

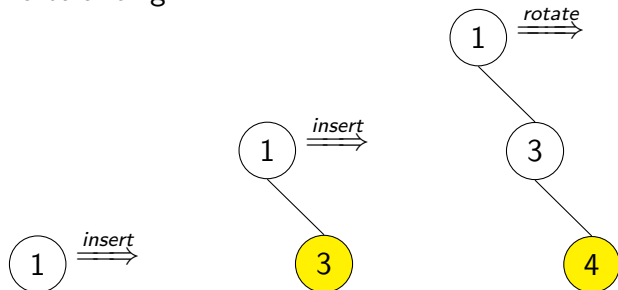
Homework 6

林宏陽、周若涓、曾守瑜

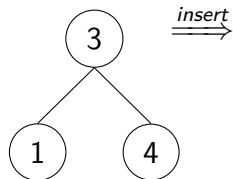
Problem1

Perform insertions of the numbers 1, 3, 4, 7, 6, 5, 2 (in this order) into an empty AVL tree. Show each AVL tree after a number has been inserted.

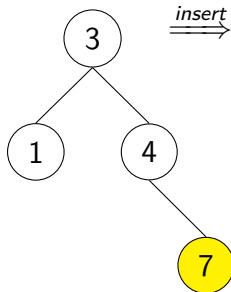
If re-balancing operations are performed, please also show the tree before re-balancing and indicate what type of rotation is used in the re-balancing.



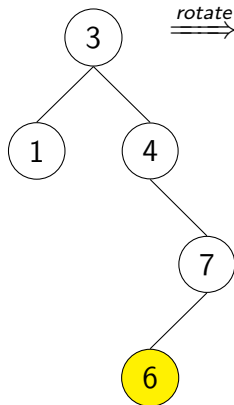
Problem1



insert \Rightarrow

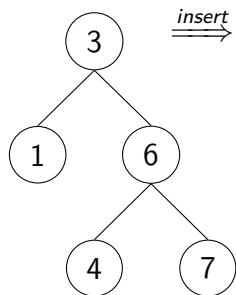


insert \Rightarrow

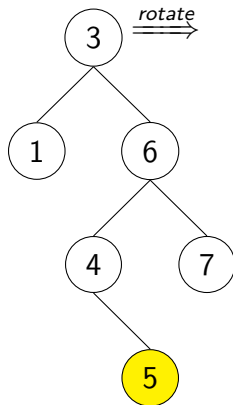


rotate \Rightarrow

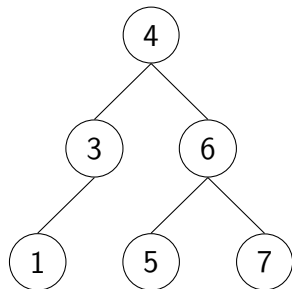
Problem1



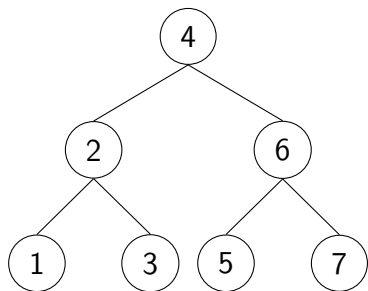
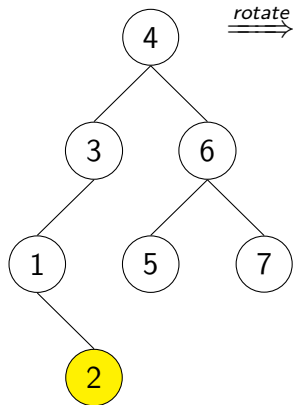
insert \Rightarrow



rotate \Rightarrow



Problem1



Problem2

(6.32) Prove that the sum of the heights of all nodes in a complete binary tree with n nodes is at most $n - 1$.

(A complete binary tree with n nodes is one that can be compactly represented by an array A of size n , where the root is stored in $A[1]$ and the left and the right children of $A[i]$, $1 \leq i \leq \lfloor \frac{x}{2} \rfloor$, are stored respectively in $A[2i]$ and $A[2i + 1]$.

Notice that, in Manber's book a complete binary tree is referred to as a balanced binary tree and a full binary tree as a complete binary tree. Manber's definitions seem to be less frequently used.

Do not let the different names confuse you. "Balanced binary tree" in the original problem description is the same as "complete binary tree")

Problem2

- h 表示樹的高度， $G(h)$ 表示在 h 高度下，每個 node 的高度總和
- For a full binary tree (complete binary trees 的一個特例),
node(n) 數量 = $2^{h+1} - 1$
 $G(h) = 2^{h+1} - h - 2 = (2^{h+1} - 1) - h - 1$
 $= (2^{h+1} - 1) - (h + 1) = n - (h + 1) \leq n - 1$
(proven in homework 2)
- With this as a basis, we prove the general case of complete binary trees by induction on the height $h (\geq 0)$ of the tree.

Problem2

- [Base Case]($h=0, n=1$): $G(h)=0 \leq 1-1$
- [Inductive step]
[Induction Hypothesis] The height of sum of complete binary is $G(h) \leq n-1$

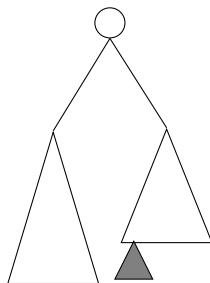
There will be four cases on complete binary tree

- ▶ [Case1] Full binary tree (we have proven this case before)
- ▶ [Case2] Full left subtree and complete right subtree (both height= h)
- ▶ [Case3] Both Left and right subtree are full ($h_L = h$ and $h_R = h-1$)
- ▶ [Case4] Full right subtree and complete left subtree ($h_L = h$ and $h_R = h-1$)

Problem2

[Case2] Full left subtree and complete right subtree(both height= h)

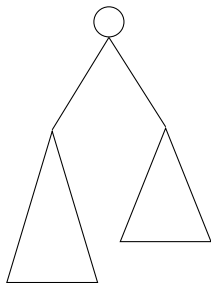
$$\begin{aligned}G_c(h+1) &= G(h_L) + G_c(h_R) + (h+1) \\ &= [n_L - (h+1)] + G_c(h_R) + (h+1) \\ &\leq [n_L - (h+1)] + [n_R - 1] + (h+1) \\ &= n_L + n_R - 1 \\ &= (n_L + n_R + 1) - 2 \\ &= n - 2 \leq n - 1\end{aligned}$$



Problem2

[Case3] Both Left and right subtree are full ($h_L = h$ and $h_R = h-1$)

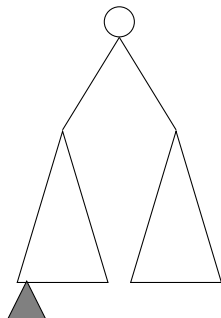
$$\begin{aligned}G_c(h+1) &= G(h_L) + G(h_R) + (h+1) \\ &= G(h) + G(h-1) + (h+1) \\ &= [n_L - (h+1)] + [n_R - (h-1+1)] + (h+1) \\ &= (n_L + n_R + 1) - h - 1 \\ &= n - (h+1) \leq n-1\end{aligned}$$



Problem2

[Case4] Full right subtree and complete left subtree($h_L = h$ and $h_R = h-1$)

$$\begin{aligned}G_c(h+1) &= G_c(h_L) + G(h_R) + (h+1) \\ &= G_c(h_L) + G(h-1) + (h+1) \\ &= G_c(h_L) + [n_R - (h-1+1)] + (h+1) \\ &\leq [n_L - 1] + [n_R - (h-1+1)] + (h+1) \\ &= n_L - 1 + n_R + 1 \\ &= (n_L + n_R + 1) - 1 \\ &= n - 1 \leq n - 1\end{aligned}$$



Problem3

這裡有 12 個硬幣，裡面有最多一個偽幣
偽幣重量和真的不一樣
給你一個天平
用三次秤重就判斷誰是假的，或大家都是真的

Problem3

善用決策樹

每次秤重，天平會給我們哪幾種資訊？

一樣重、不一樣重：兩種？

$$2^3 = 8 < 12 + 1$$

一樣重、左邊重、右邊重：三種

$$3^3 = 27 > 12 + 1$$

Problem3

從頭想解法不容易
善用決策樹的節點數能少走彎路

一開始想要左六右六秤？

一樣重：完美；某邊重？

因為不知道偽幣是比較重還是比較輕，沒有硬幣能擺脫嫌疑

此時仍有 12 種可能，但你只剩兩次比較， $3^2 = 9 < 12$

左五右五也類似

Problem 3

參考解法

首先左四右四秤，比如， $c_1 c_2 c_3 c_4 / c_5 c_6 c_7 c_8$

如果一樣重，這 8 個硬幣擺脫嫌疑

剩下 4 個硬幣，拿 $c_9 c_{10} c_{11}$ 出來與任三個真幣秤，比如

$c_9 c_{10} c_{11} / c_1 c_2 c_3$

如果這兩邊一樣重，剩下 c_{12} 有嫌疑，比一下就知道了

如果 $c_9 c_{10} c_{11}$ 比較輕/重，不但知道偽幣存在，還知道偽幣是比較輕/重

比 c_9 / c_{10} ，比較輕/重一方就是假的；若兩邊一樣就是 c_{11}

Problem 3

如果 $c_1 c_2 c_3 c_4 / c_5 c_6 c_7 c_8$ 不一樣重？

此時我們確定裡頭有偽幣，嫌疑人 8 位

我們要在剩下兩次比較解析出兇手

第二次秤重必須將 8 個嫌疑人分成三部分，3/3/2，才有機會在最後一次秤重找到兇手

將 $c_6 c_7 c_8$ 移到左方，右邊補上三個真幣 $c_9 c_{10} c_{11}$

$c_1 c_6 c_7 c_8 / c_5 c_9 c_{10} c_{11}$

如果傾斜的方向一樣，代表問題出在 $c_1 c_5$ ，拿 c_1 與真幣秤就行

如果兩邊一樣重，代表問題在 $c_2 c_3 c_4$ ；如果傾斜方向反了，問題在 $c_6 c_7 c_8$

而且能夠從第一次秤重的結果判斷偽幣輕/重

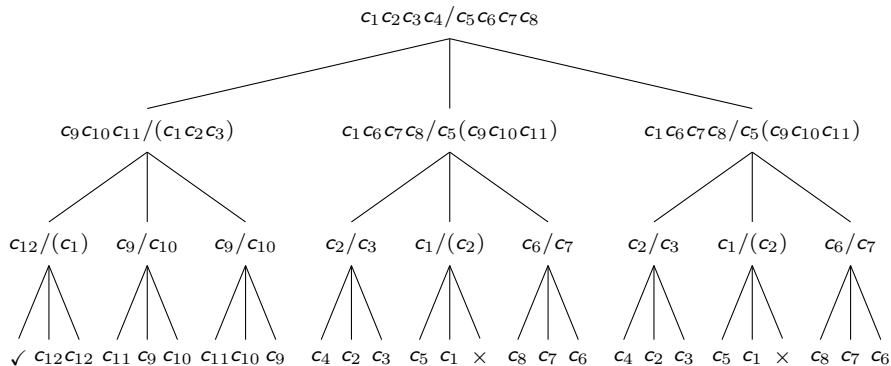
剩下的三個嫌疑人其中兩個互相秤，輕/重的一方為假，否則第三個是假

Problem 3

為了更清楚展示這個流程，畫出天平秤重的決策樹：

三個子節點分別代表：一樣重/左邊重/右邊重

括號代表當下已經確認為真貨的硬幣



Problem3

為什麼不能用 2 次比較？

因為 $3^2 = 9 < 12 + 1$

可能性只有 13 種嗎？

更好的答案是： $3^2 = 9 < 24 + 1$

由於這個問題只要我們判斷「誰是假的」，所以只需要 13 個節點
但解題過程中用到的天平使得我們依賴於偽幣的輕重，而一開始
這個訊息是未知的

我們需要兩倍：25 個節點來完整表達：誰是偽幣、比較輕還是比較重

無論是 13 還是 25，都無法用兩次比較達成

Problem4

Design an algorithm that, given a set of integers $S = \{x_1, x_2, \dots, x_n\}$, finds a nonempty subset $R \subseteq S$, such that

$$\sum_{x_i \in R} x_i \equiv 0 \pmod{n}.$$

Before presenting your algorithm, please argue why such a nonempty subset must exist.

Problem4(cont'd)

Define

$$Sum_1 = x_1$$

$$Sum_2 = x_1 + x_2$$

$$Sum_3 = x_1 + x_2 + x_3$$

...

$$Sum_n = x_1 + x_2 + x_3 + \cdots + x_n$$

If there exists an $Sum_i \equiv 0 \pmod{n}$ then done. If no, there must be at least two of them have the same remainder because there are n items, $n - 1$ kinds of remainders (from 1 to $n - 1$), by pigeonhole principle. Suppose Sum_i and Sum_j ($i < j$) have the same remainder, then $\{x_{i+1}, x_{i+2}, \cdots, x_j\}$ is what we want since we remove the elements from $\{x_1, x_2, \cdots, x_j\}$, which cause the remainder.

$\{x_{i+1}, x_{i+2}, \cdots, x_j\}$ cannot be empty because $i \neq j$.

Problem4(cont'd)

function FIND SUBSET(S, n)

$R_1 := x_1 \bmod n$ $\triangleright R$ represents the remainder of Sum

for i from 1 to $n - 1$ **do**

$R_i := (R_{i-1} + x_i) \bmod n$ \triangleright Compute $Sum_i \bmod n$

if $R_i == 0$ **then**

 print $\{x_1, x_2, \dots, x_i\}$

 return

$i, j := \text{FindIdentical}(R)$ \triangleright Get index of two items with the same remainder

print $\{x_{i+1}, x_{i+2}, \dots, x_j\}$

Problem5

Consider the *next* table as in the KMP algorithm for string $B[1..9] = \text{abaababaa}$.

1	2	3	4	5	6	7	8	9
a	b	a	a	b	a	b	a	a
-1	0	0	1	1	2	3	2	3

Suppose that, during an execution of the KMP algorithm, $B[6]$ (which is an a) is being compared with a letter in A , say $A[j]$, which is not an a and so the matching fails. The algorithm will next try to compare $B[\text{next}[6] + 1]$, i.e., $B[3]$ which is also an a , with $A[j]$. The matching is bound to fail for the same reason. This comparison could have been avoided, as we know from B itself that $B[6]$ equals $B[3]$ and, if $B[6]$ does not match $A[j]$, then $B[3]$ certainly will not, either. $B[5]$, $B[8]$, and $B[9]$ all have the same problem, but $B[7]$ does not. Please adapt the computation of the next table so that such wasted comparisons can be avoided.

Problem5(cont'd)

```
function NEW NEXT( $B, m, next$ )  
  for  $i$  from 3 to  $m$  do  
    if  $B[next[i] + 1] == B[i]$  then  
       $next[i] := next[next[i] + 1]$ 
```

1	2	3	4	5	6	7	8	9
a	b	a	a	b	a	b	a	a
-1	0	-1	1	0	-1	3	-1	1