前言：

第三周的課程介紹霍夫曼編碼 (Huffman Code)，作為貪婪演算法的最後一個部份。接著進入新的主題：動態規劃，以經典的最大權重獨立集 (Max Weight Independent Set)，講解動態規劃的精神 ：

1. 將問題拆分成多個子問題。
2. 解出較小的子問題後，可利用這些資訊，解出更大的子問題。
3. 如此重複遞迴，就可以解出最終答案。

作業為計算給定資料的 Huffman Code，並回傳最大及最小編碼長度。本文將著重在 Huffman Code，動態規劃的相關算法，請參考下週以及專項課程四的內容。

Huffman Code：

Huffman Code 為一種無失真壓縮的編碼技術，利用各符號出現的機率，來決定編碼長度。出現機率越高的符號，使用越短的編碼；機率越低的符合，則使用越長的編碼，如此一來便可降低編碼後字串的平均長度，達到資料壓縮的目的。

假設有一組符號 {A, B, C, D}要進行編碼，首先我們很直覺地會認為，機率最低的兩個編為 01、10，最高的兩高編為 0、1 就好啦，例如 {0, 01, 10, 1}。解碼後，其中一個位元碼 為 001，若依照編碼，可以是 AB，或著 AAD，我們無法明確知道何者是正確的！

Prefix-Free Codes

為了避免上述情況，有必要引入一個編碼方式：Prefix-Free Codes (無前綴碼)。例如：{0, 10, 110, 111}，這樣的一組編碼，就是無前綴的。除了 0，沒有其他以 0 開頭的編碼；除了 10 ，沒有其他以 10 開頭的編碼，如此就可以避免之前解碼時，有多種可能性的困擾。這種方法屬於變動長度的編碼方式。

平均字元所佔空間

評斷一個編碼的壓縮程度，我們可以比較其與固定長度編碼的差異。

範例：

已知 P(A) = 0.6，P(B) = 0.25，P(C) = 0.1，P(D) = 0.05，現有兩種編碼方法：

法 a. {00, 01, 10, 11} 以及

法 b. {0, 10, 110, 111}

試求兩方法的平均字串長度？

利用期望值的概念，我們知道字串長度的期望值分別為 E[法 a]以及 E[法 b]，又期望值為隨機變數發生某個值的機率，乘上隨機變數的值：

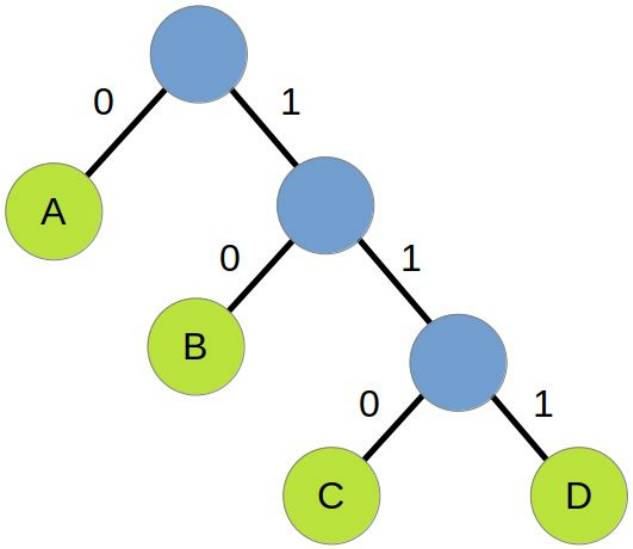
E[法 a] = (2*0.6+2*0.25+2*0.1+2*0.05) = 2；

E[法 b] = (1*0.06+2*0.25+3*0.1+3*0.05) = 1.55。

所以即便可能產生較長的編碼，但實際上每個字元平均所佔的空間，比固定長度編碼要小。

二元樹與 Huffman Code:

實現 Huffman Code 的主要方式為二元樹，左子節點的邊為 0 ，右子節點的邊為 1。由於必須為 Prefix-Free，因此所有的符號都位於葉節點 (leaf node)，而內部的其他節點僅為中繼的通過點。再以 {A, B, C, D}對應{0, 10, 110, 111}為例，二元樹的結構如下。

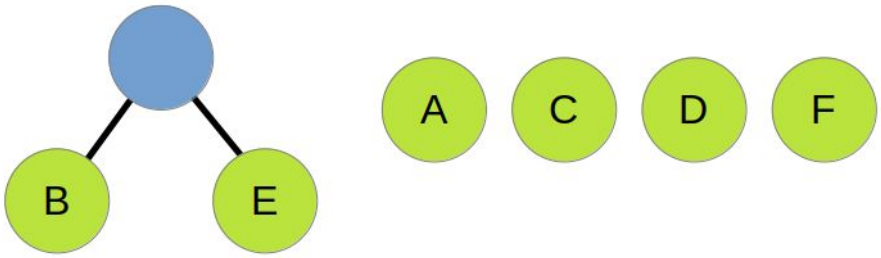


解碼的方法非常簡單，利用 DFS 遍歷所有節點即可得到各符號對應的編碼。

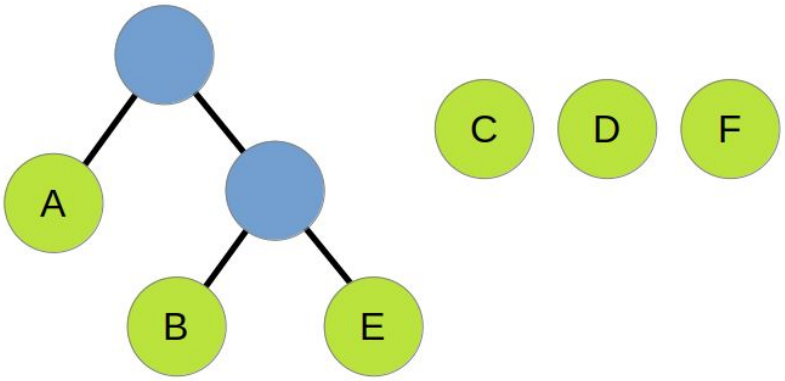
如何建立 Huffman Codes Tree：

Huffman Codes Tree 的建立方法為 Bottom-Up，也就是由下往上一步步合併，每次合併時，左節點放權重較小的子樹，右節點則放較大的子樹。假設我們有一組符號 {A, B, C, D, E, F}，權重分別為 {3, 2, 6, 8, 2, 6}：

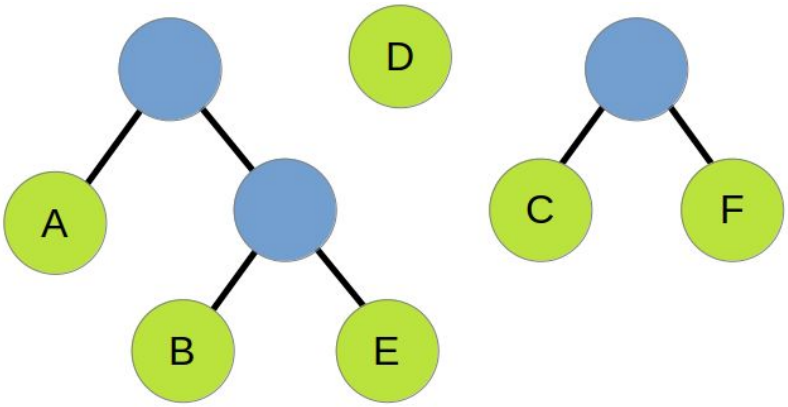
step 1. 合併 BE，對應關係變成 {A, BE, C, D, F} ⇒ {3, 4, 6, 8, 6}



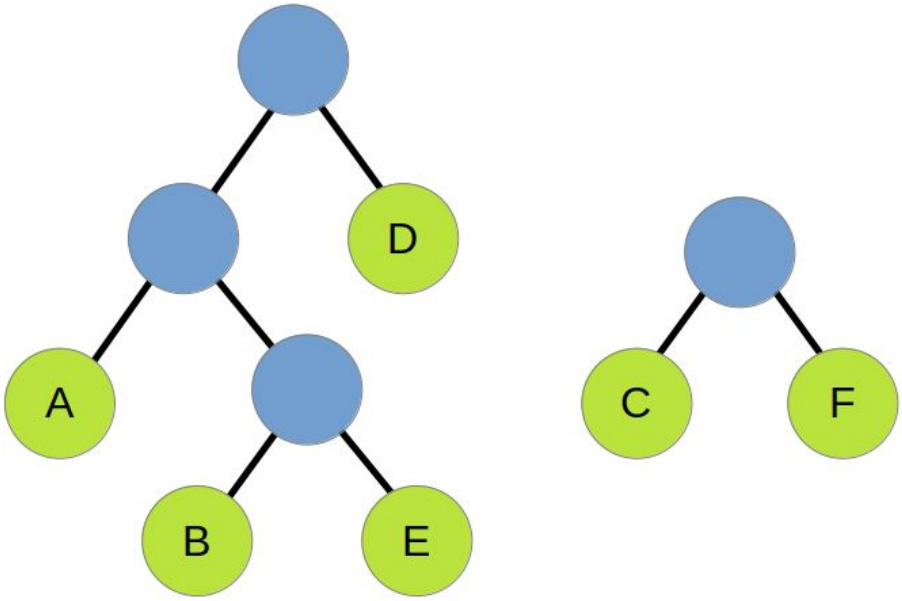
step 2. 合併 A 及 BE，對應關係變成 {ABE, C, D, F} ⇒ {7, 6, 8, 6}



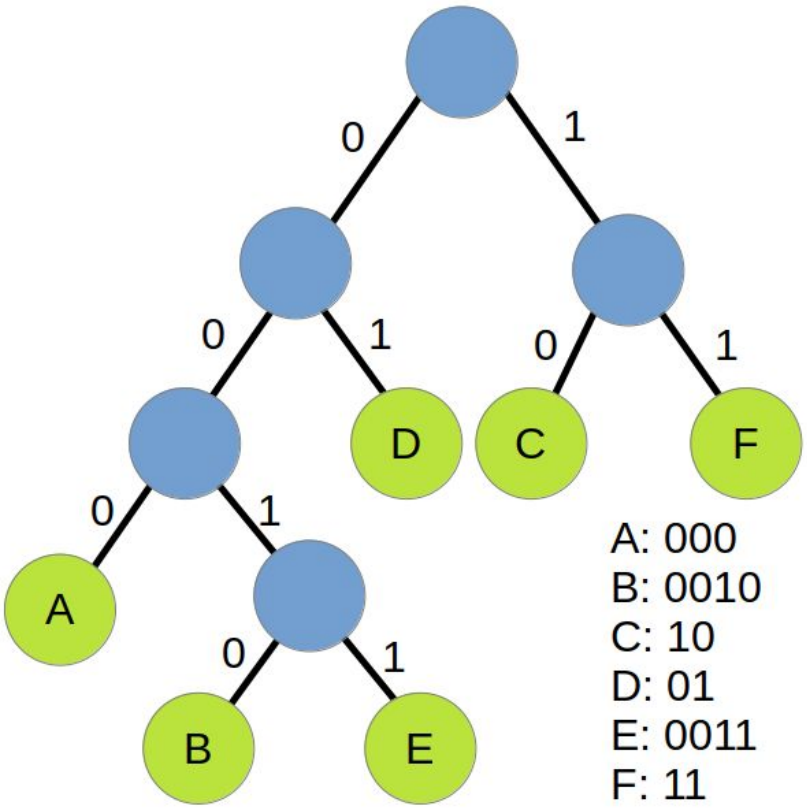
step 3. 合併 CF，對應關係變成 {ABE, CF, D} ⇒ {7, 12, 8}



step 4. 合併 ABE 跟 D，對應關係變成 {ABED, CF} ⇒ {15, 12}



step 5. 最後合併兩顆子樹，完成 Huffman Codes Tree



時間複雜度：

Huffman cods 共需合併 (n-1) 次，每次合併要遍歷所有子樹，找出最小的兩顆，因此時間複雜度為 $O(n^2)$ 。若使用 Heap 資料結構，則找出最小兩顆子樹的複雜度可降為 O(logn)，總時間複雜度為 O(nlogn)。

作業──計算 Huffman Codes 的最長及最短編碼：

問題描述：

給定一組資料的權重，請計算最長及最短的 Huffman Codes 長度。

解題方法：

依課程敘述之算法即可求解。