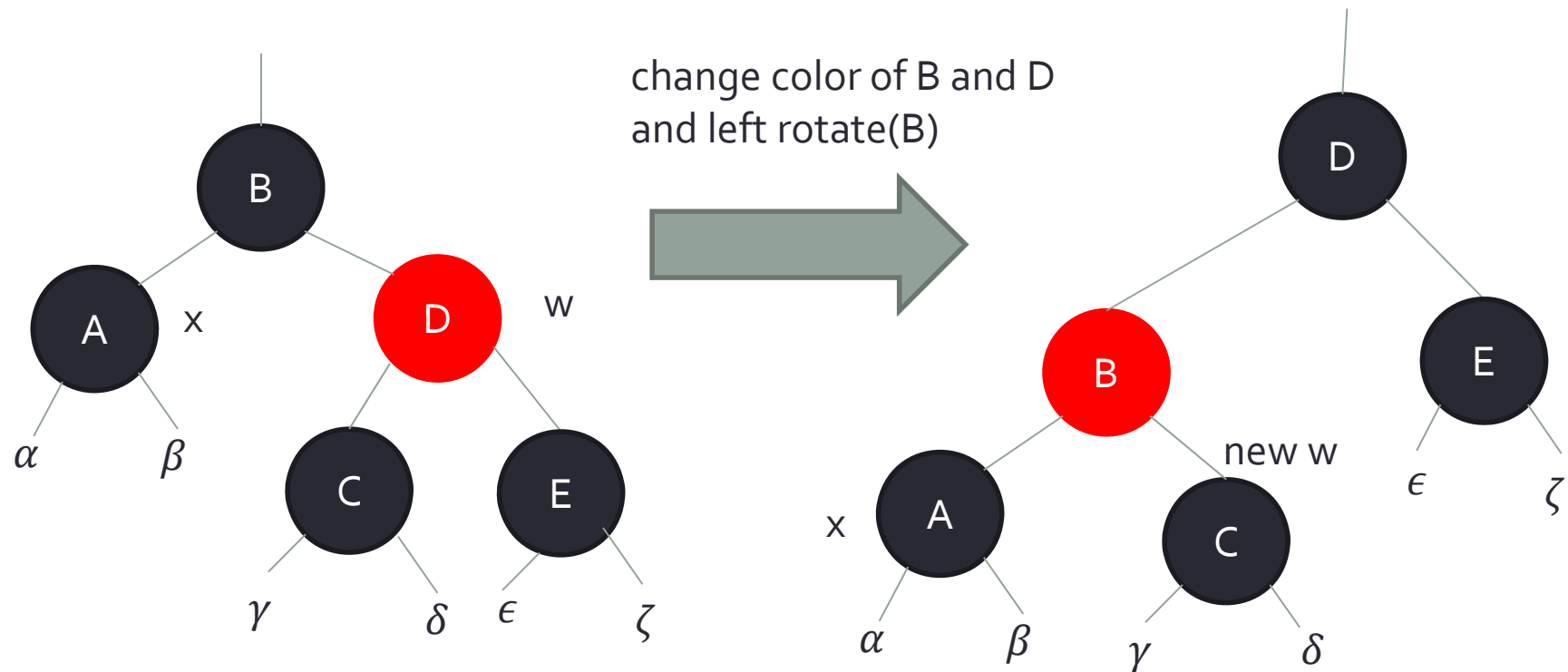
可能違反規則的情形

- 當 y 是黑色的, 可能造成違反規則的有下列情形:
- 1. y 是root. y 刪掉以後, 它的一個children是紅色的, 變成了root
- 2. y 原本的上下兩層都是紅色的, 移走或刪除以後變成兩個紅色相鄰
- 3. y 刪掉或移走以後, 造成black height不一致(因為 y 是黑的)

怎麼修正紅黑樹以符合規則呢？

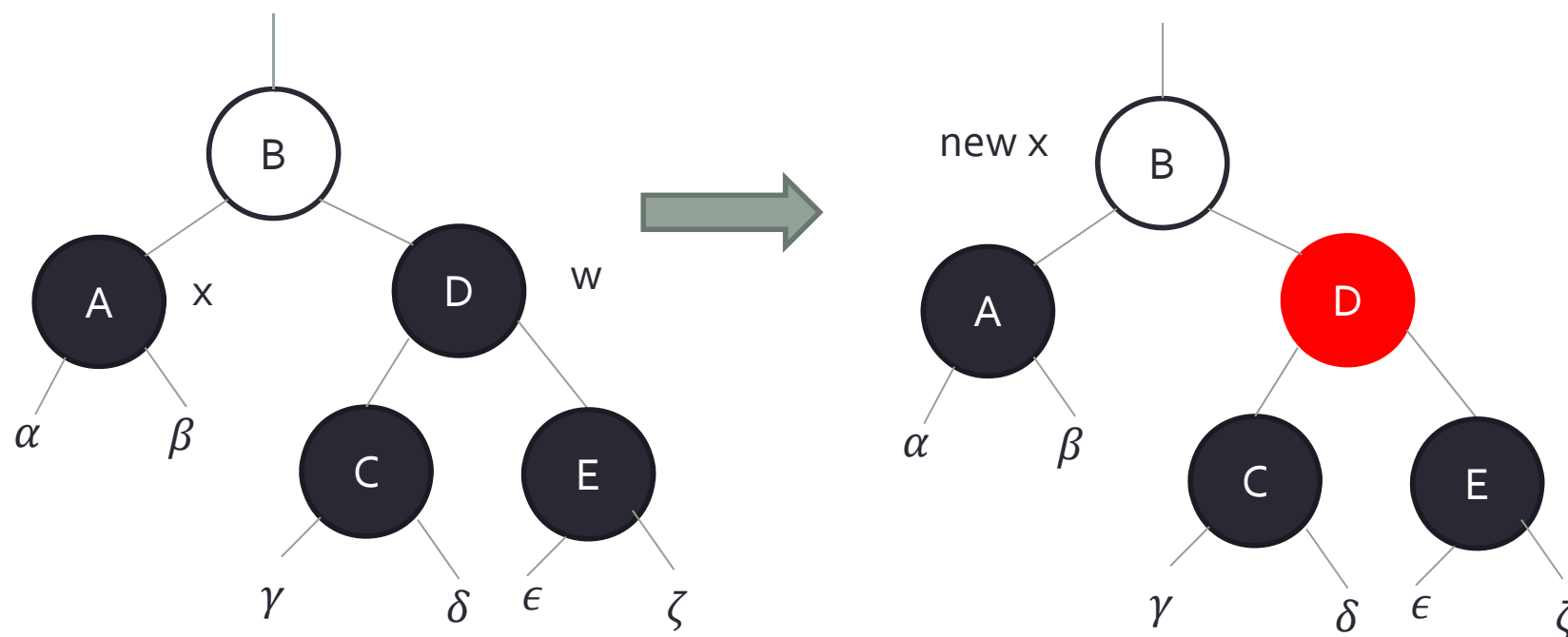
- 假設x為移動到y位置的node
- (如果忘記了的話, y是被刪掉或被移動的node)
- y原本是黑的, 則x好像要“有兩個黑”, 這樣black height才會正確
- 每次x都指到一個“兩個黑”的node (或者也有可能是“紅與黑”)
- 依序修正

情形一：x的弟弟是紅的

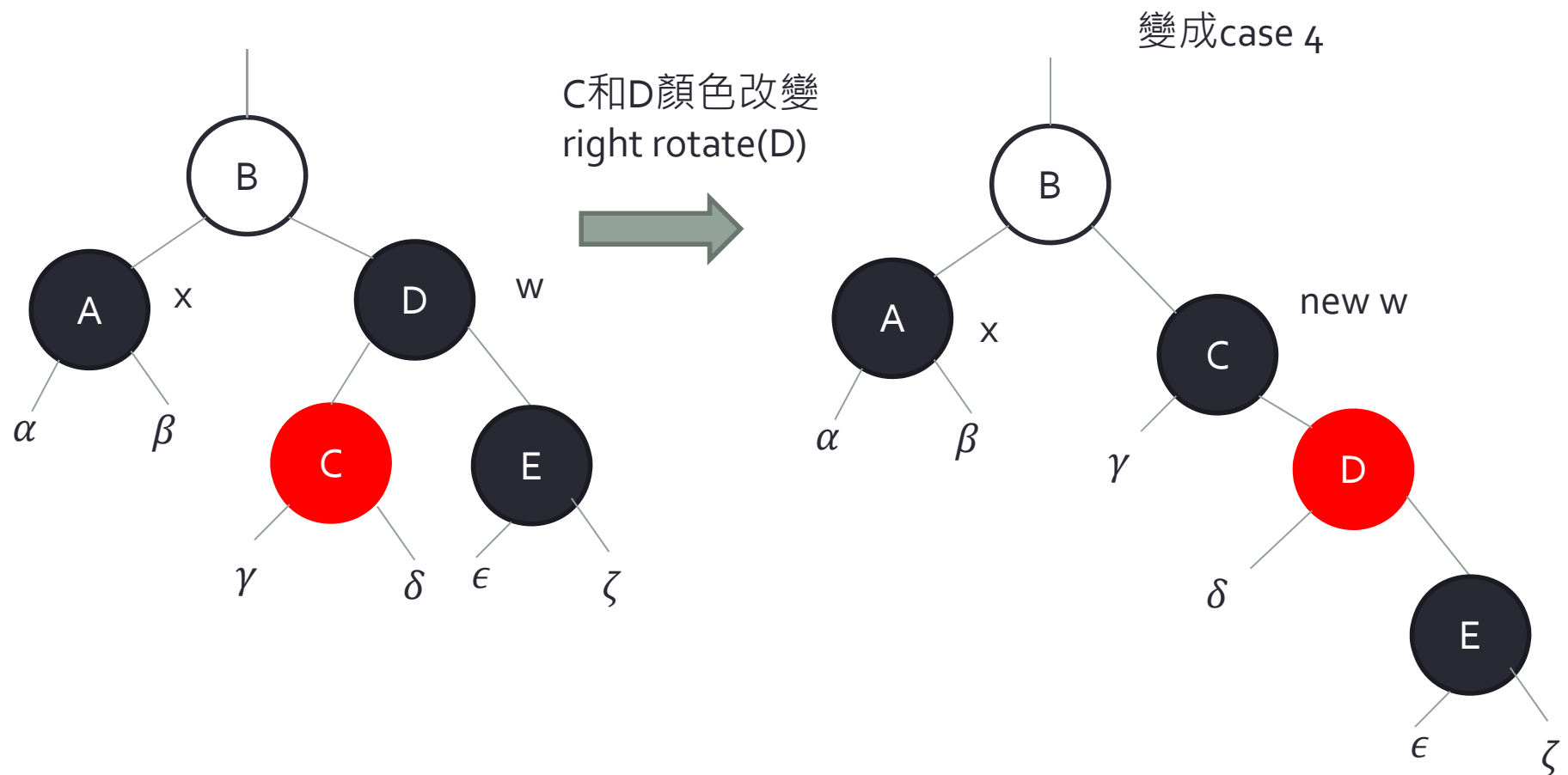


轉換成情形二、三、或四

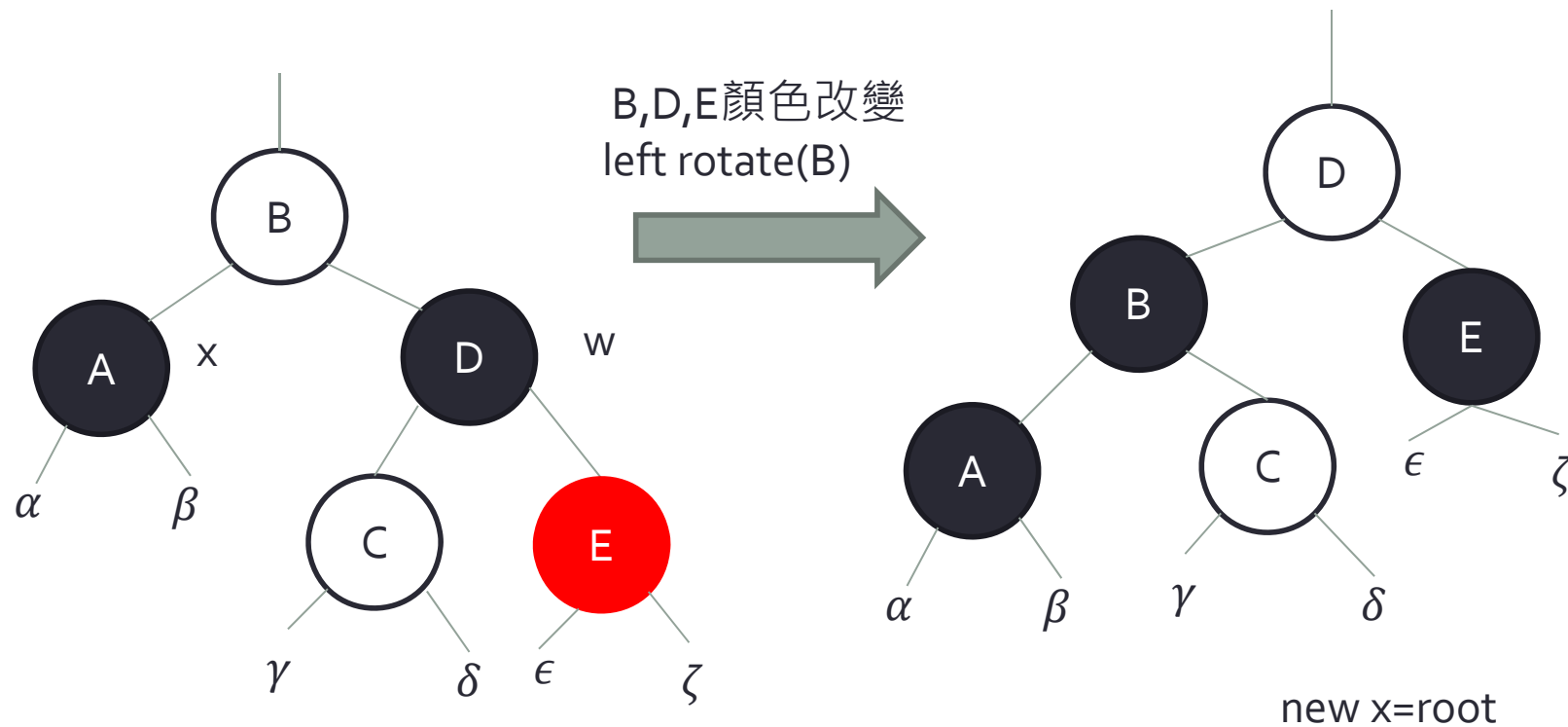
情形二：
x的弟弟是黑的&它的姪子們都是黑的



情形三：x的弟弟是黑的 它的姪子們都是左紅右黑



情形四：x的弟弟是黑的
它的姪子右邊那一個是紅的



花費的時間

- $O(\log n)$
- 最糟的情形是情形二
- 每次往上移一層 \rightarrow 最糟 $O(\log n)$
- 其他情形都是 $O(1)$

Happy New Year

- 老師的叮嚀
- 不要爆肝
- 不要不睡覺
- 身體最重要
- 心情快樂最重要
- 明年再見~~~

