2010年臺灣國際科學展覽會

優勝作品專輯

編 號：110001

作品名稱

模糊理論的基因分類演算法

得獎獎項

電腦科學科大會獎三等獎

英特爾電腦科學獎一等獎

學校名稱：臺中縣立新社高級中學

作者姓名：楊鎮榮

指導老師：胡裕仁

關鍵詞：模糊聚類、鹼基序列、胺基酸序列

作者簡介

照片說明：圖中：楊鎮榮、圖左：胡裕仁老師、圖右：房其頌主任

我叫楊鎮榮，目前就讀台中縣立新社高中一年級。新社是一個美麗的山城，素有台中後花園美稱。在民風淳樸的環境中成長，養成我勤僕誠懇的個性。我的興趣是打籃球。因為自幼身體不好，故深深體悟到沒有強健體魄是沒有逐夢的可能，所以經由籃球運動，讓我的身體健康改善許多。

猶記得開學時，胡裕仁老師告訴我們要勇於逐夢，雖然學校在比較偏僻的山城之中，但並不表示我們的能力比較差，但是資訊可能比較少，所以有很多可以表現自我、拓寬眼界的活動可能無法參與，今日在老師辛勤指導及主任用心的為我提供資源，使我有幸參與 2010 年的台灣區國際科展活動，在此要向母校的所有老師說聲謝謝!

摘要

本文藉由模糊數學理論所提的分類方式，來設計一套對於 DNA 序列的分類方法，並利用了 40 筆人工已分完的樣本，分別作為測試及學習樣本。

發現可以順利的將每筆 DNA 序列中的鹼基每 3 個一組轉成胺基酸序列，再利用模糊分類方式將所有胺基酸序列進行分類，最後並利用原先滾動方式的環狀排列方式，來對同一筆鹼基的 3 組胺基酸資料進行檢核，發現可以有效提高分類的正確性。

之後我們再利用 182 筆自然樣本進行檢檢，也發現模糊分類方式亦可正確完成此次分類結果。

Abstract

This study is mainly about a DNA sequences classification method of a fuzzy mathematical theory. The classified way this article offers is by fuzzy mathematics theory to design a classification for the sequence of DNA. We use 40 samples, which have been divided, to be the test samples.The discovery may smoothly transfer each base in DNA sequence into 3 groups of amino acid sequence of protein.

Use fuzzy mathematics theory of classification to categorize all of amino acid sequence. At last, make use of the three original groups of sequence datum of amino acid to roll and form the same item of base and check. This discovery can raise the validity of the category. Afterwards, we can use 182 items of natural samples to check and find out fuzzy clustering can also finish the result correctly.

壹、研究動機(Introduction)

在學校的專題研究課中，老師介紹我們有關數學建模競賽，有很多開發而沒有標準答案的題目可以思考，也可以當作我們做專題的題目來源。因為在數學建模競賽中往往都給一個開放式的題目，但沒有提任何解決方式；因此可以給我們很多想像及發揮的空間，於是我和同學找到一個有趣的問題『DNA 序列分類問題的數學模型』[1]。

問題說明：在人類基因組計畫中 DNA 全序列草圖分析，預計在 2011 年可以完成精確的全序列圖，此後人類將擁有一本記錄著自身生老病死及遺傳進化的全部訊息。這訊息主要是由 4 個字符 A，T，C，G 按一定順序排成的長約 30 億組的序列，其中沒有“斷句”也沒有標點符號，除了這 4 個字符表示 4 種鹼基以外，人們對它包含的“內容”知之甚少，難以讀懂。破譯這部世界上最巨量訊息的將是二十一世紀最重要的任務之一。在這個目標中，研究 DNA 全序列具有什麼架構，是由這 4 個字符排成的，且外表看似隨機排列，當中隱藏著什麼規律，將是解讀它的基礎，此點正是近年來流行的生物資訊學（Bioinformatics）中最重要的課題之一。

雖然人類對這些問題了解不多，但也發現了 DNA 序列中的一些規律性和架構。例如，在全序列中有一些是用於編碼蛋白質的序列片段，即由這 4 個字符組成的 64 種不同的 3 字元串，其中大多數用於編碼構成蛋白質的 20 種胺基酸。又例如，在不用於編碼蛋白質的序列片段中，A 和 T 的含量特別多些，於是以某些鹼基特別豐富作為特徵去研究 DNA 序列的架構也取得了一些結果。此外，利用統計的方法還發現序列的某些片段之間具有相關性，等等。這些發現讓人們相信，DNA 序列中存在著局部的和全局性的架構，充分發掘序列的架構對理解 DNA 全序列是十分有意義的。目前在這項研究中最普通的思想是省略序列的某些細節，突出特徵，然後將其表示成適當的數學對象。這種被稱為模組化的方法往往有助於研究規律性和架構，並作為研究 DNA 序列的架構嘗試，因此在此提出對序列集

合進行分類的問題：

(1)假設有 40 個已知類別的人工製造的序列，其中序列標號 1－10 及 21-30 為 A 類，11－20 及 31-40 為 B 類。請從中依其特徵，選擇分類方法，並用這些已知類別的序列，衡量你的方法是否夠好。

(2)另外以同樣的方式去分未知類別的自然界中的序列，共給定 182 個自然 DNA 序列，它們都較長。用你的分類方法對它們進行分類，像（1）一樣地給出分類結果。[1]

之所以會對 DNA 序列感到興趣主要是家人在農業改良所上班，經常在無意中提到一些基因分類及物種識別的問題，例如：由 DNA 序列探討 XXX 等基因分類問題。因此希望從事 DNA 序列分類研究，來幫媽媽解決一些基因的分類問題。

貳、研究目的(Expected Results)

由於這是開放性比賽活動，所以當年也有不少人提出解決方式及數學模型，經查證在當年度獲得一等獎的作品，是利用了系統聚類方式（K-means）設計出一組分類演算法[1][7][8]。因此在老師的建議下我們開始尋求其它的聚類方式來進行分類，最後決定一項沒人用過的方式；即模糊聚類方法 Fuzzy C-Means(FCM)[3][7][9]。

並由我負責程式設計及模擬分類，另一位同學負責將方法整合，且在老師的分工督導之下進行相關模擬分析，並且希望獲得下列結果：

（1）重新設計一組結合具有分類 DNA 序列功能的模糊分類演算法。

（2）驗證結合模糊理論的分類方法的有效性。

參、 研究過程(Materials and Methods)

一、 理論基礎

(一) DNA 去氧核糖核酸

DNA 脱氧核糖核酸（英文：Deoxyribonucleic acid，縮寫為 DNA）又稱去氧核糖核酸，是一種分子，可組成遺傳指令，以引導生物發育與生命機能運作，通常 DNA 序列具有高相似性，代表兩序列源自相同的祖先，並具有相同的空間結構，生物學家稱為同質性，而同質性序列通常具有類似的生化功能。而生物學上定義，若蛋白質中超過 25%的胺基酸序列相同，或 DNA 中超過 75%的含氮鹼基序列相同，便幾乎可以確定蛋白質及 DNA 序列具有同質性，此點亦可作為親緣判定的參考[2]。

DNA 序列通常使用一串字母表示真實的，或者假設攜帶基因訊息的 DNA 分子的一級結構。可能的字母只有 A、C、G 和 T，分別代表組成 DNA 的四種核苷酸－腺嘌呤（A）、胞嘧啶（C）、鳥嘌呤（G）、胸腺嘧啶（T）。典型的 DNA 序列無間隔的排列在一起，例如序列 AAAGTCTGAC。而任意長度大於 4 的一串核苷酸被稱作一個序列。關於它的生物功能，則依賴於上下文的序列，一個序列可能被正讀，反讀；包含編碼 或者無編碼。DNA 序列也可能包含"junk DNA"[2][4][11]。

(二) RNA 核糖核酸

核糖核酸（英文：Ribonucleic acid，縮寫：RNA）是存在於細胞生物的遺傳訊息中間載體，RNA 的鹼基主要有四種，即腺嘌呤（A）、鳥嘌呤（G）、胞嘧啶（C）和尿嘧啶（U），並參與蛋白質合成遺傳訊息的傳導流向為 DNA 到蛋白質，科學家 Crick 提出一個假說：認為 RNA 才是最早出現的遺傳訊息密碼，因為：1.RNA 具有自我複製的功能。2.RNA 同時具有酵素的功能。3.RNA 為單股結構，不似 DNA

複雜。雖然細胞核中有很多雙股的 DNA，但在細胞質中除了有很多的蛋白質之外，還發現有很多單股的 RNA 存在。胚胎細胞在發育時，RNA 的數量會增加；大腸桿菌如果長得越快，RNA 的數量也會增加的越多；此外，若有病毒感染一個細胞時，在蛋白質合成之前會先合成 RNA，而且有些病毒的基因遺傳物質是 RNA 而非 DNA，如果將此病毒的 RNA 萃取出來注入寄主細胞中，也可以製造新的病毒，所以 RNA 被認定可以攜帶遺傳訊息。

(三) 蛋白質

蛋白質（Protein）是一種複雜的有機化合物，是由胺基酸分子呈線性排列所形成，並透過形成肽鍵連接在一起，蛋白質的胺基酸序列是由對應基因所編碼。主要是遺傳密碼所編碼的 20 種「標準」胺基酸，在蛋白質中，某些胺基酸殘基還可以被翻譯後修飾而發生化學結構的變化，從而對蛋白質進行激活或調控。在 DNA 遺傳序列轉變成為蛋白質序列的過程中，必須加入 RNA 序列，也就是 DNA 必須先轉錄（transcription）成 RNA，RNA 再轉譯（translation）成蛋白質。DNA 轉錄成 RNA 就好像是中文的繁體字轉成簡體字，困難度較小。但 RNA 轉譯成蛋白質就像是中文要翻譯成英文一樣，複雜性和困難度較高。然而鹼基序列如何轉變成胺基酸序列呢？因為至少有 20 種不同的胺基酸，所以必須要有至少 20 種以上不同的鹼基排列順序[4][11]。

(四) 鹼基序列對蛋白質序列的轉換機制

密碼子 Codon 是以三個 DNA 或 RNA 鹼基為一組，下圖是 1954 年時科學家推論出來的結果，它暗示了鹼基與蛋白質互換的機制及生成原理

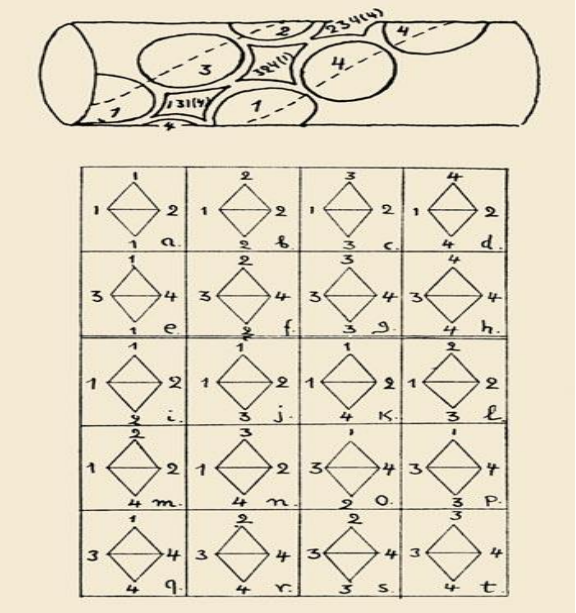

圖 3-1：George Gamow's diamond code

因為胺基酸有 20 種，而 DNA 或 RNA 只有 4 種鹼基，它們之間要如何配對呢？這就好像英文只有 26 個字母，而中文卻有上萬個字，它們之間要如何對應？4 個 DNA 或 RNA 鹼基中，若以兩個為一組來對應一個胺基酸，則只能產生 16 種組合（4^2=16），顯然無法對應 20 種胺基酸；若以三個為一組，則可產生 64 種組合（4^3=64）；若四個為一組，則可產生 256 種組合（4^4=256）。所以有可能是以三個為一組，但也有可能是以四個為一組。最後生物學家利用噬菌體的交配來發現 DNA 的語言應該是以 3 個字為一組，所以 DNA 是以三個字為單位，來產生 64 種不同的組合，並利用多對一的函數映射關係對應到 20 種胺基酸[12]。

(五) 遺傳密碼表

最後解出的遺傳密碼如下表所示。甲硫胺酸(Methionine)是只有一個密碼子與之對應的胺基酸，也是一般通用的起始密碼子（initiation codon）。然而有極少數的生物例外，是使用 GUG 作為起始密碼子。UAA、UAG、UGA 是終止密碼子（stop codon），他們並不對應任何胺基酸，就好像是句子中的「句點」一樣，當轉譯時遇到終止密碼子，轉譯就會停止。

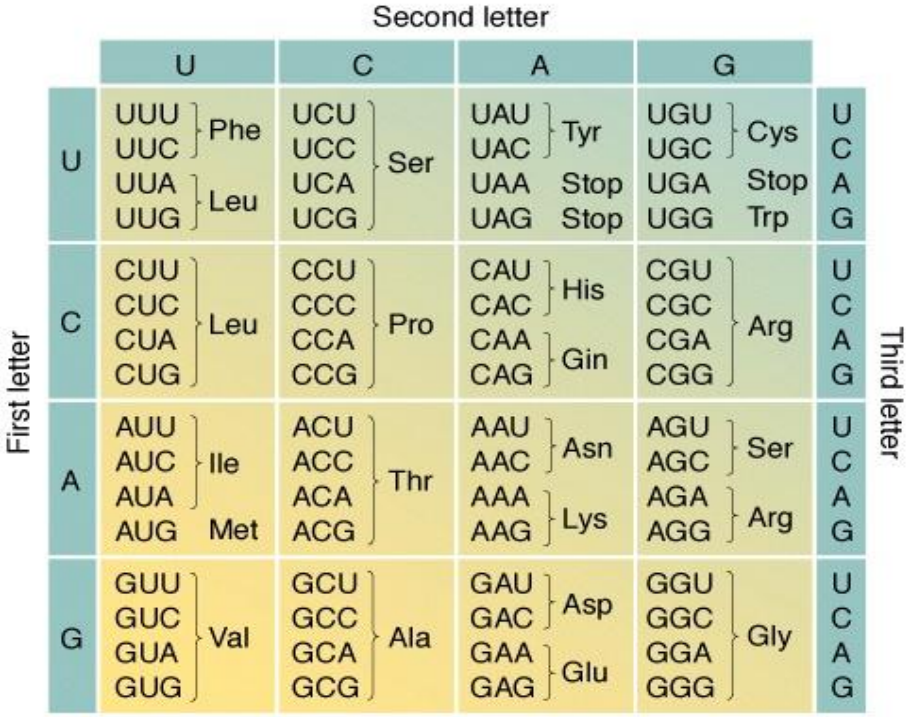

圖 3-2：遺傳密碼表

因為鹼基有 64 個（$4^3=64$）遺傳密碼子，卻只有 20 種胺基酸，所以一定有很多重複的對應，像精胺酸（Arginine）是擁有最多重複對應的胺基酸，可由六種不同的密碼子產生。此外，密碼子的第三個位置似乎較不重要。而且相似的胺基酸都排在一起，所以如果不小心發生突變的話，產生的胺基酸也會類似，例如：GAU 變成 GAA 會使天冬胺酸（Aspartic）變成麩胺酸（Glutamic），而 Asp 和 Glu 都是酸，屬於親水性，它們的差別只有一個碳；Thr 和 Ala 的密碼是第一個鹼基的 A 變成 G，而這兩個胺基酸都屬於疏水性如表 3-1，這在演化上顯然有很重要的意義。

表 3-1：蛋白質常用的 22 組胺基酸

分 類	名 稱		縮寫	
1. 唯一對稱胺基酸	1. 甘胺酸	Glycine	Gly	G
2. 含飽和碳氫基團	2. 丙胺酸	Alanine	Ala	A
	3. 纈胺酸*	Valine	Val	V
	4. 白胺酸*	Leucine	Leu	L
	5. 異白胺酸*	Isoleucine	Ile	I
3. 含芳香基團	6. 苯丙胺酸*	Phenylalanine	Phe	F
	7. 酪胺酸	Tyrosine	Tyr	Y
	8. 色胺酸*	Tryptophan	Trp	W
	9. 組胺酸*	Histidine	His	H
4. 含額外酸基 [及其醯胺]	10. 天冬胺酸	Aspartic acid	Asp	D
	11. 天冬醯胺酸	Asparagine	Asn	N
	12. 麩胺酸	Glutamic acid	Glu	E
	13. 麩醯胺酸	Glutamine	Gln	Q
5. 含額外胺基	14. 離胺酸*	Lysine	Lys	K
	15. 精胺酸*	Arginine	Arg	R
6. 含有醇基	16. 絲胺酸	Serine	Ser	S
	17. 蘇胺酸*	Threonine	Thr	T
	18. OH-脯胺酸	Hydroxy Pro		
7. 含有硫	19. 甲硫胺酸*	Methionine	Met	M
	20. 胱胺酸	Cysteine	Cys	C
	21. 雙胱胺酸 (3)	Cystine		
8. 環狀的亞胺酸	22. 脯胺酸 (3)	Proline	Pro	P

生物是一個資訊的系統，在細胞內的遺傳系統是由非常多不同的分子來運作，然而它的中心遺傳解碼器卻都在這裏，奇妙的是生物怎麼知道可以利用翻譯這種方法，這是整個演化上很重要的突破，蛋白質就像是資訊系統的硬體，而 DNA 則是軟體，主要功能為儲存、複製與傳遞資訊。這兩者之間如何互相結合，主要關鍵就在於翻譯的系統[2][4][11]。表 3-2 將表 3-1 整理成 22 個特徵向量，以進行數據分析：

表 3-2：胺基酸變數設計

特徵	Ala (A)	Cys (C)	Asp (D)	Glu (E)	Phe (F)	Gly (G)	His (H)	Ile (I)	Lys (K)	Leu (L)	Met (M)
編號	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
特徵	Asn (N)	Pro (P)	Gln (Q)	Arg (R)	Ser (S)	Thr (T)	Val (V)	Trp (W)	Tyr (Y)	*	a+t
編號	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x_{19}	x_{20}	x_{21}	x_{22}

(六)模糊理論（Fuzzy Theory）

實際上是由模糊集合（Fuzzy Set）、模糊關係（Fuzzy Relation）、模糊邏輯(Fuzzy Logic)、模糊控制(Fuzzy Control)、模糊量測(Fuzzy Measure)… 等理論整合而成。主要是將模糊概念予以量化的學問，起源於 1965 年扎德（L.A. Zadeh）教授所發表的著名論文－「模糊集合」[7]。文中首次提出表達事物模糊性的重要概念：隸屬函數，從而突破了 19 世紀末笛卡爾的經典集合理論,奠定模糊理論的基礎.模糊理論以模糊集合為基礎，以研究不確定事物為目標，接受模糊現象存在的事實，根據不清晰訊息，透過近似推理(Approximation Reasoning)過程而得到正確結果，這與人腦「過程模糊，結論清晰」的思維方式極其類似，因此已被廣泛的應用於各種不同領域的智慧型系統中[5][6][7]。

傳統分類與模糊分類的比較

傳統分類演算方法可視為模糊分類的特例，也就是在傳統的分類演算方法隸屬度在隸屬某一類時視為 1，在此時其他類隸屬度視為 0。故兩者的目標函數差異，在傳統分類演算方法為 $J(V)$，在模糊分類時多了隸屬度、即 $J(V,U)$。因此透過了隸屬度的調整強化了分類的有效性及合理性，而此點正是本文利用其方式來修正傳統 DNA 分類演算方法的最佳證明[7]。

(七)特徵的提取

為了有效地實現分類識別及親緣判定，因此利用數學方法中的模式識別計算方法，首先根據被識別的對象產生一組基本特徵，並對基本特徵進行變換，以得到最能反映分類本質的特徵，這就是特徵形成與提取的過程，因此列舉了儘可能完備的特徵參數集之後、如本文 $X=[x_1,x_2,\cdots,x_{22}]$。

（八）特徵的形成

由於在不用於編碼蛋白質的序列片段中，A 和 T 的含量特別多些，因此將 A 和 T 是否特別豐富作為一個特徵。在表 3-2 中以參數 x_{22} =A 和 T 出現的頻率之和。由於 A，T，C，G 這 4 個字符組成了 64 種不同的 3 字元串，這 64 種 3 字元串構成生物蛋白質的 20 種胺基酸，在參考文獻[12]的 Figur2 中，給出了這 20 種胺基酸的編碼（見圖 3-2）。因此，在計算 3 字元串的出現頻率時，將根據圖 1 Brian Hayes 在論文中給出的圖形，將代表同一種胺基酸的 3 字元串合成一類，只統計 20 類 3 字元串的出現頻率。（不考慮字元串在序列片段中的起始位置，也就採用“滾動”算法[12]。如 acgtcc 中就有 acg,cgt,gtc,tcc 共 4 個 3 字元串）。

二、實驗步驟

（一）將 DNA/RNA 序列轉成胺基酸序列並進行資料量化

由於三個鹼基組成一個胺基酸，且序列中起點未知，所以鹼基轉成蛋白質是利用滾動方式計算，因此我們先將所有鹼基轉成蛋白質中的 20 組胺基酸，並各增加 1 組鹼基轉胺基酸的基因終止符及 a 和 t 的鹼基配對組，如表 3-2 胺基酸變數設計，做為本研究的 22 筆特徵向量，再統計其出現的相對次數頻率以作為特徵提取之用，以將原始字串轉為可供分析之數據。其中前 20 組 DNA/RNA 序列加上基因終止符及 a 和 t 的鹼基配對組，轉成胺基

酸序列資料經滾動分析如下矩陣表示式 3.1-1，我們將透過相對次數轉換方式，將式 3.1-1 以 60 筆數值表示，先利用 sort($\min:(x_k^i - x_l^j)^2$)、$i \neq j$、$k \neq l$，$i = 1,2,...,c$ ， $k = 1,2,...,n$ ， $j = 1,2,...,c$ ， $l = 1,2,...,n$ 。來求出親緣性最佳的前 20 組滾動參數再加上終止符及 a 和 t 的鹼基配對組，以判斷最佳的變數 $x_k^{(i)}$ ，即式 3.1-2。

$$\begin{bmatrix} x_{1,n}^{(1)} & x_{2,n}^{(1)} & x_{3,n}^{(1)} \ddots & & x_{22,n}^{(1)} \\ x_{1,n}^{(2)} & x_{2,n}^{(2)} & x_{3,n}^{(2)} & \ddots & x_{22,n}^{(2)} \\ x_{1,n}^{(3)} & x_{2,n}^{(3)} & x_{3,n}^{(3)} & & \ddots x_{22,n}^{(3)} \end{bmatrix}$$ 、n 為樣本數----------(式 3.1-1)

$$\begin{bmatrix} x_{1,n}^{(i)} & x_{2,n}^{(i)} & \cdots & x_{22,n}^{(i)} \end{bmatrix}$$ 、$i = 1,2,...,c$ 、n 為樣本數---------(式 3.1-2)

經程式分類之後發現，若將終止符及 a 和 t 的鹼基配對組視為段落標號，共有 22 個參數存在，因此令 $X_{k,n}^{(i)} = \{x_{1,n}^{(i)}, x_{2,n}^{(i)}, x_{3,n}^{(i)}, \cdots, x_{21,n}^{(i)}, x_{22,n}^{(i)}\}$ ， $x_{k,n}^{(i)}$ 表示第 k 個特徵在 $W_i (i = 1,2)$ 類中出現的頻率，並將變數的個數調整使參數維度降至足以代表整體的樣本參數。

（二）模糊聚類方式設計

確定代表訊息的胺基酸序列的每一個胺基酸，利用模糊聚類的方式來確定代表 A、B 兩類中鹼基序列訊息的胺基酸序列，並利用歐氏距離來量測彼此間的實際差距。由於模糊聚類 Fuzzy C-Mean（FCM）演算法是利用模糊群集的分類方法，一般在進行模糊分類時，都是假設目標函數為：

$$J(U,V) = \sum_{i=1}^{d} \sum_{j=1}^{c} \sum_{k=1}^{n} u_{i,j,k}^{m} \mid x_{i,k} - v_{i.j} \mid^2$$ ----------(式 3.3-1)

其中

$x_1, \cdots, x_n$ 是樣本變數；

d 表示樣本胺基酸序列變數的參數個數；

$V_i = \{v_{i,1}, \cdots, v_{i,c}\}$ 是第 i 筆資料，分類的平均值（群心）向量集；

$U_i = [u_{i,k}]$ 為一 $c \times n$ 矩陣，這裡 $u_{i,j,k}$ 是第 k 次輸入樣本的第 j 個隸屬值，且滿足

$$0 \le u_{i,j,k} \le 1 \quad i = 1, 2, \ldots, d \ ; \ j = 1, 2, \ldots, c \ ; \ k = 1, 2, \ldots, n \text{------(式 3.3-2)}$$

$$\sum_{j=1}^{c} u_{i,j,k} = 1 \quad i = 1, 2, \ldots, d \ ; \ j = 1, 2, \ldots, c \ ; \ k = 1, 2, \ldots, n \text{------(式 3.3-3)}$$

$$0 < \sum_{k=1}^{n} u_{i,j,k} < n \quad i = 1, 2, \ldots, d \ ; \ j = 1, 2, \ldots, c \ ; \ k = 1, 2, \ldots, n \text{------(式 3.3-4)}$$

$m \in [1, \infty)$ 為一指數加權因子。

在此目標函數是每一個輸入樣本的歐氏距離平方和，而每一個歐氏距離的加權值是由模糊隸屬度調整。上述的 $x_{i,k}$ 表示胺基酸序列中的第 i 筆的第 k 個，$v_{i,j}$ 代表序列中的第 i 筆的所分的類數 j ，而演算法的迭代是利用式 3-5 及式 3-6：

$$v_{i,j} = \frac{1}{\sum_{k=1}^{n} u_{i,j,k}^{m}} \sum_{k=1}^{n} u_{i,j,k}^{m} \cdot x_{i,j,k} \quad j = 1, 2, \ldots, c \text{----------(式 3.3-5)}$$

$$u_{i,j,k} = \frac{\left[\frac{1}{|x_{i,k} - v_{i,j}|} \right]^{2/(m-1)}}{\sum_{j=1}^{c} \left[\frac{1}{|x_{i,k} - v_{i,j}|} \right]^{2/(m-1)}} \quad j = 1, 2, \ldots, c \ ; \ k = 1, 2, \ldots, n \text{-----(式 3.3-6)}$$

演算法執行步驟：

為了計算群心，因此所有的輸入樣本均被隸屬度考量為有等值的貢獻，且經迭代之後每一筆的樣本隸屬度，將被修正為樣本值與樣本所在群心的歐氏距離差距。

FCM 演算法步驟

1.隨機初始化$U^{(0)}$、$V^{(0)}$、ε，設定迭代次數α由 1 開始，設定參數維度d、分類個數c及資料筆數n，及指數加權m。

2.由給定的$U_i^{(\alpha)}$、及式 3-5 計算$V_i^{(\alpha)}$。

3.由給定的$V_i^{(\alpha)}$、及式 3-6 計算$U_i^{(\alpha)}$。

4.若 $\max|u_{i,j,k}^{(\alpha)} - u_{i,j,k}^{(\alpha-1)}| \leq \varepsilon$ ----------(式 3.3-7)，則終止迭代；否則令$\alpha = \alpha + 1$且回到步驟 2.，其中ε是事先設定的誤差容許值。

（三）模糊聚類程式模擬部分

由於一般使用維度遞減的方式在進行分類預測時或多或少，仍然有些樣本無法完全予以分類認定，但利用 Fuzzy C-Mean 的優點是將所有樣本參數d均列入分析，發現最後可正確分類出的序列組變多了，但為了確認我們分類結果是正確的，所以我們也利用**連續三筆資料**的結果均是**同類**時，才將其歸屬於成功的分類，因此可以得出（**百分 90.65**）的正確率，再比對事先設計的人工樣本，發現此次分類的成功。

（四）程式模擬部分圖示

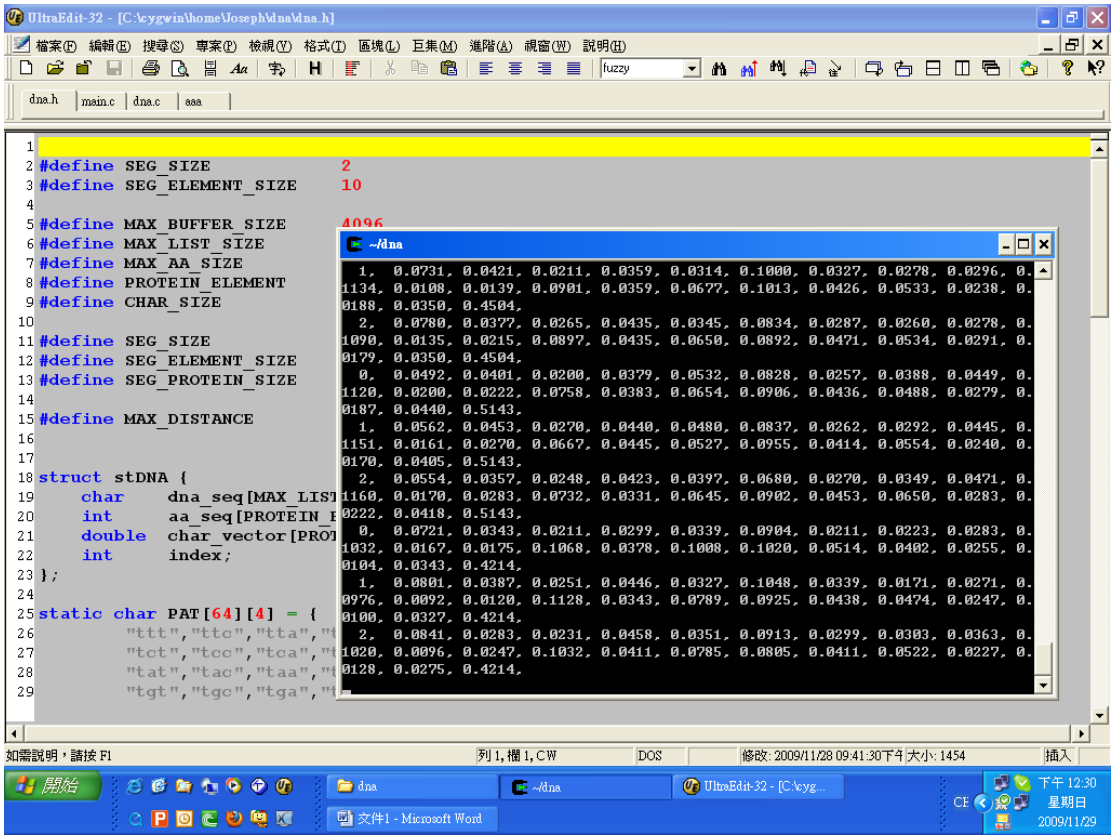

圖 3-3：在 Ultra Edit 及 Dos 模式之下程式模擬狀況（一）

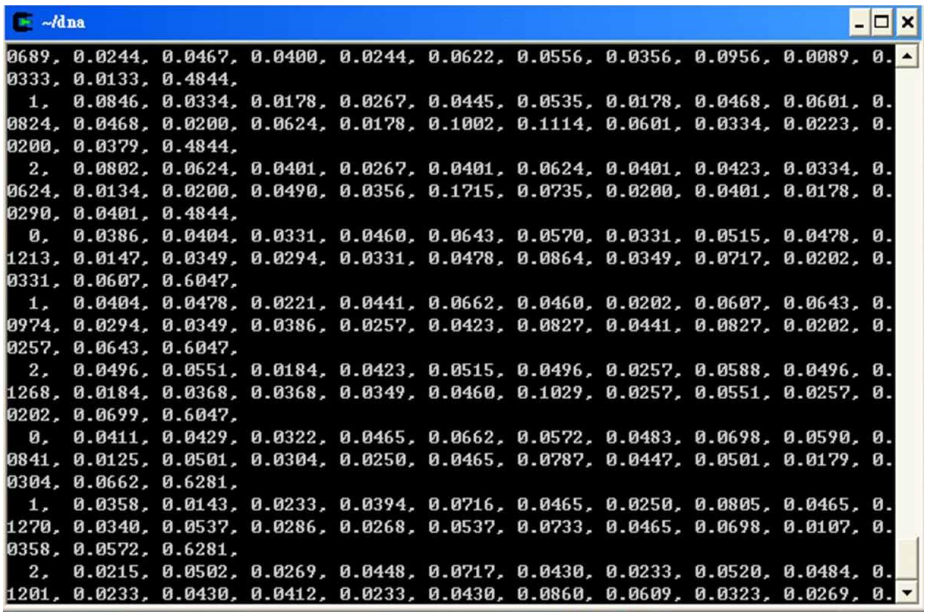

圖 3-4：在 Dos 模式之下程式模擬狀況（一）

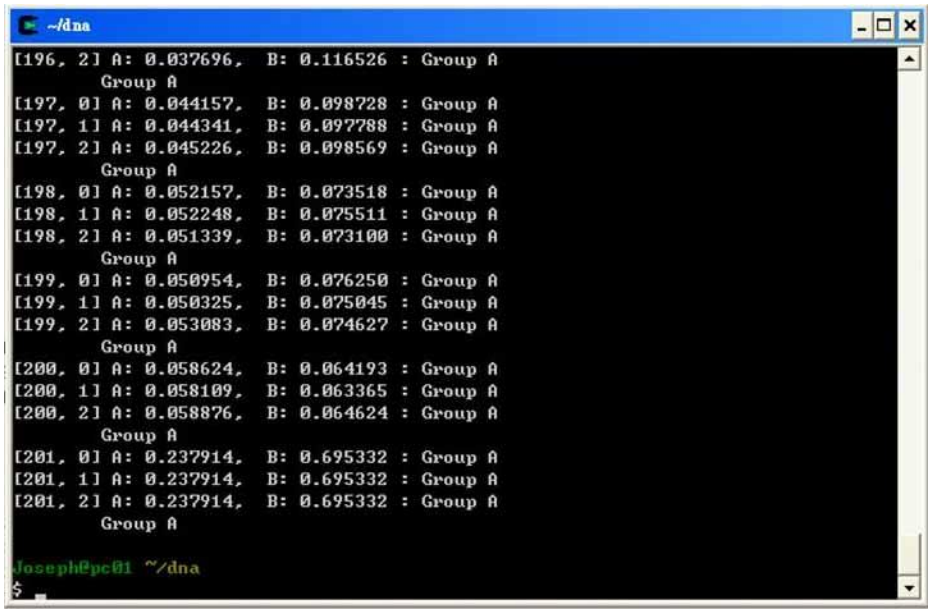

圖 3-5：在 Dos 模式之下程式模擬狀況（二）

（五）程式模擬所用設備

硬體部分：CPU:AMD XP-2000 時脈 1.67G-HZ。

RAM:1GB。

軟體部分：Ultra Edit 中文 9.0 版。

gcc version 3.4.4 (cygming special, gdc 0.12, using dmd 0.125)

Office2003 (Excel 2003/Word 2003)。

系統計算時間：以 182 筆自然 Dna 樣本計算，小於 1 秒。

肆、 討論(Discussion)

（一）DNA 序列樣本的假設

（1）任意一段 DNA 序列樣本是隨機的被歸屬於 A 類或 B 類。

（2）暫不考量胺基酸排列順序對 DNA 序列所表示的生物訊息。

（3）暫不考量 DNA 序列中的鹼基三聯組的起始位置所表達的分類結果。

（4）假設原題目所提供的參考樣本內含充分的資訊量。

（二）模型建立與 DNA 序列樣本數據分析求解

因為題目給出了 40 個已知為兩個類別的人工製造的 DNA 序列，要求我們從中提取特徵，建構分類方法，從而對 182 個自然 DNA 序列進行分類。這是模式識別中的“有人管理分類”問題，即事先規定了分類的標準和種類的數目（即前 40 筆人工 DNA 樣本），透過大批已知樣本的訊息處理找出規律，再用電腦分析預測未知，給出的已知類別的樣本稱為學習樣本。對於此類問題，將透過建立分類數學模型、檢查分類模型的效率、預測未知結果，這幾個步驟來進行，模式識別計算。本問題的學習樣本數為 40 筆。並透過研究 4 個字符 A,T,C,G 在 DNA 序列中的排列、組合特性，及字符和字元串的排列在序列中出現的頻率，從中提取 DNA 序列的形成的架構特徵參數。

（三）本文所提之數學模型之電腦數值模擬結果設計

由程式模擬結果說明，誤差容許值設定為 0.0001，結果說明如下

[20, 0] A: 0.040592,　B: 0.135095 : Group A

[20, 1] A: 0.038310,　B: 0.134566 : Group A

[20, 2] A: 0.039490,　B: 0.130869 : Group A

像這一種的是 A 類。

[26, 0] A: 0.077669,　B: 0.062116 : Group B

[26, 1] A: 0.118124,　B: 0.054009 : Group B

[26, 2] A: 0.101919,　B: 0.060363 : Group B

像這一種的是 B 類。

[31, 0] A: 0.064285,　B: 0.076378 : Group A

[31, 1] A: 0.105925,　B: 0.070017 : Group B

[31, 2] A: 0.096313,　B: 0.075384 : Group B

像這一種的是無法分類。

本次分類方式是利用人工樣本 1-10 筆算出的 22 個群心參數為 A 類，人工樣本 11-20 算出的 22 個群心參數為 B 類；再由人工樣本 21-40 群心參數直接區分檢查出來的結果正確度有 95%,比傳統[1]上利用 K-MEAN 結合 FISHER 線性函數的主成分方式，分類獲得的結果 90%而且高出 5%

學習樣本

Set_01_10 可完全分類，但有一組分錯

實驗結果：Set_11_20 可完全分類，

測試樣本

實驗結果：Set_21_40, 有一組無法分類

檢定樣本

實驗結果：182 有 12 組無法分類及 5 組資料有誤，正確度

(182-12-5)/182=90.65%。

我們先模擬了原文所提以 K-MEAN 結合 FISHER 線性函數的維度遞減方式的出的正確度有(182-43)/182=76.37%，並比較原文所提以 K-MEAN 結合 FISHER 線性函數的維度遞減的各種組合中的正確度最高只有 73.1%，且原文亦提及電腦計算誤差為正負 9%。可見結合模糊理論的方式在 DNA 的分類鑑定中具有良好的效果。

伍、 結論(Results)

一、本文對原始模型的改進方向及應用修正

1. 原文數學模型的改進：原文所提的模型沒考慮 DNA 序列的實際特性，當序列變得很多很長很複雜時，分類的準確性可能會降低，因此應增加對 DNA 序列的生物特性的考慮。
2. 原文數學模型的應用限制：原始模型採用系統聚類方式，對一般

的“有人監督分類”問題的求解有意義，對研究 DNA 序列的規律性和架構提供了一種有效的分類模型，有利於加快對基因組的研究有實質意義。但對“沒有人監督的分類”就會有出現分類錯誤，嚴重失效的問題發生。

二、本文所提新模型的改進方向及推展

本研究成功地提出以模糊分類來解決沒有人監督的分類方式[1]，並將其結果作為本文研究的目標。另外又嘗試從實際 DNA 序列樣本來修正及檢驗所提之演算方法，是否對實際生物特性存有具體意義，其結果可以說此次的數學模型方法有著突破的結果。

（一）分類模型的檢驗

前面建立的分類數學模型對 20 個測試樣本（21-40 組的人工學習樣本)進行了正確分類。為了進一步檢查分類模型的信度及效度，採用的方法是：先利用（01-20 組的人工樣本)分類進行群心的學習，之後再將（21-40 組的人工學習樣本)每組所偏移出的 3 組樣本，與 01-20 組的群心同時進行模糊分類檢測，若均為同一類則分為同一類，也就是說此為一成功的分類結果，然後將此預報成功率作為預報能力的指標。再對 21-40 筆人工的 DNA 序列進行分類，給出的樣本作預測為有效分類成功，且對比原文的結果[1]發現正確度提高了 5%。

（二）未知樣本的分類結果預測

由前面建立的數學模型對題目所給的未知類型的 182 個自然序列進行預報。結果在對未知樣本的實際比對中發現，當對 182 筆自然界的 DNA 序列進行分類中，發現正確度提由 76%提高至 90%，此點很明顯地表示此次分類的成功，並再次檢驗了方法的有效性及正確性。

（三）原始模型的優缺點分析

優點：

1、提出解決未監督式的問題，並成功地建立解決這類難題的數學模型，且可運用到實例中。

2、為解決複雜的分類問題並確定其最後結果，我們以偏移三次來確定其最後的分類正確性，而且模型假設條件少，因而能準確地反映實際情況，可靠性高。

3、採用系統化分析，逐漸深入，提升了準確性，避免了在一些細節問題上的糾纏。

缺點：由於只考慮了 DNA 樣本序列中元串出現的頻率作為特徵，並未列入基因突變的考量，故 DNA 序列的分類可能與實際情況不一定完全相符。

三、研究總結

本次實驗以模糊分類方式來設計出一套有效的 DNA 序列的分類方法，並成功了在人工預測樣本中得出 95%的分類正確性，亦在自然樣本得出 90%以上的分類正確性，因此可說此次的研究是一次成功的研究，未來將再結合由農改場所提供的生物實體樣本進行測驗，再檢驗此方法對 DNA 序列判別的一般性是否皆宜，另外亦將設計如何考量容許部分基因突變的情形以使此方法更具有實務應用價值。

陸、參考資料(References)

[1] 韓中庚、宋明武、邵廣紀，數學建模競賽，北京科學出版社，2007。

[2] 周純芬、彭洪文 編著，生物資訊輕輕鬆鬆學，合記圖書出版社，2005。

[3] 萬絢、閻嘉義，以亂數基礎分類法和 Fuzzy-C-Means 分群法分析土石流判釋問題，水保技術 4(1):37-46，2009。

[4] 遺傳密碼與基因表現-陽明大學普通生物學網路教材。

[5] 模糊理論筆記(Fuzzy Note)，http://irw.ncut.edu.tw/peterju/fuzzy.html。

[6] 林信成、彭啟峰，Oh!Fuzzy 模糊理論剖析 第 3 波出版社，1994。

[7] 楊敏生、楊鎮槐，模糊聚類及其應用，藍海文化，2009。

[8] k-means clustering，http://en.wikipedia.org/wiki/K-means_algorithm，[英文]。

[9] Dunn, J. C.，A fuzzy relative of the ISODATA process and its use in detecting compact, well-separated couster. J. Cybernet, 3,32-57，1974 [英文]。

[10] Bezdek，J. C.，Pattern Recognition with Fuzzy Objective Function Algorithms. New York: Plenum.，1980 [英文]。

[11] Brain Hayes，The Invention of the Genetic Code， American Scientist-Computing Science，Jan.-Feb.，1998。

[12] Zheru Chi、Hong Yan、Tuan Phan，Fuzzy Algoritms With Application to Image Processing and Pattern Recognition.，World Scientific Publish Co. Pte. Ltd，1996。

附件 1：FCM 對人工 21-40 及自然 182 樣本分類結果（由 20-221 編號）

測試用人工樣本

[20, 0] A: 0.040592, B: 0.135095 : Group A
[20, 1] A: 0.038310, B: 0.134566 : Group A
[20, 2] A: 0.039490, B: 0.130869 : Group A
Group A
[21, 0] A: 0.041128, B: 0.102640 : Group A
[21, 1] A: 0.043248, B: 0.099796 : Group A
[21, 2] A: 0.048550, B: 0.105683 : Group A
Group A
[22, 0] A: 0.040705, B: 0.116461 : Group A
[22, 1] A: 0.042913, B: 0.105418 : Group A
[22, 2] A: 0.046843, B: 0.119654 : Group A
Group A
[23, 0] A: 0.029850, B: 0.163612 : Group A
[23, 1] A: 0.033763, B: 0.159932 : Group A
[23, 2] A: 0.033587, B: 0.159419 : Group A
Group A
[24, 0] A: 0.059467, B: 0.293064 : Group A
[24, 1] A: 0.093566, B: 0.367103 : Group A
[24, 2] A: 0.072510, B: 0.364036 : Group A
Group A
[25, 0] A: 0.033758, B: 0.127226 : Group A
[25, 1] A: 0.031586, B: 0.120440 : Group A
[25, 2] A: 0.031254, B: 0.130509 : Group A
Group A
[26, 0] A: 0.077669, B: 0.062116 : Group B
[26, 1] A: 0.118124, B: 0.054009 : Group B
[26, 2] A: 0.101919, B: 0.060363 : Group B
Group B
[27, 0] A: 0.044563, B: 0.269252 : Group A
[27, 1] A: 0.051436, B: 0.266094 : Group A
[27, 2] A: 0.040518, B: 0.269154 : Group A
Group A
[28, 0] A: 0.054928, B: 0.157918 : Group A
[28, 1] A: 0.047538, B: 0.126145 : Group A
[28, 2] A: 0.035400, B: 0.136425 : Group A
Group A
[29, 0] A: 0.076203, B: 0.063614 : Group B
[29, 1] A: 0.092557, B: 0.051305 : Group B
[29, 2] A: 0.072580, B: 0.055565 : Group B
Group B
[30, 0] A: 0.041339, B: 0.151449 : Group A
[30, 1] A: 0.039847, B: 0.147076 : Group A
[30, 2] A: 0.043467, B: 0.151701 : Group A
Group A
[31, 0] A: 0.064285, B: 0.076378 : Group A
[31, 1] A: 0.105925, B: 0.070017 : Group B
[31, 2] A: 0.096313, B: 0.075384 : Group B
Ungroup
[32, 0] A: 0.059743, B: 0.314405 : Group A
[32, 1] A: 0.132987, B: 0.427359 : Group A
[32, 2] A: 0.099132, B: 0.421419 : Group A
Group A
[33, 0] A: 0.041530, B: 0.204735 : Group A
[33, 1] A: 0.042488, B: 0.202510 : Group A
[33, 2] A: 0.034993, B: 0.218908 : Group A
Group A
[34, 0] A: 0.043468, B: 0.270006 : Group A
[34, 1] A: 0.119246, B: 0.373589 : Group A
[34, 2] A: 0.077306, B: 0.372807 : Group A
Group A
[35, 0] A: 0.078138, B: 0.391422 : Group A
[35, 1] A: 0.118147, B: 0.454080 : Group A
[35, 2] A: 0.116465, B: 0.465393 : Group A
Group A
[36, 0] A: 0.053090, B: 0.135156 : Group A
[36, 1] A: 0.048882, B: 0.109311 : Group A
[36, 2] A: 0.035760, B: 0.117166 : Group A
Group A
[37, 0] A: 0.035675, B: 0.210047 : Group A
[37, 1] A: 0.060652, B: 0.245399 : Group A
[37, 2] A: 0.032476, B: 0.244940 : Group A

Group A

[38, 0] A: 0.043264, B: 0.189835 : Group A

[38, 1] A: 0.028329, B: 0.202893 : Group A

[38, 2] A: 0.035918, B: 0.183819 : Group A

Group A

[39, 0] A: 0.026684, B: 0.192436 : Group A

[39, 1] A: 0.029519, B: 0.193448 : Group A

[39, 2] A: 0.021057, B: 0.184099 : Group A

Group A

測試用自然樣本

[40, 0] A: 0.030187, B: 0.143611 : Group A

[40, 1] A: 0.034602, B: 0.146377 : Group A

[40, 2] A: 0.033861, B: 0.160388 : Group A

Group A

[41, 0] A: 0.062680, B: 0.064768 : Group A

[41, 1] A: 0.063670, B: 0.064996 : Group A

[41, 2] A: 0.068616, B: 0.069753 : Group A

Group A

[42, 0] A: 0.066418, B: 0.055916 : Group B

[42, 1] A: 0.072241, B: 0.048736 : Group B

[42, 2] A: 0.075350, B: 0.053513 : Group B

Group B

[43, 0] A: 0.056088, B: 0.069671 : Group A

[43, 1] A: 0.053645, B: 0.075264 : Group A

[43, 2] A: 0.053146, B: 0.070577 : Group A

Group A

[44, 0] A: 0.035389, B: 0.107822 : Group A

[44, 1] A: 0.031443, B: 0.108728 : Group A

[44, 2] A: 0.038705, B: 0.108742 : Group A

Group A

[45, 0] A: 0.055420, B: 0.075441 : Group A

[45, 1] A: 0.052581, B: 0.077228 : Group A

[45, 2] A: 0.056624, B: 0.068230 : Group A

Group A

[46, 0] A: 0.076846, B: 0.386157 : Group A

[46, 1] A: 0.053592, B: 0.338762 : Group A

[46, 2] A: 0.113127, B: 0.406235 : Group A

Group A

[47, 0] A: 0.053059, B: 0.074570 : Group A

[47, 1] A: 0.049921, B: 0.068932 : Group A

[47, 2] A: 0.052325, B: 0.070110 : Group A

Group A

[48, 0] A: 0.035115, B: 0.143609 : Group A

[48, 1] A: 0.031709, B: 0.151522 : Group A

[48, 2] A: 0.035497, B: 0.147740 : Group A

Group A

[49, 0] A: 0.055773, B: 0.075761 : Group A

[49, 1] A: 0.056219, B: 0.075534 : Group A

[49, 2] A: 0.056503, B: 0.077742 : Group A

Group A

[50, 0] A: 0.032841, B: 0.181216 : Group A

[50, 1] A: 0.026891, B: 0.174830 : Group A

[50, 2] A: 0.031802, B: 0.182111 : Group A

Group A

[51, 0] A: 0.064417, B: 0.369178 : Group A

[51, 1] A: 0.063075, B: 0.338534 : Group A

[51, 2] A: 0.108357, B: 0.394562 : Group A

Group A

[52, 0] A: 0.025456, B: 0.152804 : Group A

[52, 1] A: 0.031945, B: 0.154153 : Group A

[52, 2] A: 0.026445, B: 0.155344 : Group A

Group A

[53, 0] A: 0.088287, B: 0.071686 : Group B

[53, 1] A: 0.094982, B: 0.055420 : Group B

[53, 2] A: 0.070723, B: 0.059935 : Group B

Group B

[54, 0] A: 0.028719, B: 0.179867 : Group A

[54, 1] A: 0.030042, B: 0.180871 : Group A

[54, 2] A: 0.032864, B: 0.181127 : Group A

Group A

[55, 0] A: 0.033339, B: 0.126458 : Group A

[55, 1] A: 0.034192, B: 0.117229 : Group A

[55, 2] A: 0.029820, B: 0.123895 : Group A

Group A

[56, 0] A: 0.038584, B: 0.128802 : Group A

[56, 1] A: 0.043277, B: 0.125134 : Group A

[56, 2] A: 0.043762, B: 0.135905 : Group A
Group A
[57, 0] A: 0.026596, B: 0.117710 : Group A
[57, 1] A: 0.038345, B: 0.127981 : Group A
[57, 2] A: 0.035371, B: 0.117813 : Group A
Group A
[58, 0] A: 0.030591, B: 0.175193 : Group A
[58, 1] A: 0.029614, B: 0.175416 : Group A
[58, 2] A: 0.030819, B: 0.176958 : Group A
Group A
[59, 0] A: 0.059169, B: 0.084114 : Group A
[59, 1] A: 0.077413, B: 0.079049 : Group A
[59, 2] A: 0.064022, B: 0.089755 : Group A
Group A
[60, 0] A: 0.028626, B: 0.168419 : Group A
[60, 1] A: 0.035652, B: 0.181584 : Group A
[60, 2] A: 0.030883, B: 0.174398 : Group A
Group A
[61, 0] A: 0.030171, B: 0.180147 : Group A
[61, 1] A: 0.021789, B: 0.184367 : Group A
[61, 2] A: 0.030896, B: 0.176202 : Group A
Group A
[62, 0] A: 0.149602, B: 0.026864 : Group B
[62, 1] A: 0.150362, B: 0.021942 : Group B
[62, 2] A: 0.137805, B: 0.024250 : Group B
Group B
[63, 0] A: 0.030181, B: 0.170592 : Group A
[63, 1] A: 0.034012, B: 0.178827 : Group A
[63, 2] A: 0.032342, B: 0.170282 : Group A
Group A
[64, 0] A: 0.049862, B: 0.209091 : Group A
[64, 1] A: 0.038663, B: 0.227768 : Group A
[64, 2] A: 0.031609, B: 0.206070 : Group A
Group A
[65, 0] A: 0.050187, B: 0.080048 : Group A
[65, 1] A: 0.050637, B: 0.086668 : Group A
[65, 2] A: 0.045369, B: 0.081817 : Group A
Group A
[66, 0] A: 0.036976, B: 0.111669 : Group A
[66, 1] A: 0.043651, B: 0.116582 : Group A
[66, 2] A: 0.031699, B: 0.123850 : Group A
Group A
[67, 0] A: 0.052916, B: 0.087471 : Group A
[67, 1] A: 0.052259, B: 0.090518 : Group A
[67, 2] A: 0.051011, B: 0.094071 : Group A
Group A
[68, 0] A: 0.052031, B: 0.312602 : Group A
[68, 1] A: 0.045201, B: 0.303699 : Group A
[68, 2] A: 0.060535, B: 0.329163 : Group A
Group A
[69, 0] A: 0.076752, B: 0.075515 : Group B
[69, 1] A: 0.063568, B: 0.071743 : Group A
[69, 2] A: 0.061109, B: 0.078574 : Group A
Ungroup
[70, 0] A: 0.043804, B: 0.111947 : Group A
[70, 1] A: 0.076788, B: 0.109115 : Group A
[70, 2] A: 0.053644, B: 0.118664 : Group A
Group A
[71, 0] A: 0.041845, B: 0.148645 : Group A
[71, 1] A: 0.028104, B: 0.167871 : Group A
[71, 2] A: 0.031782, B: 0.153738 : Group A
Group A
[72, 0] A: 0.034221, B: 0.114712 : Group A
[72, 1] A: 0.033983, B: 0.108169 : Group A
[72, 2] A: 0.031654, B: 0.114344 : Group A
Group A
[73, 0] A: 0.050694, B: 0.074798 : Group A
[73, 1] A: 0.052869, B: 0.067369 : Group A
[73, 2] A: 0.060463, B: 0.067947 : Group A
Group A
[74, 0] A: 0.030968, B: 0.205660 : Group A
[74, 1] A: 0.032933, B: 0.209068 : Group A
[74, 2] A: 0.026019, B: 0.207444 : Group A
Group A
[75, 0] A: 0.044974, B: 0.107857 : Group A
[75, 1] A: 0.040139, B: 0.109842 : Group A

[75, 2] A: 0.045708, B: 0.102819 : Group A
Group A
[76, 0] A: 0.037139, B: 0.098380 : Group A
[76, 1] A: 0.048748, B: 0.097854 : Group A
[76, 2] A: 0.046191, B: 0.094494 : Group A
Group A
[77, 0] A: 0.038619, B: 0.193659 : Group A
[77, 1] A: 0.058712, B: 0.197553 : Group A
[77, 2] A: 0.042663, B: 0.222467 : Group A
Group A
[78, 0] A: 0.053796, B: 0.291503 : Group A
[78, 1] A: 0.054855, B: 0.285245 : Group A
[78, 2] A: 0.050931, B: 0.286802 : Group A
Group A
[79, 0] A: 0.034883, B: 0.136454 : Group A
[79, 1] A: 0.025678, B: 0.129553 : Group A
[79, 2] A: 0.040072, B: 0.137204 : Group A
Group A
[80, 0] A: 0.055003, B: 0.279510 : Group A
[80, 1] A: 0.048266, B: 0.294479 : Group A
[80, 2] A: 0.040190, B: 0.254859 : Group A
Group A
[81, 0] A: 0.032052, B: 0.206366 : Group A
[81, 1] A: 0.029137, B: 0.198511 : Group A
[81, 2] A: 0.029248, B: 0.198400 : Group A
Group A
[82, 0] A: 0.043824, B: 0.099625 : Group A
[82, 1] A: 0.044952, B: 0.094516 : Group A
[82, 2] A: 0.048028, B: 0.096941 : Group A
Group A
[83, 0] A: 0.030794, B: 0.196723 : Group A
[83, 1] A: 0.035073, B: 0.200687 : Group A
[83, 2] A: 0.039038, B: 0.208756 : Group A
Group A
[84, 0] A: 0.105030, B: 0.046030 : Group B
[84, 1] A: 0.093061, B: 0.052468 : Group B
[84, 2] A: 0.101974, B: 0.047968 : Group B
Group B
[85, 0] A: 0.036224, B: 0.131024 : Group A
[85, 1] A: 0.034234, B: 0.127445 : Group A
[85, 2] A: 0.035741, B: 0.124071 : Group A
Group A
[86, 0] A: 0.042912, B: 0.288326 : Group A
[86, 1] A: 0.042456, B: 0.286263 : Group A
[86, 2] A: 0.048688, B: 0.295700 : Group A
Group A
[87, 0] A: 0.035917, B: 0.217020 : Group A
[87, 1] A: 0.031465, B: 0.202029 : Group A
[87, 2] A: 0.031248, B: 0.218278 : Group A
Group A
[88, 0] A: 0.048675, B: 0.291989 : Group A
[88, 1] A: 0.052818, B: 0.293856 : Group A
[88, 2] A: 0.049535, B: 0.289650 : Group A
Group A
[89, 0] A: 0.073622, B: 0.346609 : Group A
[89, 1] A: 0.073683, B: 0.362385 : Group A
[89, 2] A: 0.068728, B: 0.368982 : Group A
Group A
[90, 0] A: 0.237914, B: 0.695332 : Group A
[90, 1] A: 0.237914, B: 0.695332 : Group A
[90, 2] A: 0.237914, B: 0.695332 : Group A
Group A
[91, 0] A: 0.040173, B: 0.128420 : Group A
[91, 1] A: 0.036556, B: 0.124238 : Group A
[91, 2] A: 0.039036, B: 0.129248 : Group A
Group A
[92, 0] A: 0.039606, B: 0.251338 : Group A
[92, 1] A: 0.026307, B: 0.231845 : Group A
[92, 2] A: 0.047630, B: 0.259064 : Group A
Group A
[93, 0] A: 0.099271, B: 0.075920 : Group B
[93, 1] A: 0.127218, B: 0.068868 : Group B
[93, 2] A: 0.103128, B: 0.066131 : Group B
Group B
[94, 0] A: 0.123974, B: 0.061355 : Group B
[94, 1] A: 0.193775, B: 0.068582 : Group B

[94, 2] A: 0.175177, B: 0.062675 : Group B
Group B
[95, 0] A: 0.062416, B: 0.072789 : Group A
[95, 1] A: 0.060120, B: 0.068233 : Group A
[95, 2] A: 0.063801, B: 0.068705 : Group A
Group A
[96, 0] A: 0.030355, B: 0.110328 : Group A
[96, 1] A: 0.028988, B: 0.108922 : Group A
[96, 2] A: 0.033541, B: 0.115573 : Group A
Group A
[97, 0] A: 0.038993, B: 0.137301 : Group A
[97, 1] A: 0.036177, B: 0.130724 : Group A
[97, 2] A: 0.040308, B: 0.138582 : Group A
Group A
[98, 0] A: 0.041270, B: 0.284256 : Group A
[98, 1] A: 0.042543, B: 0.277499 : Group A
[98, 2] A: 0.045444, B: 0.289326 : Group A
Group A
[99, 0] A: 0.095633, B: 0.066319 : Group B
[99, 1] A: 0.079944, B: 0.053029 : Group B
[99, 2] A: 0.085509, B: 0.086506 : Group A
Ungroup
[100, 0] A: 0.042399, B: 0.189628 : Group A
[100, 1] A: 0.047050, B: 0.196364 : Group A
[100, 2] A: 0.041304, B: 0.189881 : Group A
Group A
[101, 0] A: 0.032460, B: 0.208790 : Group A
[101, 1] A: 0.030562, B: 0.201911 : Group A
[101, 2] A: 0.032427, B: 0.211486 : Group A
Group A
[102, 0] A: 0.053486, B: 0.084908 : Group A
[102, 1] A: 0.058068, B: 0.072524 : Group A
[102, 2] A: 0.051854, B: 0.079213 : Group A
Group A
[103, 0] A: 0.066644, B: 0.078485 : Group A
[103, 1] A: 0.080270, B: 0.058660 : Group B
[103, 2] A: 0.073290, B: 0.065723 : Group B
Ungroup
[104, 0] A: 0.074429, B: 0.052892 : Group B
[104, 1] A: 0.075433, B: 0.050783 : Group B
[104, 2] A: 0.070804, B: 0.048057 : Group B
Group B
[105, 0] A: 0.069226, B: 0.069422 : Group A
[105, 1] A: 0.078122, B: 0.054637 : Group B
[105, 2] A: 0.064670, B: 0.067109 : Group A
Ungroup
[106, 0] A: 0.035621, B: 0.117902 : Group A
[106, 1] A: 0.045050, B: 0.129125 : Group A
[106, 2] A: 0.048525, B: 0.113175 : Group A
Group A
[107, 0] A: 0.038632, B: 0.099551 : Group A
[107, 1] A: 0.049473, B: 0.107735 : Group A
[107, 2] A: 0.047933, B: 0.094576 : Group A
Group A
[108, 0] A: 0.035820, B: 0.264177 : Group A
[108, 1] A: 0.026719, B: 0.235682 : Group A
[108, 2] A: 0.046688, B: 0.254148 : Group A
Group A
[109, 0] A: 0.055282, B: 0.296348 : Group A
[109, 1] A: 0.058669, B: 0.293584 : Group A
[109, 2] A: 0.055537, B: 0.291260 : Group A
Group A
[110, 0] A: 0.041152, B: 0.269053 : Group A
[110, 1] A: 0.039453, B: 0.265407 : Group A
[110, 2] A: 0.036739, B: 0.260576 : Group A
Group A
[111, 0] A: 0.080486, B: 0.043976 : Group B
[111, 1] A: 0.081504, B: 0.044905 : Group B
[111, 2] A: 0.082074, B: 0.045150 : Group B
Group B
[112, 0] A: 0.036730, B: 0.136410 : Group A
[112, 1] A: 0.050835, B: 0.131515 : Group A
[112, 2] A: 0.032141, B: 0.149753 : Group A
Group A
[113, 0] A: 0.045662, B: 0.088741 : Group A
[113, 1] A: 0.048898, B: 0.093208 : Group A

[113, 2] A: 0.045397, B: 0.093066 : Group A
Group A
[114, 0] A: 0.033743, B: 0.136259 : Group A
[114, 1] A: 0.033130, B: 0.140191 : Group A
[114, 2] A: 0.035942, B: 0.136651 : Group A
Group A
[115, 0] A: 0.036658, B: 0.121307 : Group A
[115, 1] A: 0.036615, B: 0.121723 : Group A
[115, 2] A: 0.038725, B: 0.120767 : Group A
Group A
[116, 0] A: 0.061134, B: 0.067854 : Group A
[116, 1] A: 0.053515, B: 0.080440 : Group A
[116, 2] A: 0.051711, B: 0.083716 : Group A
Group A
[117, 0] A: 0.064758, B: 0.062481 : Group B
[117, 1] A: 0.067370, B: 0.056297 : Group B
[117, 2] A: 0.072353, B: 0.060244 : Group B
Group B
[118, 0] A: 0.047011, B: 0.079719 : Group A
[118, 1] A: 0.046913, B: 0.084431 : Group A
[118, 2] A: 0.046162, B: 0.086208 : Group A
Group A
[119, 0] A: 0.037495, B: 0.224642 : Group A
[119, 1] A: 0.062198, B: 0.273021 : Group A
[119, 2] A: 0.037738, B: 0.251283 : Group A
Group A
[120, 0] A: 0.237914, B: 0.695332 : Group A
[120, 1] A: 0.237914, B: 0.695332 : Group A
[120, 2] A: 0.237914, B: 0.695332 : Group A
Group A
[121, 0] A: 0.073844, B: 0.068154 : Group B
[121, 1] A: 0.063494, B: 0.069477 : Group A
[121, 2] A: 0.060265, B: 0.072907 : Group A
Ungroup
[122, 0] A: 0.063220, B: 0.060675 : Group B
[122, 1] A: 0.060250, B: 0.062956 : Group A
[122, 2] A: 0.062955, B: 0.061283 : Group B
Ungroup
[123, 0] A: 0.059441, B: 0.357907 : Group A
[123, 1] A: 0.072245, B: 0.351695 : Group A
[123, 2] A: 0.071920, B: 0.375074 : Group A
Group A
[124, 0] A: 0.237914, B: 0.695332 : Group A
[124, 1] A: 0.237914, B: 0.695332 : Group A
[124, 2] A: 0.237914, B: 0.695332 : Group A
Group A
[125, 0] A: 0.028913, B: 0.167096 : Group A
[125, 1] A: 0.033050, B: 0.169568 : Group A
[125, 2] A: 0.023991, B: 0.174228 : Group A
Group A
[126, 0] A: 0.072771, B: 0.056474 : Group B
[126, 1] A: 0.072718, B: 0.058083 : Group B
[126, 2] A: 0.070196, B: 0.056158 : Group B
Group B
[127, 0] A: 0.034212, B: 0.134877 : Group A
[127, 1] A: 0.037804, B: 0.140805 : Group A
[127, 2] A: 0.035127, B: 0.138434 : Group A
Group A
[128, 0] A: 0.067772, B: 0.335135 : Group A
[128, 1] A: 0.057678, B: 0.331279 : Group A
[128, 2] A: 0.054843, B: 0.300664 : Group A
Group A
[129, 0] A: 0.087649, B: 0.042587 : Group B
[129, 1] A: 0.085288, B: 0.043648 : Group B
[129, 2] A: 0.096053, B: 0.042528 : Group B
Group B
[130, 0] A: 0.036059, B: 0.192038 : Group A
[130, 1] A: 0.037801, B: 0.191269 : Group A
[130, 2] A: 0.040874, B: 0.190725 : Group A
Group A
[131, 0] A: 0.098577, B: 0.406082 : Group A
[131, 1] A: 0.082150, B: 0.377201 : Group A
[131, 2] A: 0.069753, B: 0.356614 : Group A
Group A
[132, 0] A: 0.036909, B: 0.117576 : Group A
[132, 1] A: 0.034726, B: 0.117666 : Group A

[132, 2] A: 0.033303, B: 0.118562 : Group A
Group A
[133, 0] A: 0.128563, B: 0.038401 : Group B
[133, 1] A: 0.119953, B: 0.044240 : Group B
[133, 2] A: 0.112670, B: 0.043578 : Group B
Group B
[134, 0] A: 0.039839, B: 0.236587 : Group A
[134, 1] A: 0.036298, B: 0.234255 : Group A
[134, 2] A: 0.032805, B: 0.231140 : Group A
Group A
[135, 0] A: 0.047800, B: 0.082921 : Group A
[135, 1] A: 0.060347, B: 0.082833 : Group A
[135, 2] A: 0.051122, B: 0.084542 : Group A
Group A
[136, 0] A: 0.054666, B: 0.325316 : Group A
[136, 1] A: 0.097909, B: 0.383968 : Group A
[136, 2] A: 0.075056, B: 0.393719 : Group A
Group A
[137, 0] A: 0.039386, B: 0.260304 : Group A
[137, 1] A: 0.042360, B: 0.270766 : Group A
[137, 2] A: 0.039926, B: 0.265722 : Group A
Group A
[138, 0] A: 0.072050, B: 0.057458 : Group B
[138, 1] A: 0.074349, B: 0.053220 : Group B
[138, 2] A: 0.075440, B: 0.067936 : Group B
Group B
[139, 0] A: 0.064613, B: 0.287229 : Group A
[139, 1] A: 0.040441, B: 0.275056 : Group A
[139, 2] A: 0.032587, B: 0.231912 : Group A
Group A
[140, 0] A: 0.058722, B: 0.075925 : Group A
[140, 1] A: 0.049712, B: 0.077689 : Group A
[140, 2] A: 0.049391, B: 0.074446 : Group A
Group A
[141, 0] A: 0.073841, B: 0.052288 : Group B
[141, 1] A: 0.072090, B: 0.050470 : Group B
[141, 2] A: 0.073060, B: 0.053629 : Group B
Group B
[142, 0] A: 0.052775, B: 0.077507 : Group A
[142, 1] A: 0.052456, B: 0.076981 : Group A
[142, 2] A: 0.052569, B: 0.076550 : Group A
Group A
[143, 0] A: 0.035444, B: 0.222583 : Group A
[143, 1] A: 0.032732, B: 0.225253 : Group A
[143, 2] A: 0.030973, B: 0.216634 : Group A
Group A
[144, 0] A: 0.055210, B: 0.071056 : Group A
[144, 1] A: 0.050211, B: 0.068052 : Group A
[144, 2] A: 0.054373, B: 0.072774 : Group A
Group A
[145, 0] A: 0.039022, B: 0.097634 : Group A
[145, 1] A: 0.039180, B: 0.097126 : Group A
[145, 2] A: 0.041335, B: 0.097661 : Group A
Group A
[146, 0] A: 0.080110, B: 0.054744 : Group B
[146, 1] A: 0.075535, B: 0.069732 : Group B
[146, 2] A: 0.063933, B: 0.071732 : Group A
Ungroup
[147, 0] A: 0.095414, B: 0.044982 : Group B
[147, 1] A: 0.112397, B: 0.041238 : Group B
[147, 2] A: 0.107686, B: 0.045449 : Group B
Group B
[148, 0] A: 0.070148, B: 0.055433 : Group B
[148, 1] A: 0.062636, B: 0.066177 : Group A
[148, 2] A: 0.061749, B: 0.070209 : Group A
Ungroup
[149, 0] A: 0.070718, B: 0.057980 : Group B
[149, 1] A: 0.064008, B: 0.068636 : Group A
[149, 2] A: 0.059501, B: 0.071323 : Group A
Ungroup
[150, 0] A: 0.237914, B: 0.695332 : Group A
[150, 1] A: 0.237914, B: 0.695332 : Group A
[150, 2] A: 0.237914, B: 0.695332 : Group A
Group A
[151, 0] A: 0.033107, B: 0.173034 : Group A
[151, 1] A: 0.033228, B: 0.176600 : Group A

[151, 2] A: 0.033346, B: 0.173800 : Group A
Group A
[152, 0] A: 0.034190, B: 0.119944 : Group A
[152, 1] A: 0.036248, B: 0.123364 : Group A
[152, 2] A: 0.036745, B: 0.122606 : Group A
Group A
[153, 0] A: 0.032633, B: 0.202941 : Group A
[153, 1] A: 0.033983, B: 0.207408 : Group A
[153, 2] A: 0.030784, B: 0.197759 : Group A
Group A
[154, 0] A: 0.034799, B: 0.136608 : Group A
[154, 1] A: 0.034350, B: 0.138427 : Group A
[154, 2] A: 0.034256, B: 0.137699 : Group A
Group A
[155, 0] A: 0.035949, B: 0.244849 : Group A
[155, 1] A: 0.044995, B: 0.263103 : Group A
[155, 2] A: 0.033987, B: 0.258418 : Group A
Group A
[156, 0] A: 0.046234, B: 0.141752 : Group A
[156, 1] A: 0.039795, B: 0.151648 : Group A
[156, 2] A: 0.032325, B: 0.139789 : Group A
Group A
[157, 0] A: 0.077606, B: 0.058365 : Group B
[157, 1] A: 0.091073, B: 0.045146 : Group B
[157, 2] A: 0.083329, B: 0.053128 : Group B
Group B
[158, 0] A: 0.033017, B: 0.160980 : Group A
[158, 1] A: 0.032898, B: 0.160889 : Group A
[158, 2] A: 0.030773, B: 0.156351 : Group A
Group A
[159, 0] A: 0.051060, B: 0.077623 : Group A
[159, 1] A: 0.051428, B: 0.075731 : Group A
[159, 2] A: 0.053899, B: 0.077452 : Group A
Group A
[160, 0] A: 0.047237, B: 0.259819 : Group A
[160, 1] A: 0.036644, B: 0.250613 : Group A
[160, 2] A: 0.039897, B: 0.245112 : Group A
Group A
[161, 0] A: 0.074122, B: 0.050615 : Group B
[161, 1] A: 0.067712, B: 0.062085 : Group B
[161, 2] A: 0.065162, B: 0.063529 : Group B
Group B
[162, 0] A: 0.083577, B: 0.052730 : Group B
[162, 1] A: 0.080148, B: 0.064765 : Group B
[162, 2] A: 0.066198, B: 0.067787 : Group A
Ungroup
[163, 0] A: 0.071560, B: 0.047554 : Group B
[163, 1] A: 0.073866, B: 0.048892 : Group B
[163, 2] A: 0.074335, B: 0.049323 : Group B
Group B
[164, 0] A: 0.029300, B: 0.141990 : Group A
[164, 1] A: 0.033503, B: 0.146085 : Group A
[164, 2] A: 0.029412, B: 0.143400 : Group A
Group A
[165, 0] A: 0.067470, B: 0.054234 : Group B
[165, 1] A: 0.063706, B: 0.063522 : Group B
[165, 2] A: 0.060349, B: 0.067790 : Group A
Ungroup
[166, 0] A: 0.237914, B: 0.695332 : Group A
[166, 1] A: 0.237914, B: 0.695332 : Group A
[166, 2] A: 0.237914, B: 0.695332 : Group A
Group A
[167, 0] A: 0.038944, B: 0.174993 : Group A
[167, 1] A: 0.033423, B: 0.181354 : Group A
[167, 2] A: 0.026182, B: 0.166205 : Group A
Group A
[168, 0] A: 0.038143, B: 0.102957 : Group A
[168, 1] A: 0.037785, B: 0.103228 : Group A
[168, 2] A: 0.039088, B: 0.105914 : Group A
Group A
[169, 0] A: 0.038869, B: 0.111442 : Group A
[169, 1] A: 0.041191, B: 0.112262 : Group A
[169, 2] A: 0.052088, B: 0.106197 : Group A
Group A
[170, 0] A: 0.081393, B: 0.047600 : Group B
[170, 1] A: 0.080152, B: 0.047658 : Group B

[170, 2] A: 0.079562, B: 0.046268 : Group B
[171, 0] A: 0.043077, B: 0.101533 : Group A
[171, 1] A: 0.045030, B: 0.109367 : Group A
[171, 2] A: 0.034395, B: 0.108127 : Group A
[172, 0] A: 0.034731, B: 0.117222 : Group A
[172, 1] A: 0.033037, B: 0.114947 : Group A
[172, 2] A: 0.034063, B: 0.116894 : Group A
[173, 0] A: 0.027190, B: 0.167166 : Group A
[173, 1] A: 0.026604, B: 0.166626 : Group A
[173, 2] A: 0.028716, B: 0.169438 : Group A
[174, 0] A: 0.040990, B: 0.273924 : Group A
[174, 1] A: 0.068793, B: 0.310605 : Group A
[174, 2] A: 0.053286, B: 0.328275 : Group A
[175, 0] A: 0.059550, B: 0.064331 : Group A
[175, 1] A: 0.058043, B: 0.065262 : Group A
[175, 2] A: 0.061498, B: 0.066806 : Group A
[176, 0] A: 0.033614, B: 0.130952 : Group A
[176, 1] A: 0.032631, B: 0.130759 : Group A
[176, 2] A: 0.036758, B: 0.126606 : Group A
[177, 0] A: 0.050980, B: 0.312243 : Group A
[177, 1] A: 0.056051, B: 0.322153 : Group A
[177, 2] A: 0.056048, B: 0.321441 : Group A
[178, 0] A: 0.061999, B: 0.065451 : Group A
[178, 1] A: 0.062904, B: 0.066294 : Group A
[178, 2] A: 0.061474, B: 0.066904 : Group A
[179, 0] A: 0.042177, B: 0.094907 : Group A
[179, 1] A: 0.043718, B: 0.086183 : Group A
[179, 2] A: 0.041936, B: 0.090631 : Group A
[180, 0] A: 0.090408, B: 0.041631 : Group B
[180, 1] A: 0.087129, B: 0.039693 : Group B
[180, 2] A: 0.087395, B: 0.039377 : Group B
[181, 0] A: 0.069254, B: 0.057201 : Group B
[181, 1] A: 0.069929, B: 0.061140 : Group B
[181, 2] A: 0.070275, B: 0.059940 : Group B
[182, 0] A: 0.079869, B: 0.047771 : Group B
[182, 1] A: 0.080452, B: 0.046415 : Group B
[182, 2] A: 0.079258, B: 0.047646 : Group B
[183, 0] A: 0.042242, B: 0.103274 : Group A
[183, 1] A: 0.042649, B: 0.102825 : Group A
[183, 2] A: 0.041167, B: 0.105077 : Group A
[184, 0] A: 0.039827, B: 0.106655 : Group A
[184, 1] A: 0.040505, B: 0.108406 : Group A
[184, 2] A: 0.041572, B: 0.106575 : Group A
[185, 0] A: 0.065721, B: 0.059979 : Group B
[185, 1] A: 0.066233, B: 0.061457 : Group B
[185, 2] A: 0.065848, B: 0.061089 : Group B
[186, 0] A: 0.038683, B: 0.114206 : Group A
[186, 1] A: 0.035944, B: 0.112560 : Group A
[186, 2] A: 0.039069, B: 0.114981 : Group A
[187, 0] A: 0.054112, B: 0.068384 : Group A
[187, 1] A: 0.056764, B: 0.068700 : Group A
[187, 2] A: 0.054128, B: 0.069682 : Group A
[188, 0] A: 0.237914, B: 0.695332 : Group A
[188, 1] A: 0.237914, B: 0.695332 : Group A
[188, 2] A: 0.237914, B: 0.695332 : Group A
[189, 0] A: 0.084200, B: 0.043609 : Group B
[189, 1] A: 0.084413, B: 0.044214 : Group B

[189, 2] A: 0.082301, B: 0.042908 : Group B
Group B
[190, 0] A: 0.069465, B: 0.310831 : Group A
[190, 1] A: 0.053539, B: 0.327455 : Group A
[190, 2] A: 0.040007, B: 0.273389 : Group A
Group A
[191, 0] A: 0.032841, B: 0.162814 : Group A
[191, 1] A: 0.032892, B: 0.160297 : Group A
[191, 2] A: 0.032419, B: 0.160138 : Group A
Group A
[192, 0] A: 0.053867, B: 0.073014 : Group A
[192, 1] A: 0.053325, B: 0.079313 : Group A
[192, 2] A: 0.051778, B: 0.074174 : Group A
Group A
[193, 0] A: 0.048330, B: 0.077910 : Group A
[193, 1] A: 0.052408, B: 0.074755 : Group A
[193, 2] A: 0.050462, B: 0.081979 : Group A
Group A
[194, 0] A: 0.051174, B: 0.076893 : Group A
[194, 1] A: 0.052953, B: 0.073955 : Group A
[194, 2] A: 0.053192, B: 0.070307 : Group A
Group A
[195, 0] A: 0.038089, B: 0.212507 : Group A
[195, 1] A: 0.034109, B: 0.210627 : Group A
[195, 2] A: 0.033700, B: 0.205707 : Group A
Group A
[196, 0] A: 0.039935, B: 0.118972 : Group A
[196, 1] A: 0.039888, B: 0.116245 : Group A
[196, 2] A: 0.037696, B: 0.116526 : Group A
Group A
[197, 0] A: 0.044157, B: 0.098728 : Group A
[197, 1] A: 0.044341, B: 0.097788 : Group A
[197, 2] A: 0.045226, B: 0.098569 : Group A
Group A
[198, 0] A: 0.052157, B: 0.073518 : Group A
[198, 1] A: 0.052248, B: 0.075511 : Group A
[198, 2] A: 0.051339, B: 0.073100 : Group A
Group A
[199, 0] A: 0.050954, B: 0.076250 : Group A
[199, 1] A: 0.050325, B: 0.075045 : Group A
[199, 2] A: 0.053083, B: 0.074627 : Group A
Group A
[200, 0] A: 0.058624, B: 0.064193 : Group A
[200, 1] A: 0.058109, B: 0.063365 : Group A
[200, 2] A: 0.058876, B: 0.064624 : Group A
Group A
[201, 0] A: 0.082387, B: 0.043323 : Group B
[201, 1] A: 0.083042, B: 0.041505 : Group B
[201, 2] A: 0.082039, B: 0.041796 : Group B
Group B

附件 2：FCM 的分類程式碼（Dna.c/Main.c/Dna.h）

Dna.c 程式碼

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "dna.h"

struct stDNA unsegDNA[256] = {0};
struct stDNA DNA[SEG_SIZE][SEG_ELEMENT_SIZE] = {0};
char tmpbuf[MAX_BUFFER_SIZE] = {0};

double V[SEG_SIZE][CHAR_SIZE] = {0};

int ObtainAmino(char *list)
{
int i;
for (i = 0; i < 64; i++)
{
if (memcmp(list, PAT[i], 3) == 0) {
// printf("%s ", PAT[i]);

return CHARACTERISTIC[i];
}
}

// printf("Error in ObtainAmino(), list: %s\n", list);
// exit(-1);
return -1;
}

int DNA2AA(struct stDNA* pDNA)
{
char *str;
int len;
int i,j;
int ch;

len = strlen(pDNA->dna_seq);
for (i = 0; i < PROTEIN_ELEMENT; i++)
{
for (j = i; j <= len-PROTEIN_ELEMENT; j += PROTEIN_ELEMENT) {
ch = ObtainAmino(pDNA->dna_seq+j);
if (ch < 0)
return ch;
pDNA->aa_seq[i][ch]++;
}
}

return 0;
}

void AA2Char(struct stDNA* pDNA)
{
double at_counter;
double char_counter;
int len = strlen(pDNA->dna_seq);
int i,j;

for (j = 0; j < PROTEIN_ELEMENT; j++) {
at_counter = 0;
for (i = 0; i < len; i++)
{
if (pDNA->dna_seq[i] == 'a' || pDNA->dna_seq[i] == 't') {
```

```
                    at_counter++;
                }
            }
        pDNA->aa_seq[j][CHAR_SIZE-1] =
                at_counter;
    pDNA->char_vector[j][CHAR_SIZE-1] =
            at_counter/len;

            char_counter = 0;
        for (i = 0; i < CHAR_SIZE-1; i++) {
          char_counter += pDNA->aa_seq[j][i];
                }
        for (i = 0; i < CHAR_SIZE-1; i++) {
              pDNA->char_vector[j][i] =
    pDNA->aa_seq[j][i]/char_counter;
                }
            }

                #if 1
    for (i = 0; i < PROTEIN_ELEMENT; i++) {
          printf("%3d, ", i);
        for (j = 0; j < CHAR_SIZE; j++) {
                printf(" %.4f,",
        pDNA->char_vector[i][j]);
    //                  printf(" %4d",
          pDNA->aa_seq[i][j]);
                }
              printf("\n");
              }
             #endif
              }

//      讀取鹼基序列再轉換成氨基酸特徵序列
    void ReadCharacteristic(char *filename, int
            seg_index)
              {
        FILE            *pFile;
      struct stDNA      *pstDNA;
            int             i;
          char*           pstr;

printf("\nObtain the characteristic of %s\n",
              filename);

      pFile = fopen(filename, "r");
        if (pFile == NULL) {
      printf("%s is wrong \n", filename);
              return ;
              }

            i = 0;
 if (seg_index < SEG_ELEMENT_SIZE) {
      pstDNA = &DNA[seg_index][i++];
              } else {
        pstDNA = &unsegDNA[i++];
              }

while (fgets(tmpbuf, MAX_BUFFER_SIZE,
              pFile))
              {
    pstr = strtok(tmpbuf, "0123456789:.>
              \t\n\r");
            if (pstr == NULL) {
  //                printf("got null\n");
          if (pstDNA->dna_seq[0] == 0) {
                    continue;
              }
                } else {
//                printf("got: (%d) %s\n",
          strlen(pstr), pstr);
        //      讀取 DNA 序列後，放置於
      stDNA 結構中
          strcat(pstDNA->dna_seq, pstr);
                continue;
              }
```

```
//                printf("[%3d]: %s (%6d)\n\n", i,
pstDNA->dna_seq, strlen(pstDNA->dna_seq));

                    #if 1
//      由 DNA 序列產生對應的 3 組胺基酸
                    序列
                if(DNA2AA(pstDNA)) {
               printf("%s\n", pstDNA->dna_seq);
                    printf("error\n");
              printf("error\n");
                    } else {
//      算出每組胺基酸序列對應的
                    特徵值
                AA2Char(pstDNA);
                    }
                #endif

        if (seg_index < SEG_ELEMENT_SIZE) {
              pstDNA = &DNA[seg_index][i++];
                    } else {
                pstDNA = &unsegDNA[i++];
                    }
                }

                fclose(pFile);
                }

double Char2Dis(double *charA, double *charB)
                {
                int             i;
            double      dSum = 0.0;

        for (i = 0; i < CHAR_SIZE; i++)
                {
        dSum += (charA[i] - charB[i])*(charA[i] -
                charB[i]);
                }
            return dSum;
                }

            void ObtainAAIndex(int seg_index)
                {
            double
DisArray[SEG_PROTEIN_SIZE][SEG_PROTEI
            N_SIZE] = {0};
            int             i,j,d,k;
             double      min;
          int           minA, minB;
  int           Vector[SEG_ELEMENT_SIZE] =
                {0};

    for (i = 0; i < SEG_PROTEIN_SIZE; i++)
                {
      for (j = i+1; j < SEG_PROTEIN_SIZE; j++)
                {
                DisArray[i][j] =
Char2Dis(DNA[seg_index][i/PROTEIN_ELEMEN
T].char_vector[i%PROTEIN_ELEMENT],

DNA[seg_index][j/PROTEIN_ELEMENT].char_
        vector[j%PROTEIN_ELEMENT]);
            DisArray[j][i] = DisArray[i][j];
                }
        DisArray[i][i] = MAX_DISTANCE;
                }

                #if 0
    for (i = 0; i < SEG_PROTEIN_SIZE; i++)
                {
      for (j = 0; j < SEG_PROTEIN_SIZE; j++)
                {
            printf(" %2.3lf", DisArray[i][j]);
                }
             printf("\n");
                }
            #endif
```

```
                min = MAX_DISTANCE;
        for (i = 0; i < SEG_PROTEIN_SIZE; i++)
                        {
          for (j = i+1; j < SEG_PROTEIN_SIZE; j++)
                        {
                if (DisArray[i][j] && (DisArray[i][j]
                < min) &&
                 (i/PROTEIN_ELEMENT !=
              j/PROTEIN_ELEMENT)) {
                        min = DisArray[i][j];
                                minA = i;
                                minB = j;
                                }
                        }
                    }

                Vector[0] = minA;
                Vector[1] = minB;

        for (d = 2; d < SEG_ELEMENT_SIZE; d++)
                        {
                min = MAX_DISTANCE;
                  for (i = 0; i < d; i++) {
                for (j = 0; j < SEG_PROTEIN_SIZE;
                        j++) {
                        if (DisArray[Vector[i]][j] &&
            (DisArray[Vector[i]][j] < min)) {

                                for (k = 0; k < d; k++) {
                                                if
            (Vector[k]/PROTEIN_ELEMENT ==
                        j/PROTEIN_ELEMENT)
                                                        break;
                                        }
                                if(k == d) {
                                        min =
                DisArray[Vector[i]][j];
                                        minA = Vector[i];
                                                minB = j;
                                                }
                                        }
                                }
                        }
                Vector[d] = minB;
                        for (i =
        (Vector[d]/PROTEIN_ELEMENT)*PROTEIN_EL
        EMENT, j = 0; j < PROTEIN_ELEMENT; j++) {
                for (k = 0; k < SEG_PROTEIN_SIZE;
                        k++) {
                                DisArray[i+j][k] =
            DisArray[k][i+j] = MAX_DISTANCE;
                                }
                        }
                        }

        for (i = 0; i < SEG_ELEMENT_SIZE; i++) {

        DNA[seg_index][Vector[i]/PROTEIN_ELEMEN
        T].index = Vector[i]%PROTEIN_ELEMENT;
                        }

        for (i = 0; i < SEG_ELEMENT_SIZE; i++) {
                printf(" %d", Vector[i]);
                        }
                printf("\n");

        for (i = 0; i < SEG_ELEMENT_SIZE; i++) {
                        printf(" (%2d,%2d)", i,
            DNA[seg_index][i].index);
                        }
                printf("\n");
                        }

                void fuzzy(void)
                        {
```

```
double      X[SEG_SIZE][30][CHAR_SIZE] =
                {0};
double      U[SEG_SIZE][30][CHAR_SIZE] =
                {0};
double      preV[SEG_SIZE][CHAR_SIZE] =
                {0};

            int         m = 2;
          int           c = SEG_SIZE;
        int         n = SEG_ELEMENT_SIZE *
          PROTEIN_ELEMENT;
        int         d = CHAR_SIZE;
        double      Epsilon = 0.0001;
          double      delta = 0.0;
        double      dSum = 0.0;
          double      dTemp;

            int         i,j,k;

          printf("Run fuzzy ... \n");

        for (i = 0; i < SEG_SIZE; i++)
                {
      for (j = 0; j < SEG_ELEMENT_SIZE; j++)
                {
          for (k = 0; k < PROTEIN_ELEMENT;
                k++) {

memcpy(&X[i][j*PROTEIN_ELEMENT+k][0],
        &DNA[i][j].char_vector[k][0],
        CHAR_SIZE*sizeof(double));
                }
            }
              }

            #if 1
        for (i = 0; i < SEG_SIZE; i++)
            {
        for (j = 0; j < 30; j++)
                {
    for (k = 0; k < CHAR_SIZE; k++) {
          printf(" %.4f,", X[i][j][k]);
                }
            printf("\n");
        }
          }
          #endif

      for (i = 0; i < c; i++)
            {
        for (j = 0; j < n; j++)
                {
          for (k = 0; k < d; k++)
                {
            U[i][j][k] = 1.0/c;
                }
              }
              }

            #if 1
      for (i = 0; i < c; i++)
            {
        for (j = 0; j < n; j++)
                {
          for (k = 0; k < d; k++)
        {
          printf(" %.6lf", U[i][j][k]);
                }
            printf("\n");
              }
              }
           #endif

            do {
        for (i = 0; i < c; i++)
                {
```

```
        for (k = 0; k < d; k++)
                {
              V[i][k] = 0.0;
              dSum = 0.0;
            for (j = 0; j < n; j++)
                {
                dTemp =
U[i][j][k]*U[i][j][k];
                dSum += dTemp;
              V[i][k] += dTemp *
      X[i][j][k];
                }
            V[i][k] /= dSum;
                }
            }

          #if 1
 for (i = 0; i < c; i++)
            {
    printf(" V[%d] = ", i);
    for (k = 0; k < d; k++)
  {
     printf(" %.6lf", V[i][k]);
              }
     printf("\n");
            }
    #endif
     //
        dTemp = 0.0;
    for (i = 0; i < c; i++)
              {
      for (k = 0; k < d; k++)
              {
       if (V[i][k] - preV[i][k] >=
      Epsilon)
                {
                 i = c;
                 k = d;
                dTemp = 1.0;
              break;
                }
              }
            }

        if (dTemp == 0.0) {
          #if 1
    printf("end of fuzzy algorithm.\n\n");
          #endif
                break;
              }

        memcpy(preV, V, sizeof(V));

          for (j = 0; j < n; j++)
        {
            for (k = 0; k < d; k++)
          {
                dSum = 0.0;
            for (i = 0; i < c; i++)
                {
                dTemp =
1/((X[i][j][k]-V[i][k])*(X[i][j][k]-V[i][k]));
                U[i][j][k] = dTemp;
                dSum += dTemp;
                }
            for (i = 0; i < c; i++)
                U[i][j][k] /= dSum;
        }
          }

          #if 1
      for (i = 0; i < c; i++)
              {
        for (j = 0; j < n; j++)
              {
          for (k = 0; k < d; k++)
```

```
                {
                  printf(" %.6lf", U[i][j][k]);
                        }
                    printf("\n");
                    }
                    }
                printf("\n\n");
                  #endif
                } while (1);


                    }


              void verify(void)
                    {
                  int     i,j,k;
              double       dSumA;
              double       dSumB;
int           group[PROTEIN_ELEMENT];

//struct stDNA       unsegDNA[256] = {0};
                  //struct stDNA
DNA[SEG_SIZE][SEG_ELEMENT_SIZE] =
                    {0};


                  //       Group A
          printf("Classify Group A: \n");
for (i = 0; i < SEG_ELEMENT_SIZE; i++)
                    {
    for (j = 0; j < PROTEIN_ELEMENT; j++)
                      {
                    dSumA = 0;
            for (k = 0; k < CHAR_SIZE; k++)
                      {
                     dSumA +=
pow(DNA[0][i].char_vector[j][k] - V[0][k], 2);
                    }
                    dSumB = 0;
            for (k = 0; k < CHAR_SIZE; k++)
                    {
                    dSumB +=
pow(DNA[0][i].char_vector[j][k] - V[1][k], 2);
                    }

                if (dSumA < dSumB)
                    group[j] = 1;
                  else
                  group[j] = 0;

              printf("[%3d,%2d] A: %.6lf,    B:
  %.6lf : Group %s\n", i, j, dSumA, dSumB,
((dSumA < dSumB) ? "A":"B"));
                    }
      if (group[0] == group[1] && group[0] ==
                  group[2]) {
                  if (group[0] == 1) {
                    printf("\tGroup A\n");
                      } else {
                    printf("\tGroup B\n");
                      }
                    } else {
                  printf("\tUngroup\n");
                  }
                    }


                //       Group B
          printf("Classify Group B: \n");
for (i = 0; i < SEG_ELEMENT_SIZE; i++)
                    {
    for (j = 0; j < PROTEIN_ELEMENT; j++)
                    {
                    dSumA = 0;
            for (k = 0; k < CHAR_SIZE; k++)
                    {
                     dSumA +=
pow(DNA[1][i].char_vector[j][k] - V[0][k], 2);
                  }
```

```
                dSumB = 0;
            for (k = 0; k < CHAR_SIZE; k++)
                {
                dSumB +=
pow(DNA[1][i].char_vector[j][k] - V[1][k], 2);
                }
            if (dSumA < dSumB)
                group[j] = 1;
              else
                group[j] = 0;

          printf("[%3d,%2d] A: %.6lf,    B:
%.6lf : Group %s\n", i, j, dSumA, dSumB,
((dSumA < dSumB) ? "A":"B"));
                }
        if (group[0] == group[1] && group[0] ==
group[2]) {
            if (group[0] == 1) {
              printf("\tGroup A\n");
                } else {
              printf("\tGroup B\n");
                }
            } else {
          printf("\tUngroup\n");
                }
            }

        //      ungroup
      printf("Classify the rest: \n");
        for (i = 0; i < 256; i++)
                {
    if (unsegDNA[i].dna_seq[0] == 0) {
                break;
                }

    for (j = 0; j < PROTEIN_ELEMENT; j++)
                {
                dSumA = 0;
            for (k = 0; k < CHAR_SIZE; k++)
                {
                dSumA +=
pow(unsegDNA[i].char_vector[j][k] - V[0][k], 2);
                }
                dSumB = 0;
            for (k = 0; k < CHAR_SIZE; k++)
                {
                dSumB +=
pow(unsegDNA[i].char_vector[j][k] - V[1][k], 2);
                }
            if (dSumA < dSumB)
                group[j] = 1;
              else
                group[j] = 0;

          printf("[%3d,%2d] A: %.6lf,    B:
%.6lf : Group %s\n", i, j, dSumA, dSumB,
((dSumA < dSumB) ? "A":"B"));
                }
        if (group[0] == group[1] && group[0] ==
group[2]) {
            if (group[0] == 1) {
              printf("\tGroup A\n");
                } else {
              printf("\tGroup B\n");
                }
            } else {
          printf("\tUngroup\n");
            }
                }
                }
```

Dna.h 程式碼

```
#define     SEG_SIZE            2
#define     SEG_ELEMENT_SIZE    10
```

```
#define        MAX_BUFFER_SIZE        4096
#define        MAX_LIST_SIZE          32768
#define        MAX_AA_SIZE            10922
 #define       PROTEIN_ELEMENT        3
 #define       CHAR_SIZE              22

  #define      SEG_SIZE               2
#define        SEG_ELEMENT_SIZE       10
  #define      SEG_PROTEIN_SIZE 30

#define        MAX_DISTANCE           22



                struct stDNA {
          char  dna_seq[MAX_LIST_SIZE];
                int
aa_seq[PROTEIN_ELEMENT][MAX_AA_SIZE
                ];
                double
      char_vector[PROTEIN_ELEMENT][22];
                int     index;
                };

          static char PAT[64][4] = {
                "ttt","ttc","tta","ttg",
               "tct","tcc","tca","tcg",
               "tat","tac","taa","tag",
               "tgt","tgc","tga","tgg",
               "ctt","ctc","cta","ctg",
              "cct","ccc","cca","ccg",
              "cat","cac","caa","cag",
             "cgt","cgc","cga","cgg",
              "att","atc","ata","atg",
              "act","acc","aca","acg",
              "aat","aac","aaa","aag",
             "agt","agc","aga","agg",
              "gtt","gtc","gta","gtg",
             "gct","gcc","gca","gcg",
                "gat","gac","gaa","gag",
                "ggt","ggc","gga","ggg"
                };

      static int CHARACTERISTIC[64] = {
                4,4,9,9,
              15,15,15,15,
              19,19,20,20,
               1,1,20,18,
               9,9,9,9,
              12,12,12,12,
               6,6,13,13,
              14,14,14,14,
               7,7,7,10,
              16,16,16,16,
              11,11,8,8,
              15,15,14,14,
              17,17,17,17,
               0,0,0,0,
               2,2,3,3,
               5,5,5,5
               };

      static char CHAR_VECTOR[22] = {
              'A','C','D','E','F','G','H','I','K','L',
              'M','N','P','Q','R','S','T','V','W','Y',
                      '*','+'
               };

      void ReadCharacteristic(char *filename, int
               seg_index);
      void fuzzy(void);
      void verify(void);
```

Main.c 程式碼

```
#include "dna.h"
int main(int argc, char **argv)
{
```

```
//      讀取 A 類的特徵值
ReadCharacteristic("segA", 0);

//      讀取 B 類的特徵值
ReadCharacteristic("segB", 1);


ReadCharacteristic("unseg",
SEG_ELEMENT_SIZE);

//      決定 A 類中，代表 DNA 序列的胺基酸序
                列
//      ObtainAAIndex(0);

//      ObtainAAIndex(1);
```

```
//fuzzy
fuzzy();

verify();


return 0;
}

/*
Art: 37          sag
Net: 71:     ccn
     101:        anc
     105:        gas
     131:tng
*/
```