Algorithm Design Methods
Greedy Algorithm

by Chin Huang Lin

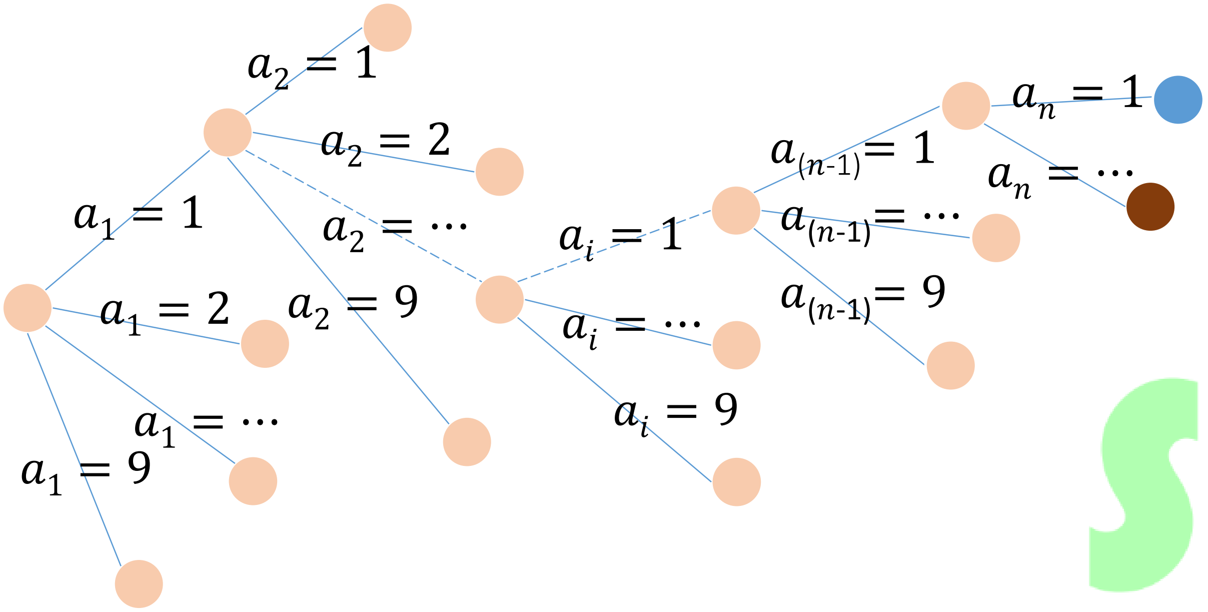
Sprout

什麼是「貪心」？

- 常見的貪心：
  1. 爸媽買了布丁，偷偷吃掉老哥的
  2. 國中生追女生，一個不夠追兩個
  3. 期末公佈成績，A 不夠還要 A+
  4. 廠商收入縮減，先降低員工薪資
  5. 參加朋友喜宴，自備塑膠袋打包
  6. ……
- Want more and better!

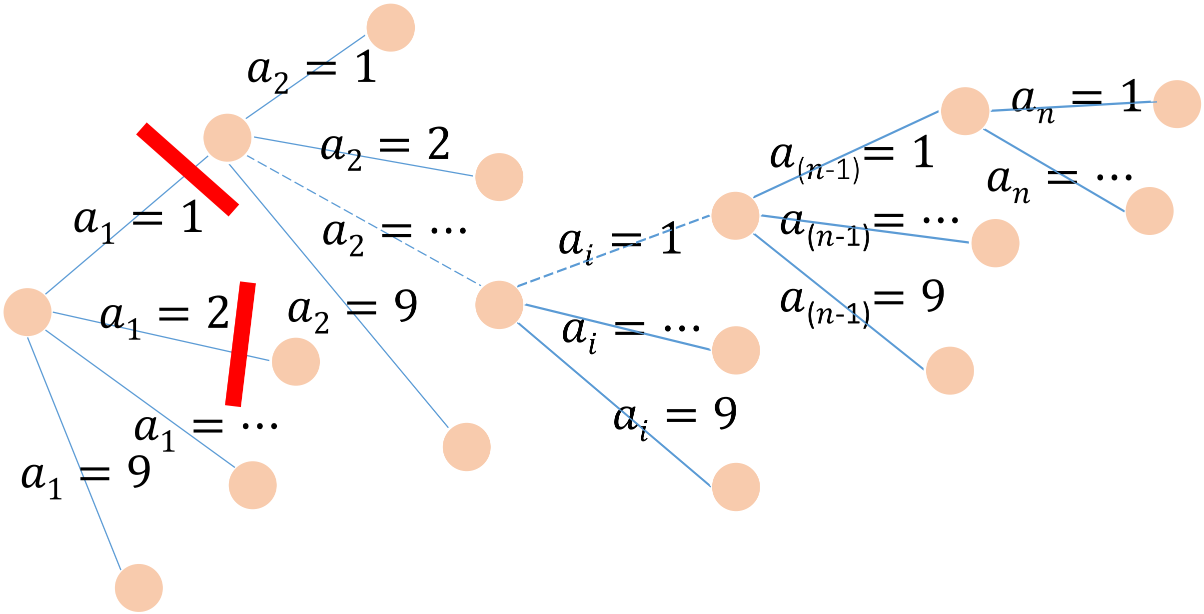
看不見的樹

- 對於一個問題，我們考慮的每一組方案事實上對應到一組「決策」
  - ex. 數獨問題中有 n 格空格，依序為編號為 $1\sim n$，那麼我們事實上是在決策 $a_1, a_2, \dots, a_n$，其中 a_i 為第 i 格的值，可以為 1~9 之間的數字。
- 整個決策的過程可以畫成一棵樹，每個葉節點對應到一個可能的方案，其中有些滿足所有條件形成解，其他則否
  - ex.



枚舉在幹麼？

- 枚舉事實上就是試圖拜訪整棵決策樹，找出樹裡的解節點
  - 剪枝事實上就是根據一些推理排除絕對不可能的分支



5	8	a_1
2	a_2	7
3	a_3	1

- 有時候，我們可以根據一些問題的特性，確定解節點所在的分支！
  - 總是選擇當前看起來最有利的分支，然後義無反顧

生活中的貪心（一）

- 黑猩國是一個由大量的黑腥猩建立的國度，裡面共有 n 種面額的貨幣。其中必定有一種面額是 1 元，剩餘的面額不一定，但對於任意兩種面額 a, b $(a \leqq b)$，必定有 $a \mid b$。那麼當黑腥猩要付 x 元購物的時候，最少需要用到幾個貨幣呢？
- $n \leqq 30$，$x \leqq 2147483647$

這麼簡單，我幼稚園就會了！

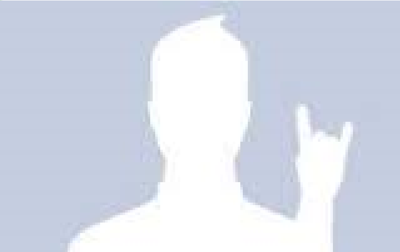
- 現行的貨幣在不考慮 20 元的情況下事實上就滿足條件
- 生活中你都怎麼付款？
  - 當然都盡量以大面額的付阿！
- 為什麼呢？

♪ 想一想：如果沒有「任意兩面額必定有一者是另一者的因數」這個條件，這樣做還會正確嘛？為什麼？

生活中的貪心（二）

- Zerojudge b105 誰先晚餐 (2005 NPSC 國中組決賽)
  - http://zerojudge.tw/ShowProblem?problemid=b105
- 生活中，你會怎麼做？
  - 當然是先讓吃最慢的人吃阿！
- 為什麼呢？

貪心法的簡單證明



老師，我這題這樣做有什麼問題？

嗯……你怎麼知道你這作法是對的？

可是我範測和自己產的測資都對阿

而且我也想不到更好的方法了

貪心法的簡單證明

- 一時找不到反例，不代表這個作法是對的
  - 輸入有這麼多種可能，有測到的佔不到 1%，這樣就說正確的話……
  - 不就跟只訪問 4 個人他們總統選舉會投誰，就斷定誰會當選一樣？
- 想不到更好的方法，不構成這個作法的證明
  - 怎麼知道不是運氣不好或者資質駑鈍，恰好沒有想到更好的方案？
- 貪心法需要更積極的證明方式！

貪心法的簡單證明

- 對於一個問題 P，提出一個作法 A
- 目標：證明 A 給出的解 S 總是最優的
- 先假設 A 給出的解 S 在某種情況下不是最優的
- 假設此時 P 的真正最優解應為 S’，那麼證明 S’ 不存在（利用矛盾)
  - 通常有兩個方法：
    1. 對於任意的 S'，都構造出一組相異於 S' 的解 S''，並且滿足 S'' 比 S' 還好
    2. 設法證明 S 不比 S' 還糟
- 於是 A 給出的解 S 總是最優的

貪心法的簡單證明

- Zerojudge b105 誰先晚餐 (2005 NPSC 國中組決賽)
- 符號定義：對於一組方案 S，用 $|S|$ 表示 S 方案下全部的人都吃完飯的時間
- 證明架構：
  1. 為了方便表示與比較，先用代數表達出問題與解答的數學模型
  2. 假設我們的算法給出的解 S 是錯的，那麼存在真正的最優解 S'，且 $|S'| < |S|$
  3. 如果有兩個人編號分別為 i, j (編號就代表吃飯的順序) 滿足「$i < j$ 且 i 吃得比 j 快」，我們就說這兩人形成一組「逆序對」。既然最優解 $S' \neq S$，那麼 S' 裡面一定存在相鄰的兩元素形成逆序對 (想想看，為什麼)
  4. 透過把這兩個人的順序交換，獲得方案 S_1'，並證明 $|S_1'| \leqq |S'|$
  5. 如果 S_1' 中還存在相鄰逆序，則再把該對逆序元素交換形成 S_2'，類似 4 的證明可知 $|S_2'| \leqq |S_1'|$
  6. 不斷重複類似 5 的步驟，直到當前的解 S_x' 中不存在相鄰逆序。此時 S_x' 必定不存在任何逆序對 (想想看，為什麼)，從而有 $S_x' = S$
  7. 於是有 $|S| \leqq |S_x'| \leqq |S_{x-1}'| \leqq \cdots \leqq |S_1'| \leqq |S'|$，即 $|S| \leqq |S'|$，與 $|S'| < |S|$ 的假設矛盾，從而證明 S 是最佳解

貪心法的簡單證明

1. 為了方便表示與比較，先用代數表達出問題與解答的數學模型
   - 一組方案中，設吃飯順序依序為 $\{1,2,\ldots,n\}$，則
   - 吃飯所需時間依序以 $\{e_1,e_2,\ldots,e_n\}$ 來代表
   - 料理所需時間依序以 $\{c_1,c_2,\ldots,c_n\}$ 來代表
   - 由於廚師一定不會停手，第 x 個人吃完飯的時間為 $f(x)=\sum_{j=1}^{x}c_j+e_x$
   - 題目要求 $\max\{f(x)|1\leqq x\leqq n\}$ 盡量小

Sprout

貪心法的簡單證明

2. 假設存在真正的最優解 S'，且 $|S'| < |S|$
3. 既然最優解 $S' \neq S$，那麼 S' 裡面一定存在相鄰的兩元素形成逆序對
   - 不妨假設就是第 i 個人和第 $i+1$ 個人形成相鄰逆序對

貪心法的簡單證明

4. 透過把這兩個人的順序交換，獲得方案 S_1'，並證明 $|S_1'| \leqq |S'|$
5. 如果 S_1' 中還存在相鄰逆序，則再把該對逆序元素交換形成 S_2'，類似 4 的證明可知 $|S_2'| \leqq |S_1'|$
6. 不斷重複類似 5 的步驟，直到當前的解 S_x' 中不存在相鄰逆序。此時 S_x' 必定不存在任何逆序對，從而有 $S_x' = S$
7. 於是有 $|S| \leqq |S_x'| \leqq |S_{x-1}'| \leqq \cdots \leqq |S_1'| \leqq |S'|$，即 $|S| \leqq |S'|$，與 $|S'| < |S|$ 的假設矛盾，從而證明 S 是最佳解

- 事實上要證明的只有兩件事情：
  1) 對於一個存在相鄰逆序對的方案 S_j'，可以透過交換該組逆序對獲得 S_{j+1}'，且 $|S_{j+1}'| \leqq |S_j'|$
  2) 不斷重複地把相鄰逆序對交換，總有一天會變成一組不存在相鄰逆序的方案

貪心法的簡單證明

1) 對於一個存在相鄰逆序對的方案 S'_j，可以透過交換該組逆序對獲得 S'_{j+1}，且 $|S'_{j+1}| \leqq |S'_j|$

- S'_j 的情形：
  - 吃飯順序依序為 $\{1,2,\dots,i,i+1,\dots,n\}$
  - 吃飯所需時間依序為 $\{e_1,e_2,\dots,e_i,e_{i+1},\dots,e_n\}$
  - 料理所需時間依序為 $\{c_1,c_2,\dots,c_i,c_{i+1},\dots,c_n\}$
  - 第 x 個人吃完飯的時間為 $f(x) = \sum_{j=1}^{x} c_j + e_x$
  - 第 i 個人和第 $i+1$ 個人形成相鄰逆序對
  - $|S'_j| =$ $\max(\{f(x) \mid x \neq i \,\&\&\, x \neq i+1\}, f(i), f(i+1))$

- S'_{j+1} 的情形：
  - 吃飯順序依序為 $\{1,2,\dots,i+1,i,\dots,n\}$
  - 吃飯所需時間依序為 $\{e_1,e_2,\dots,e_{i+1},e_i,\dots,e_n\}$
  - 料理所需時間依序為 $\{c_1,c_2,\dots,c_{i+1},c_i,\dots,c_n\}$
  - 第 x 個人吃完飯的時間為 $f'(x)$，等下討論
  - 第 i 個人和第 $i+1$ 個人不再是相鄰逆序對
  - $|S'_{j+1}| =$ $\max(\{f'(x) \mid x \neq i \,\&\&\, x \neq i+1\}, f'(i), f'(i+1))$

貪心法的簡單證明

1) 對於一個存在相鄰逆序對的方案 S_j'，可以透過交換該組逆序對獲得 S_{j+1}'，且 $|S_{j+1}'| \leqq |S_j'|$

- $f'(x)$：
  - 對於 $x < i$，$f'(x) = \sum_{j=1}^{x} c_j + e_x = f(x)$
  - 對於 $x > i + 1$，$f'(x) = \sum_{j=1}^{i-1} c_j + c_{i+1} + c_i + \sum_{j=i+1}^{x} c_j + e_x = \sum_{j=1}^{x} c_j + e_x = f(x)$
  - $f'(i) = \sum_{j=1}^{i-1} c_j + c_{i+1} + e_{i+1}$
  - $f'(i+1) = \sum_{j=1}^{i-1} c_j + c_{i+1} + c_i + e_i$
  - 發現

    $\max(\{f'(x) \mid x \neq i \,\&\&\, x \neq i+1\}, f'(i), f'(i+1))$

    $= \max(\{f(x) \mid, x \neq i \,\&\&\, x \neq i+1\}, f'(i), f'(i+1))$

- S_{j+1}' 的情形：
  - 吃飯順序依序為 $\{1,2,\dots,i+1,i,\dots,n\}$
  - 吃飯所需時間依序為 $\{e_1,e_2,\dots,e_{i+1},e_i,\dots,e_n\}$
  - 料理所需時間依序為 $\{c_1,c_2,\dots,c_{i+1},c_i,\dots,c_n\}$
  - 第 x 個人吃完飯的時間為 $f'(x)$，等下討論
  - 第 i 個人和第 $i+1$ 個人不再是相鄰逆序對
  - $|S_{j+1}'| =$

    $\max(\{f'(x) \mid x \neq i \,\&\&\, x \neq i+1\}, f'(i), f'(i+1))$

貪心法的簡單證明

1) 對於一個存在相鄰逆序對的方案 S_j'，可以透過交換該組逆序對獲得 S_{j+1}'，且 $|S_{j+1}'| \leqq |S_j'|$

- $f'(x)$：
  - 對於 $x < i$，$f'(x) = \sum_{j=1}^{x} c_j + e_x = f(x)$
  - 對於 $x > i + 1$，$f'(x) = \sum_{j=1}^{i-1} c_j + c_{i+1} + c_i + \sum_{j=i+1}^{x} c_j + ex = \sum_{j=1}^{x} c_j + ex = f(x)$
  - $f'(i) = \sum_{j=1}^{i-1} c_j + c_{i+1} + e_{i+1}$
  - $f'(i+1) = \sum_{j=1}^{i-1} c_j + c_{i+1} + c_i + e_i$
  - 發現

    $\max(\{f'(x) \mid x \neq i \;\&\&\; x \neq i+1\}, f'(i), f'(i+1))$

    $= \max(\{f(x) \mid, x \neq i \;\&\&\; x \neq i+1\}, f'(i), f'(i+1))$

- 比較 $|S_j'|$ 與 $|S_{j+1}'|$ ：
  - 為了簡潔，用 α 代表 $\{f'(x) \mid x \neq i \;\&\&\; x \neq i+1\}$
  - $|S_j'| = \max(\alpha, f(i), f(i+1))$
  - $|S_{j+1}'| = \max(\alpha, f'(i), f'(i+1))$
  - 問題：$\max(f(i), f(i+1)), \max(f'(i), f'(i+1))$，誰大？

貪心法的簡單證明

1) 對於一個存在相鄰逆序對的方案 S_j'，可以透過交換該組逆序對獲得 S_{j+1}'，且 $|S_{j+1}'| \leqq |S_j'|$

- $f(i), f(i+1), f'(i), f'(i+1)$：
  - 已知條件：$e_i < e_{i+1} \Rightarrow e_{i+1} - e_i > 0$
  - 為了簡潔，用 k 代表 $\sum_{j=1}^{i-1} c_j$
  - $f(i) = k + c_i + e_i$
  - $f(i+1) = k + c_i + c_{i+1} + e_{i+1}$
  - $f'(i) = k + c_{i+1} + e_{i+1}$
  - $f'(i+1) = k + c_{i+1} + c_i + e_i$
  - $f(i+1) - f(i) = c_{i+1} + e_{i+1} - e_i > 0$
  - $f(i+1) - f'(i) = c_i > 0$
  - $f(i+1) - f'(i+1) = e_{i+1} - e_i > 0$
  - ♪ $f(i+1)$ 比其他三者都還大，從而有 $\max\big(f(i), f(i+1)\big) > \max(f'(i), f'(i+1))$

- 比較 $|S_j'|$ 與 $|S_{j+1}'|$ ：
  - 為了簡潔，用 α 代表 $\left\{f'(x) \,\middle|\, x \neq i \;\&\&\; x \neq i+1\right\}$
  - $\left|S_j'\right| = \max\big(\alpha, f(i), f(i+1)\big)$
  - $\left|S_{j+1}'\right| = \max\big(\alpha, f'(i), f'(i+1)\big)$
  - 問題：$\max(f(i), f(i+1)), \max(f'(i), f'(i+1))$，誰大？

Sprout

貪心法的簡單證明

2) 不斷重複地把相鄰逆序對交換，總有一天會變成一組不存在相鄰逆序的方案

- 根據逆序對的定義可知，每次交換後，逆序對總數會 -1
- 由「逆序對數不為 0 則存在相鄰逆序對」可知，永遠找得到相鄰逆序對交換
- 逆序對數為 0 時，顯然相鄰逆序對數也為 0，得証

不直接的貪心

- UVa 714 Copying Books
  - http://uva.onlinejudge.org/external/7/714.html
- 為了懲罰你和你的朋友們 (共 m 個人) 一起把垃圾偷偷丟到隔壁班的花盆裡，老師精選了 n 篇的文章給你們在午休時站著罰抄。
- 這 n 篇文章在排成一列的 m 個人面前排成一列，為了作業方便你們決定每個人都只能負責抄寫其中連續排列的若干篇文章 (例如某個人可以負責抄寫第 2~5 篇文章，卻不能夠負責抄寫第 2,3,5,8 篇文章)；而且為了工整性，老師要求一篇文章一定只能由一個人抄寫，否則會有不同的字跡。
- 由於你們的感情很好，所以就算每個人負責的份量不同也不會吵架。但是為了早點抄完回去打鬧，你們希望負責份量最多的人負責的份量盡量少。
- 在你們已經知道這 n 篇文章的頁數依序為 $\{p_1, p_2, \ldots, p_n\}$，請問負責份量最多的那位同學最少要負責幾頁的抄寫量呢？

一點瓶頸

- 如果後面人還很多，就應該抄少一點；如果後面人少，就應該抄多一點
  - 但是怎麼拿捏呢？
- 如果自己不是第一名，那應該盡量抄多一點；如果自己是第一名，那應該盡量抄少一點；但是第一名又必須要是第一名（？）
- 如果我們知道到底第一名最後抄多少書就好了……
- 最優化決策很難 (要把決策樹上所有葉節點拿出來比較)，但是判定問題簡單很多—怎麼把最優化決策問題轉為判定問題呢？
  - 當然就是枚舉啦！
- 直接枚舉第一名要抄多少頁 (其實就是枚舉每個人最多不能夠超過多少頁)，然後貪心地盡量抄到不能再抄
  - 如果抄不完，代表當前枚舉得太理想
  - 複雜度 $O(m\sum_{i=1}^{n} p_i)$，好慢！

Sprout

有點單調

- 我們要求的是「抄得完書的情況下，負擔最重的人要抄的最少頁數 p」
- 如果我們枚舉負擔最重的人要抄 x 頁，那麼對於所有 $x < p$，一定都抄不完；對於所有 $x \geqq p$，一定都抄得完
- 不就是摔蛋問題嘛？
  - 馬上把枚舉的部份改成二分枚舉，複雜度降為 $O(m \log \sum_{i=1}^{n} p_i)$

霍夫曼編碼

- 一般而言，電腦中的每個字元都以相同的位元數來儲存，舉例而言只要是 char 都用 8 bits 來儲存，因此有 n 個字元，就會佔用 $8n$ bits
- 如果已經知道哪些字元各會出現幾次，事實上我們有更好的辦法！舉例來說：
  - 基因序列裡只會有四種字元：ATCG
  - 已知 A 出現 185 次，T 出現 47 次，C 出現 59 次，G 出現 308 次
  - 兩種不同的表達法：

A	T	C	G
00	01	10	11

A	T	C	G
10	101	100	0

  - 左邊的表達法總共需要 1198 bits，右邊的表達法只需要 996 bits

非固定長度編碼

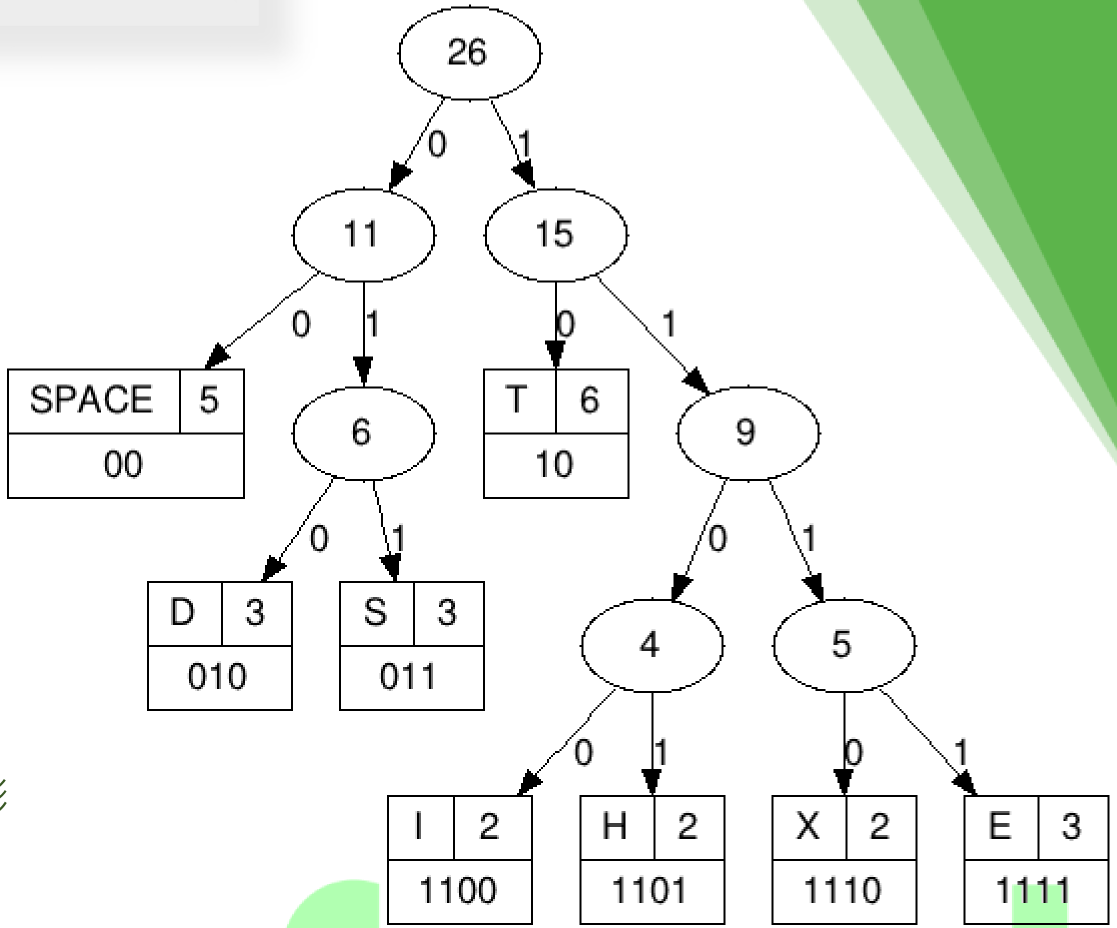
- 問：為什麼不要用更短的編碼？
  - ex.

A	T	C	G
1	01	00	0

- 答：這樣子在解釋 000 時會有歧異，可解釋為 CG 或 GC
- 所以任意字元的編碼不能是另一字元編碼的前綴 (prefix)
  - 這代表所有字元的編碼 (即一個編碼方案) 可以形成一棵 tree！
- 其中最優 (最省空間) 的編碼方案對應到的 tree，稱為最優編碼樹

最優編碼樹

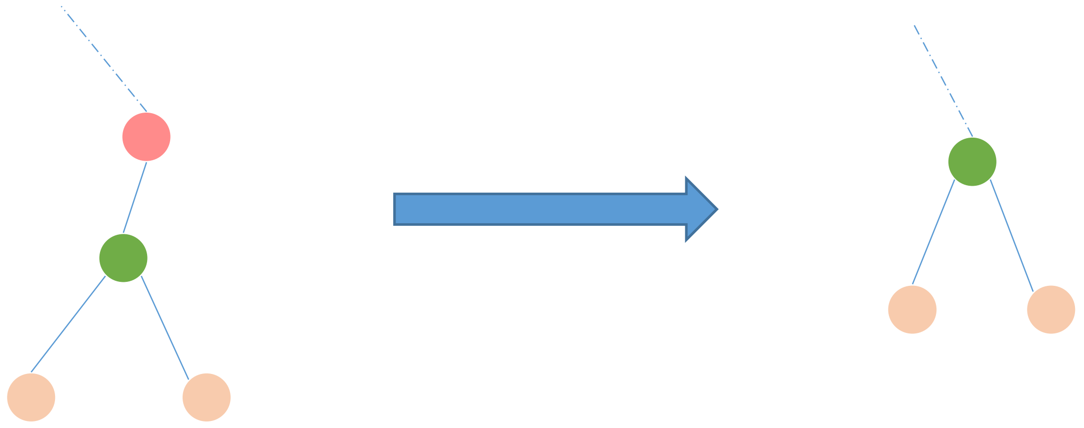
- 右邊是一棵當文本是「THIS IS THE TEST TEXT XDDD」時的最優編碼樹
- 最優編碼樹的一些性質
  1. 一定不會有只有一個兒子的節點
  2. 頻率越高的字元對應的葉節點深度越淺
  3. 必定存在一棵最優編碼樹，使得頻率最低的兩個字元對應的葉節點形成兄弟
  4. 定義 $fs(x)$ 為節點 x 形成的子樹中，所有葉節點的頻率和。則某編碼樹為 T_0，把 T_0 中某節點 x 形成的子樹整棵替換為一個頻率為 $fs(x)$ 的字元形成新樹 T，那麼 T 為新字元集的最優編碼樹當起僅當 T_0 為最優編碼樹。



圖片來源: http://huffman.ooz.ie/

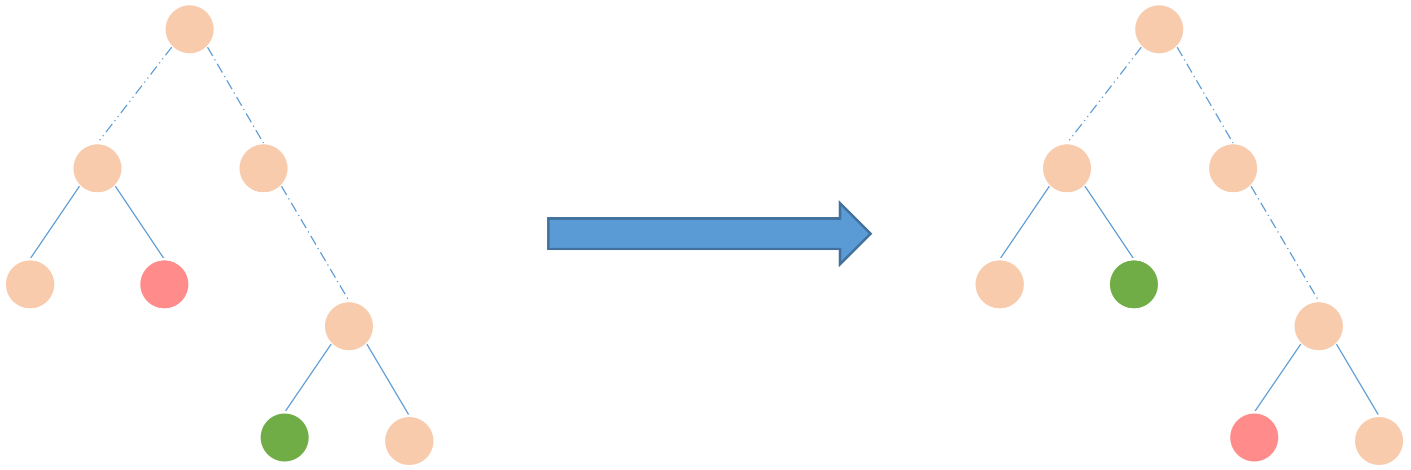
最優編碼樹

1. 一定不會有只有一個兒子的節點
   - 否則我們直接用這個節點的子節點取代它自己，依舊是合法的編碼樹但空間消耗更小



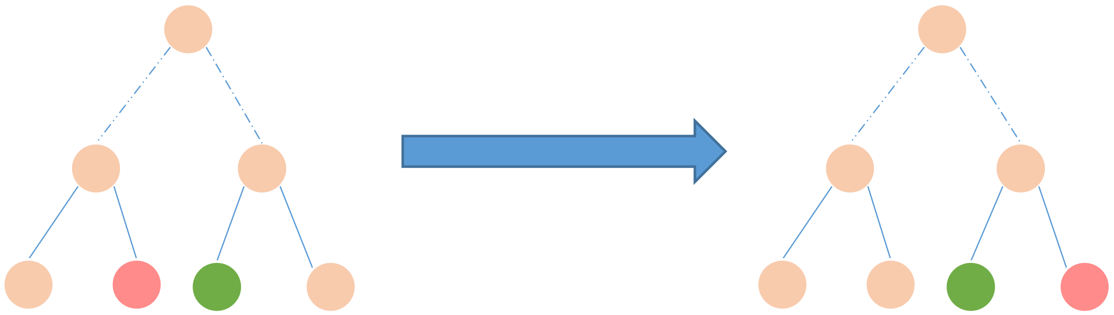
最優編碼樹

2. 頻率越高的字元對應的葉節點深度越淺
   - 否則我們交換兩者的位置，空間消耗更小



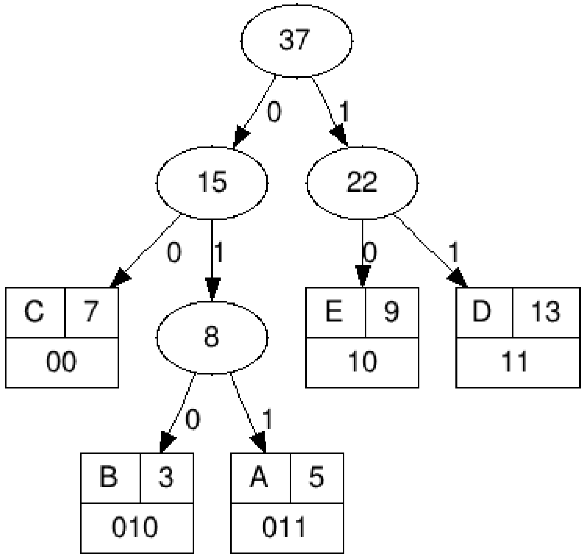
最優編碼樹

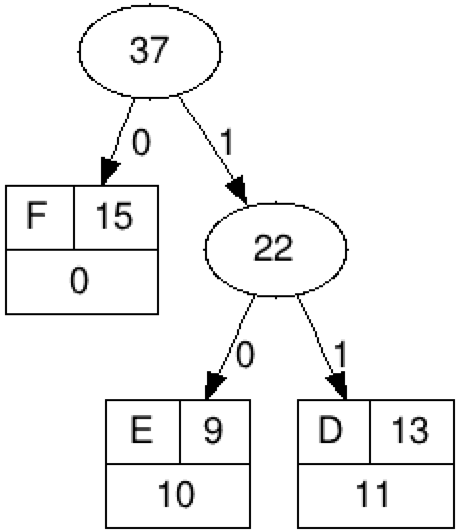
3. 必定存在一棵最優編碼樹，使得頻率最低的兩個字元對應的葉節點形成兄弟
   - 由性質 1 可知深度最大的一層至少有兩個葉節點
   - 再配合性質 2 可確定頻率最低的兩個字元皆位於深度最大的一層
   - 如果兩者並非兄弟，則把他們位置換成兄弟，解的優劣不變



最優編碼樹

4. 某編碼樹為 T_0，把 T_0 中某節點 x 形成的子樹整棵替換為一個頻率為 $fs(x)$ 的字元形成新樹 T，那麼 T 為新字元集的最優編碼樹當起僅當 T_0 為最優編碼樹





圖片來源: http://huffman.ooz.ie/

最優編碼樹

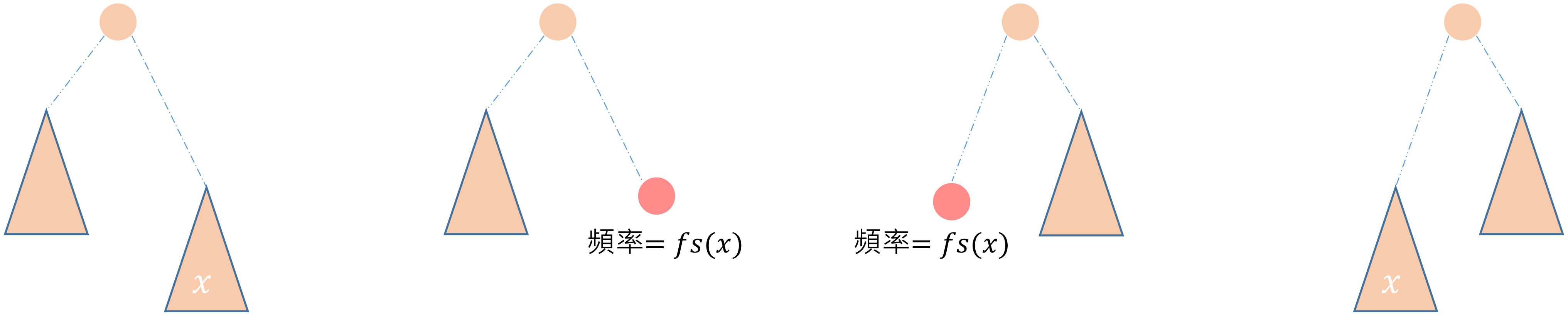
4. 某編碼樹為 T_0，把 T_0 中某節點 x 形成的子樹整棵替換為一個頻率為 $fs(x)$ 的字元形成新樹 T，那麼 T 為新字元集的最優編碼樹當起僅當 T_0 為最優編碼樹

- 每個葉節點貢獻的空間消耗量為「字元頻率×節點深度」
- 設節點 x 形成的子樹中有 n 個葉節點，頻率分別為 $\{f_1, f_2, \ldots, f_n\}$，與節點 x 的深度差分別為 $\{l_1, l_2, \ldots, l_n\}$
- 當節點 x 的深度為 d 時，整棵子樹的空間消耗量為

$$\sum_{i=1}^{n} f_i(l_i + d) = \sum_{i=1}^{n} f_i l_i + d \sum_{i=1}^{n} f_i = \sum_{i=1}^{n} f_i l_i + d\, fs(x)$$

- 假如把 x 形成的子樹整棵替換為一個頻率為 $fs(x)$ 的字元形成的葉節點，新得到的樹卻不是最優編碼樹的話……

最優編碼樹



原先的最優編碼樹 T_0
總空間消耗$=$ x 子樹消耗
+其他部份消耗 1

T_0 變換節點後的編碼樹 T
總空間消耗$=$ x 子樹消耗
+其他部份消耗 1
$-\sum_{i=1}^{n} f_i l_i$

假設中比 T 更好的最優編碼樹 T'
總空間消耗$=$ x 子樹消耗
+其他部份消耗 2
$-\sum_{i=1}^{n} f_i l_i$

把 T' 中的紅點再次換回 x 子樹的 T_0'
總空間消耗$=$ x 子樹消耗
+其他部份消耗 2

- 根據假設，必須有 其他部份消耗1 ＞ 其他部份消耗2
- 從而 T_0' 優於 T_0，與假設矛盾！

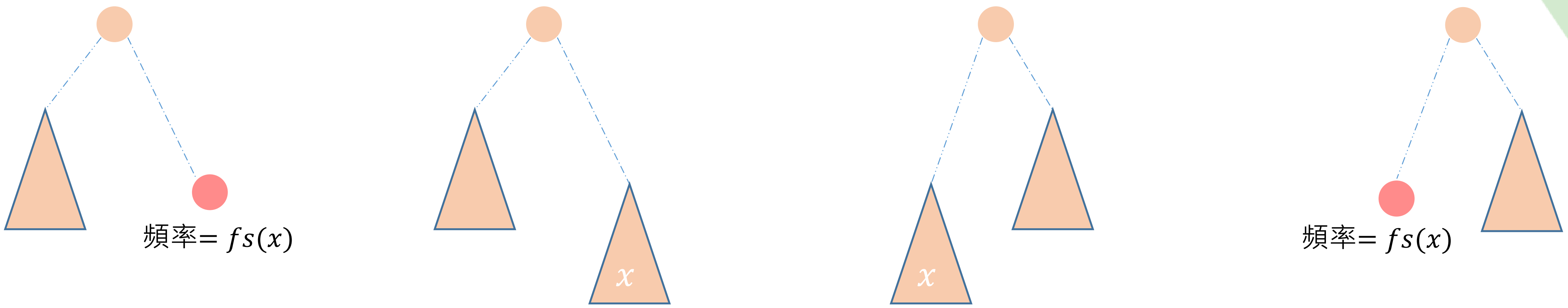
證明時間

4. 某編碼樹為 T_0，把 T_0 中某節點 x 形成的子樹整棵替換為一個頻率為 $fs(x)$ 的字元形成新樹 T，那麼 T 為新字元集的最優編碼樹當起僅當 T_0 為最優編碼樹
   - 每個葉節點貢獻的空間消耗量為「字元頻率×節點深度」
   - 設節點 x 形成的子樹中有 n 個葉節點，頻率分別為 $\{f_1, f_2, \dots, f_n\}$，與節點 i 的深度差分別為 $\{l_1, l_2, \dots, l_n\}$
   - 當節點 x 的深度為 d 時，整棵子樹的空間消耗量為

$$\sum_{i=1}^{n} f_i(l_i + d) = \sum_{i=1}^{n} f_i l_i + d\sum_{i=1}^{n} f_i = \sum_{i=1}^{n} f_i l_i + d\, fs(x)$$

   - 假如把 x 形成的子樹整棵替換為一個頻率為 $fs(x)$ 的字元形成的葉節點，新得到的樹卻不是最優編碼樹的話……
   - 類似地如果 T 是最優編碼樹，但 T_0 不是最優編碼樹的話……

最優編碼樹



T_0 變換節點後的最優編碼樹 T
總空間消耗$=$ x 子樹消耗
$+$其他部份消耗 1
-$\sum_{i=1}^{n} f_i l_i$

原先的編碼樹 T_0
總空間消耗$=$ x 子樹消耗
$+$其他部份消耗 1

假設中比 T_0 更好的最優編碼樹 T_0'
總空間消耗$=$ x 子樹消耗
$+$其他部份消耗 2

把 T_0' 中紅點換回 x 子樹的 T'
總空間消耗$=$ x 子樹消耗
$+$其他部份消耗 2
-$\sum_{i=1}^{n} f_i l_i$

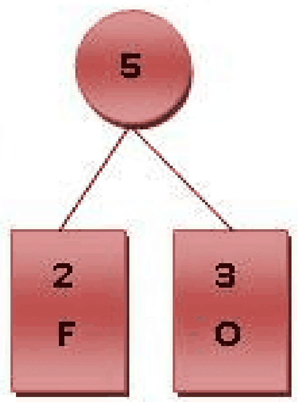
- 根據假設，必須有 其他部份消耗1 $>$ 其他部份消耗2
- 從而 T' 優於 T，與假設矛盾！

霍夫曼編碼

1. 把所有字元都視為一棵單節點樹，初始權重為字元頻率，全部的字元形成一個森林
2. 每次從森林裡面取出權重最小的兩棵樹，並把它們合併成一棵樹，權重為兩棵樹的權重和
3. 把合併後獲得的樹再加回去森林中，重複 2 直到森林中剩下一棵樹
4. 此時該樹即為最優編碼樹

霍夫曼編碼

F： 2
O： 3
E： 5
R： 4
G： 4
T： 7



圖片來源: http://zh.wikipedia.org/wiki/File:Huffman_algorithm.gif

野生的新證明手法

- 我們目前已經學過：
  - 數學歸納法
  - 直接反證法
- 遞迴證明法！

1) 定義 P_n 為輸入規模為 n 時的子問題
2) 證明當 $n = 1$ 時，命題成立
3) 證明已知包含 P_{n-1} 的解的情況下，貪心策略正確
4) 證明存在 P_n 的解包含 P_{n-1} 的解
5) 命題得證

野生的新證明手法

1) 定義 P_n 為輸入規模為 n 時的子問題
   - 定義 P_n 為共有 n 種字元時的最佳解
2) 證明當 $n = 1$ 時，命題成立
   - 不能更明顯了 XD
3) 證明已知包含 P_{n-1} 的解的情況下，貪心策略正確
   - 由性質 3，至少存在一組解包含此貪心策略
4) 證明存在 P_n 的解包含 P_{n-1} 的解
   - 根據性質 4，找一個恰包含兩葉節點的子樹（根據性質 1，只要 $n \geqq 2$ 這樣的子樹必定存在）進行縮點，此時 P_n 的解必定包含 P_{n-1} 的解
5) 命題得證