階梯式 DNA 序列搜尋演算法

朱家漢、白敦文
國立台灣海洋大學資訊工程學系
Email：twp@mail.ntou.edu.tw

張大慈、戴琇苓
國立清華大學生命科學系
Email： lscmdt@life.nthu.edu.tw

摘要

本論文提出一個可以供多重 DNA 序列進行共同區段之比對演算法，應用數值編碼之唯一性及階梯步進式或區間跳躍式比對之法則，增進共同區段之搜尋速度。演算法共分為三個步驟：編碼、排序與搜尋，編碼階段負責將 DNA 鹼基轉換至數值空間集合，排序階段採快速排序演算法完成排序相關動作，而搜尋階段則依數列大小進行步進式比對或依比對樣本長度進行數值區間最佳之均勻切割或位元切割，以增加區間跳躍之機率並提升比對之速度。實驗結果證明本演算法可以將傳統比對所需之時間複雜度由 $O(mL_i(L_i-m+1)+mL_j(L_j-m+1))$ 降低至 $O(|I^i|+|I^j|)$。

關鍵詞：雜湊編碼（Hash coding）、階梯式步進法(Ladderlike Stepping)、階梯式區間跳躍法(Ladderlike Interval Jumping)。

一、背景動機

值此後基因體時代，許多物種的基因體 DNA 序列已經完成解碼及功能分析，後續的研究重點則為基因註解(gene annotation)及功能分析[1]，才能瞭解染色體上的基因區段及調控區段的特性、結構與功能[2][3]。一般而言，染色體上同源的蛋白質家族的基因序列彼此具有相當高的相似度，其轉譯而成的蛋白質之胺基酸序列也會呈現高度相似性。如此的特徵藉由現有的各式基因資料庫或蛋白質資料庫都能迅速找到比對結果[4][5]。但是染色體上仍有廣大的區域包含未知功能的基因及其他區段，則必須運用新的生物資訊工具進行分析[6][7][8]。

染色體上常含有特殊的短串 DNA 序列提供相對應之調控蛋白結合之用，這種 DNA 與蛋白質結合的交互作用能影響基因表現或蛋白質的功能。舉例而言，真核生物的基因上游為啟動子區域(promoter region)[9][10][11]，須提供核酸聚合酶(RNA Polymerase)及調控蛋白結合的部位，稱之為轉錄子(transcription element)，對應之蛋白稱為轉錄因子(transcription factor)。雖然已知的短串 DNA 序列特徵能由部份程式識別[12][13][14][15]，但是通常為一對多的方式比對獲得，即輸入一段 DNA 字串至資料庫中搜尋已知特徵的片段，但是對於未知的片段則不易鑑定。因此我們發展出一套程式，針對某一特定之 DNA 家族序列，利用區間式跳躍的方法快速比對搜尋具相關功能的基因家族，並尋找出啟動子區域內具特殊保留性的短串 DNA 序列，以提供新穎的解決方法。

搜尋演算法的優劣通常以比對過程所需的時間複雜度為評比依據，假設某一 DNA 序列的長度為 n ,而搜尋樣本的字串長度為 m ，以傳統字串比對搜尋演算法在該序列中找尋樣本字串所需花費的時間複雜度為 $O(mn)$ ，若假設該長度為 m 的樣本字串是由另一條長度為 n 的序列中所有可能子序列的組合，即有 $n-m+1$ 個可能樣本字串需要進行比對，且進行比對所需花費的時間複雜度劇增為 $O(mn\times(n-m+1))$。倘若有 K 串具相同長度的序列進行傳統字串比對，則所需花費的時間複雜度為 $O(mn\times(n-m+1)\times K)$ ，才能從 K 串序列中找出長度為 m 的共同子序列，但進行比對所耗費之時間著時可觀。因此在樣本比對這個知識領域中，一般常用在字串比對方法,除了傳統的比對演算法外，還有許多已被提出來降低時間複雜度的演算法，如：Karp-Rabin 演算法(KR-A)[16][17]、Knuth-Morris-Pratt 演算法(KMP-A)[17][18][19]、Boyer-Moore 演算法(BM-A)[17][20][21]、快速搜尋演算法(QS-A)[17] [22]等演算法，於 Karp-Rabin 的演算法中是採用雜湊函式將字串編碼到數字領域的方法,其前置處理的時間複雜度為 $O(m)$ ，在搜尋階段中最差情況的時間複雜度為 $O(mn)$ ，亦即在一序列中找尋某一樣本字串的情形，而平均期望的時間複雜度為 $O(m+n)$，以平均期望時間複雜度來衡量比較兩相同長度序列之時間複雜度則為 $O((n-m+1)\times(m+n))$，相同地，若共有 K 串相同長度序列進行比對時，時間複雜度為 $O((n-m+1)\times(m+n)\times K)$。經由詳細的比較，其結果分析如表(一)，其中 PPTC 是表示前置處理的時間複雜度(Preprocessing Phase Time Complexity)、SPTC 是指搜尋階段之時間複雜度(Searching Phase Time Complexity)、ECTC(BCTC)是指期望或最佳之時間複雜度

(Expected Case Time Complexity 或 Best Case Time Complexity)，在 SPTC 欄位中所表示的均是在比對過程中最差情況之時間複雜度，其中 m 是指比對樣本的長度，n 是指序列的長度，而 σ 則表示是在 BM 演算法中的劣字元(bad character)表格的大小。由 Karp 與 Rabin 所提出的 KR 演算法是將字串進行編碼，轉換字串到數字集合後再來執行比對以降低字串比對的次數，而 Knuth-Morris-Pratt 演算法和 Boyer-Moore 演算法則提供了採用區段比對的概念，若能夠將這些特性加以整合運用，則可減少搜尋重複相同 DNA 字串之時間複雜度。

表(一)：各種演算法時間複雜度之比較

	PPTC	***SPTC***	**ECTC(BCTC)**
BF-A	-	O(mn)	O(mn)
KR-A	O(m)	O(mn)	O(m+n)
KMP-A	O(m)	O(n+m)	-
BM-A	O(m+σ)	O(mn)	O(n / m)

二、符號定義

本論文主要之目標是希望從一個 DNA 家族序列中尋找長度不定的共同字串，DNA 序列是由字母表 $\Sigma = \{A,C,G,T\}$ 所構成之字串，其中 A,C,G,T 為 DNA 序列中的四個鹼基。以 S 代表 DNA 家族序列之集合，在 S 集合中若一個 DNA 序列以 y 來表示，$y \in S$，其中 $|y|$ 代表該序列的長度，且該序列的索引值由 0 到 $|y|-1$，索引值由 i 到 j 的子序列是以 $y[i...j]$ 表示，且 $0 \le i \le j \le |y|-1$。若 ψ^y 集合是由序列 y 來組成所有可能子序列區段之集合，其中每一個不同長度的子序列區段即為比對樣本之候選字串，而 ψ_m^y 則為 ψ^y 中樣本長度為 m 的子集合，若某一樣本以 w 符號代表，則 $|w|$ 代表該樣本的長度，索引值是由 0 到 $|w|-1$。假設在 S 集合中，序列之總數為 N，其中第 i 個序列以 S^i 表示，該序列之長度為 $|S^i| = L_i$。在本論文所討論之演算法中，字串將先進行編碼轉換之過程，並形成一個以數值代表的新集合，以 $\tilde{\psi}^y$ 來表示之，對子序列 $\tilde{\psi}^y$ 集合進行編碼後之數值，而 $\tilde{\psi}_m^y$ 為數值集合 $\tilde{\psi}^y$ 中樣本長度為 m 的子集合，$\tilde{\psi}^y$ 集合中依其數值大小進行排序及分組，讓具有某些共同範圍之數值可以群聚在某一特別區間，並以 I 符號來代表編碼數值區間之集合，其中對 y 序列編碼後之數值區間集合以 I^y 表示，則 $|I^y|$ 為第 y 個已編碼序列區間集合的個數，${}^A I^y$ 代表對 y 序列之編碼數值中數字 A 所在之區間位置，即該編碼數值之區間索引值。

三、演算法說明

本論文針對比對多重 DNA 序列之共同子字串提出兩種演算法，第一種為階梯式步進比對演算法(Ladderlike Stepping Searching Algorithm ,LSSA)，第二種為階梯式區間跳躍比對演算法(Ladderlike Interval Jumping Searching Algorithm, LIJSA)，兩種演算法皆包含三個階段：編碼、排序和搜尋階段。第一階段和第二階段在兩種演算法之中皆執行相同的程序，僅有在第三個階段中，階梯式區間跳躍演算法提出了更進階的演算模組及更快的比對速度。第一階段編碼的精神在於字串序列轉換成數字序列時，經由編碼後的數值具有唯一性，所以能夠降低原本逐一字元符號比對的時間複雜度，第二個階段是執行排序之動作，主要的目的是為了在數值比對時可以依數值之大小循序比對，可避免未經排序數列之重複搜尋，而第三個階段是執行及記錄最後比對的過程，第一種步進式比對沒有額外的區間切割程序，而第二種演算法將同一區間之編碼數值在排序之後集中放置，以不同方式進行區間分割，期能將不同序列在同一區間之編碼數值數量差距加大，增加區間跳躍之機率並減少序列比對所需之時間，以下分別說明這三個階段的執行方式與改進效率。

(一)編碼階段：在此採用的編碼方式和 Karp-Rabin 演算法[23]相似，不同點在於本演算法所取的換算基底不是 ASCII 編碼數字而是以 4 為基底，其中 G、A、T、C 分別以 0、1、2、3 來取代。至於以 4 為編碼基底的原因在於將字串轉變成數字時具有'唯一'之特性，可避免在 Karp-Rabin 演算法中所取的編碼基底小於 ASCII 數字，造成編碼後產生衝突的情況發生，運用公式(1)及(2)取得每一 DNA 序列區段中樣本長度為 m 的'唯一'雜湊值，其中 w 是長度為 m 的樣本字串，$base$ 為編碼基底，而 y 為進行編碼之 DNA 序列。編碼過程的時間複雜度為 $O(N \times (L_{\max} - m + 1))$，其中 N 為家族序列集合的個數，$L_{\max}$ 為集合中最長的序列長度，m 為某一樣本的長度，其中 $2 \le m \le L_{\max}$。

$$hashfunc(w[0..m-1]) = w[0]base^{m-1} + w[1]base^{m-2} + ... + w[m-1]base^0 \quad (1)$$

$$rehashfunc \quad (y[i], y[i+m], hash([i...i+m-1])) \quad (2)$$

(二)排序階段：長度為$L_{\max}$之DNA序列依固定長度由頭至尾依序取m個鹼基進行區段編碼後，將產生$L_{\max}-m+1$個數值，在此採取由小至大之快速搜尋排序法進行編碼數值排序[19][24]，採用快速排序演算法(Quick Sort)所花費之時間複雜度為$O(n\log n)$，其中$n=L_{\max}-m+1$。

(三)搜尋階段：階梯式步進式比對演算法和階梯式區間跳躍比對演算法皆建議採用兩兩序列的漸進比較法則，在任一個比對過程中都是由兩組序列經編碼及排序後之集合進行比對，且兩集合的比對過程如同下階梯一般，前者不需要經由區間切割之過程，可將每一個經編碼之數值視為一區間，其搜尋比對採一次單階步進之方式，後者則是經過進一步的區間切割，可一次單階或多階的躍進，這兩種演算法比較說明將於下列章節描述。階梯式步進演算法排序階段後的搜尋比對，是不再進行任何特定區間分割，因此在任意i,j兩序列的比對過程中就像是在階梯中行進一般，每一個經排序之編碼數值如同一個階梯，一階一階地往下比對，故只須選取兩序列從頭至尾比對一次即可將相同字串全數找出，所花費之搜尋比對時間複雜度為$O(L_i+L_j)$，$1\le i,j\le N$且$i\neq j$，L_i與L_j分別為序列i與j之長度值。在開始的比對階段，本演算法可任意挑選出兩序列進行比對，若以尋找長度m的共同子序列，在比對這兩組經編碼之數值集合之後，相同的數值將形成一個新的序列集合，以$\tilde{\psi}_m^{i,j}$，該集合中的數值經解碼後的內容即為這兩條比對序列所共同存在的字串，亦即$\psi_m^{i,j}$表示，若是$m>1$，則$\psi_m^{i,j}$集合中的數量必然小於較短的DNA序列長度，$|\psi^{i,j}|\le \min(L_i,L_j)$，$1\le i,j\le N$且$i\neq j$，接下來的搜尋比對即以此新的編碼數值集合$\tilde{\psi}_m^{i,j}$與另一編碼過的數值序列$\tilde{\psi}_m^{k}$繼續進行相同的比對，由於愈往後面的比對階梯數減少，相等數值的機率也愈來愈少，亦即共同字串的可能性降低，因此愈後面搜尋比對的時間複雜度一定具備逐漸下降的趨勢，。在多重DNA家族序列集合中，搜尋比對之最佳與最差的狀況分析分別如下所述，最佳狀況是在第一次比對中，對搜尋某一樣本長度m之共同子區段，在兩序列當中已無相同的編碼數值，即$|\psi_m^{i,j}|=0$，故所花費之時間複雜度僅為$O(L_i+L_j)$，最差的情形在於每次比對後相同的編碼數值即為較短一條序列的所有編碼數值，亦即$|\psi_m^{i,j}|=\min(L_i,L_j)$，時間複雜度為$O(L_i+L_j+\min(L_i,L_j)+L_k+\min(\min(L_I,L_J),L_k)+L_m+...)$，這種情形發生在每條序列都是其它序列的子區段，$S^i\subset S^j\subset S^k$，這種特別的情形其時間複雜度可以趨近於$O(N\times\min(L_1,L_2,L_3,...))$，即使在最差的情況下仍然低於傳統比對方式的時間複雜度。本演算法除了可以快速找出所有指定序列所共同擁有的字串外，並可以同時得到該共同字串之出現次數與出現在序列中的位置，比對結果可迅速提供生物學家進一步的迅速分析比較同一家族序列之共同區段與特殊功能之因子。

(四)階梯式進步之改進方式：在上述搜尋階段中所強調的是任意選擇兩個序列進行進步式的比對，但為了進一步降低比對的時間複雜度，提出更進階的改善方法，主要的精神在於兩個 DNA 序列經由編碼後的數值比對過程中，期望能群聚編碼數值範圍相近的數字，使其具備區間跳躍的環境，快速地跳過比原本編碼數值還要小的項目，直接比對與其相等或者較大的數值區間，若是可以略過的數值區間愈多，比對的速度就愈快，如同下階梯一般，比對時可以一階一階行進，甚至有時可以一次跳躍數個階梯，故稱之為階梯式區間跳躍法則。本論文所提之階梯式區間跳躍法則主要分為兩種區間切割法來定義階梯之實際範圍，第一種稱為位元切割法，第二種稱為均勻分佈切割法。位元切割法之優點在於能夠快速地計算出編碼數值應屬於哪一個區間，該切割法是根據比對樣本之長度(m)來進行區間切割，其公式如下，

$$|I_m|=\lceil \log_{base} 4^m \rceil \qquad (3)$$

其中$base$之值設定為10，由公式(3)可以得知分割區間個數為取4的倍數後，再取上限之整數值，第一區間為單位元區間，以此類推為二位元區間、三位元區間等，其中位元的相對意義即為該數字的長度，例如，0~9的數值為第一區間、10~99為第二區間，以此類推，因為此位元切割法是在編碼階段即可完成，並不會額外佔用其它計算之需求，接下來的搜尋比對階段當中，仍然依照先前的比對規則，只是在比對兩序列數值前根據區間跳躍規則(Jump Rule，JR)進行比對，其JR規則如下之判斷式，

Case1: ${}^A I_m^1 \neq {}^B I_m^2$ and $A>B$ (4)

➔B *jump to* ${}^A I_m^2$

Case2：${}^A I_m^1 \neq {}^B I_m^2$ and $A<B$

➔ A *jump to* ${}^B I_m^1$

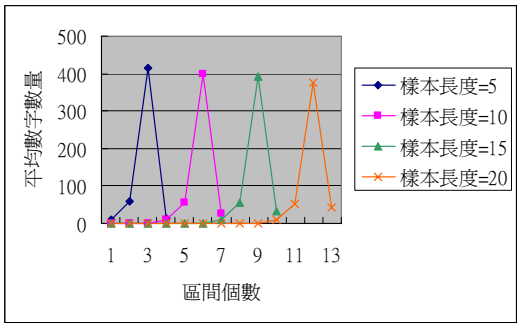
Case3：${}^A I_m^1 = {}^B I_m^2$ and $A\neq B$

➔*Step by Step in LSSA*

Case4： ${}^{A}I_m^1 = {}^{B}I_m^2$ and $A=B$

➔*Check if A and B repeat continuously in interval* I_m^1 *and* I_m^2

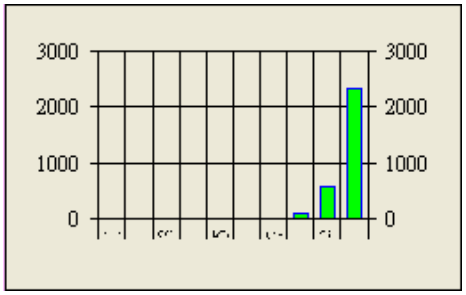
其中 ${}^{A}I_m^1$ 表示第一個已編碼數值序列中數字 A 所在之區間位置，${}^{B}I_m^2$ 為第二個已編碼數值序列中數字 B 所在之區間位置，如果比對的兩個數字不是位在同一區間，則停留在數字較小的數值序列需要跳躍至較大數字所在之區間，接著才繼續以 LSSA 進行比對的工作，若兩個比對數字落在同一個區間內，則直接進行 LSSA 比對之程序，若是序列的子區段內容不同，則編碼數值就不會相同，若是分配至不同的切割區間內，很明顯地，跳躍規則將可用來進行區間之跳躍，必能降低在搜尋比對階段之時間複雜度。在執行位元區間分割法之後，經由理論分析與統計實驗結果，各區間內的數值個數如圖(一)所示，大部份的字串經編碼後其數值仍分佈在分割區域中的最後兩個區間內，因此，最主要的切割區應強調在最後的兩區間中，均勻分佈切割法之應用也因此在本論文中提出。首先須了解切割區間之原則以及區間切割的總數，才可以決定所有字串經編碼後數值的分組方式。



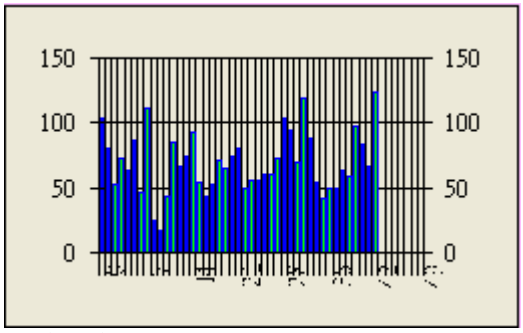
圖(一)：RNase 家族序列經位元分割後平均區間數字數量分佈在最後兩區間的情形

本演算法對目前的切割區域分類為兩種情形，第一種類別為一般切割區(General Segmentation Region)及第二種類別為均勻分佈切割區(Uniform Distributed Segmentation Region)。一般切割區視為非集中區間，也就是除了最後兩區間以外之集合，以 I_{m_G} 表示樣本長度為 m 之非集中區間，因為數量所佔比例低，所以不再進行任何區間切割，一般切割區之區間總數為 $|I_{m\_G}|=1$。均勻分佈切割區存在大部份的編碼數值，演算法對此集中之區間進行切割時，首先必須先了解區間與區間之間隔，可採用統計直方等化法進行分析，其結果如圖(二 a)與(圖二 b)所示，切割後讓每一個區間內的數值具體地分佈分散，以因此提昇了區間跳躍之機率。原始區間總數為 $\lceil \log_{base} 4^m \rceil$，其中 *base* 之數值為 10，且採用 $|I_{m\_original}|$ 之符號代表。在未經均勻分佈切割時，演算法直接計算其切割基底，計算方式為 $10^{|I_{m\_original}|-2}$，並以 *sbase* 表示。例如取樣本長度為 5 的比對字串，原始區間之總數為 $\lceil \log_{10} 4^5 \rceil$=4，$sbase = 10^{4-2} = 100$，倒數第二個區間數 $|I_{m\_L2}| = 9$，最後一個區間數 $|I_{m\_L1}| = (4^m - 10^{|I_{m\_original}|-1}) / (10^{|I_{m\_original}|-2})$，故得總區間數 $|I_{m\_total}| = |I_{m\_G}| + |I_{m\_L1}| + |I_{m\_L2}|$。

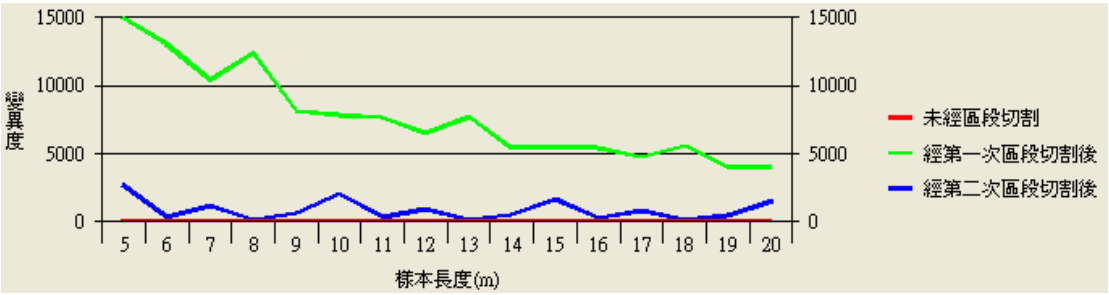
此均勻切割法亦是在編碼階段完成，所以不會造成額外的時間複雜度，同樣地，亦能透過公式 $\lfloor A/sbase \rfloor$ 的計算即可立刻求得該數值 A 所屬之區間位置，和位元分割法一樣在搜尋比對階段中依照 JR 執行序列比對。若是相對應較大數字所在區間並無任何元素存在，則指標依序往下一區間跳躍，直至有元素存在的區間才停止跳躍。無論是位元切割法或者是均勻切割法之最終目的是為降低演算法之時間複雜度，希望能夠避免演算法對每一筆數值資料進行比對的缺點，進而提供以區間跳躍之方式來加快搜尋比對速度，期望時間複雜度由原本兩序列傳統比對之時間複雜度 $O(L_i+L_j)$ 下降至 $O(|I^i|+|I^j|)$。



圖(二 a) RNase 序列切割前分佈情況



圖(二 b)RNase 序列經均勻切割後分佈情況



圖(三):RNase 家族序列經位元切割、均勻切割與未經切割時之平均變異度比較圖，其中欲比對之樣本長度為 5~20 個鹼基

四、結果與討論

在上述章節中所討論之階梯式共同子序列搜尋演算法中，進步式比對演算法 LSSA 經由表(二)中與 BF-A(傳統字串比對演算法)、KR-A(Karp-Rabin 演算法)比較過後，得知 LSSA 演算法有較快的搜尋比對速度，由 LSSA 中可看出編碼階段、排序階段、搜尋階段之時間複雜度分別為 $O((L_{\max}-m)^2\times N)$ 、 $O(N\times(L_{\max}-m)\log(L_{\max}-m))$ 、 $O(N\times(L_{\max}-m))$，未經區間切割部份與 KR-A 之時間複雜度相對比值為 $O(\log(L_{\max}-m)/L_{\max})$，又 KR-A 與傳統演算法時間複雜度比值為 $O((L_{\max}-m)/(L_{\max}m))$，故可得知未經區間切割之演算法所花費之時間複雜度比傳統演算法和 KR-A 更低，最後透過區間切割法達到期望時間複雜度 $O(|I^i|+|I^j|)$，$1\le i\le j\le N$ 且 $i\ne j$，但是位元切割法和均勻切割法之期望時間複雜度均為相同($O(|I^i|+|I^j|)$)，為了更詳細的比較兩種區間切割方法的比對速度，本論文提出在編碼數值序列進行分段後，對每一區間所存在之編碼資料數量計算其平均變異度，期望經編碼之資料可以透過適當之區間分組，使所有序列在各區間之平均變異度能夠差距越大越好，亦即將比對序列在同一個區間之數量差異性提高時，可以使階梯式比對法則進行快速跳躍的機率增高，所以序列之間每一區間的平均變異度相差越大，就越有可能降低演算法之時間複雜度，平均變異度之比較可以由圖(三)所示，無論是位元切割法或者是均勻切割法均比原先未經區間切割時之變異度還要高，故可得知經區間切割後再進行搜尋比對時，能進一步提昇演算法之整體表現。

經位元切割法後之序列平均變異度有隨著樣本長度上升而下降之趨勢，原因為隨著樣本長度的增加，經編碼過後的數值越大，而且大部份的數值會落在最後幾個區間當中，造成序列在同一區間的變異度下降。經均勻切割法後，序列平均變異度亦隨樣本長度上升而下降，由於切割之區間範圍較小，隨著樣本長度增長編碼數值變大，落在區間中經編碼的數量則不會突然遞增或減少，所以變異度的差異性並無太大的變動，無論樣本長度大小為何，透過均勻切割法後均保持週期性的脈動，故有可能當樣本長度繼續增長的情形下，均勻切割法會比位元切割法保持較大之變異度，搜尋比對速度也會有較佳的表現。所以本演算法建議樣本長度較小時使用位元切割法，反之，當樣本長度較長時使用均勻切割法。

五、參考文獻

[1]Lincoln Stein,2001,Genome annotation : from sequence to biology *Nature reviews genetics*, 2:493 – 503

[2] Jacek Majewski and Jurg Ott, 2002, Distribution and characterization of regulatory elements in the human genome, *Genome research*, 12:1827~1836

[3] Len A. Pennacchio and Edward M. Rubin, 2001, Genomic strategies to identify mammalian regulatory sequences, *Nature reviews genetics*, Vol. 2, 100~109

[4] http://www.genome.ad.jp/ , Aug. 14 2003

[5] http://www-btls.jst.go.jp , Aug. 14 2003

[6] Yutaka Suzuki, Tatsuhiko Tsunoda, Jun Sese, Hirotoshi Taira *et.al*.,2001, Identification and characterization of the potential promoter regions of 1031 kinds of human genes, *Genome research*,11:677–684

[7] Gabriela G. Loots, Ivan Ovcharenko, Lior Pachter, Inna Dubchak, and Edward M. Rubin, 2002, rVista for comparative sequence-based discovery of functional transcription factor binding sites, *Genome research*, 12:832–839

[8]Ramana V. Davuluri, Ivo Grosse, Michael Q. Zhang ,2001, Computational identification of promoters and first exons in the human genome, *Nature Genetics,* 29: 412 - 417

[9]Nathan D. Trinklein, Shelley J. Force Aldred, Alok J. Saldanha,and Richard M. Myers, 2003, Identification and functional analysis of human transcriptional promoters, *Genome research*, 13:308~312

[10]Jacek Majewski and Jurg Ott, 2002, Distribution and characterization of regulatory elements in the human genome, *Genome research*, 12:1827~1836

[11]Yitzhak Pilpel, Priya Sudarsanam, George M. Church, 2001, Identifying regulatory networks

by combinatorial analysis of promoter elements, *Nature Genetics* 29:153~159
[12]Buhler,J.,2001,Effective large-scale sequence comparison by locality-sensitive hashing.*Bioinformatics*, 17 ,419-428.
[13]Ning Z,Cox AJ, Mullikin JC., October 2001,vol. 11,SSAHA:a fast search method for large DNA databases, *Genome Research*,11,1725-1729.
[14]Natalia Volfovsky,Brian J Haas and Steven L Salberg, 2001,A clustering method for repeat analysis in DNA sequences,*Genome Biology*.
[15]Pierre Baldi and Pierre-François Baisnée, 2000, Sequence analysis by additive scales:DNA structure for sequences and repeats of all lengths, *Bioinformatics*, 16, 865-889.
[16]KARP R.M.**,** RABIN M.O., 1987, Efficient randomized pattern-matching algorithms. *IBM J. Res. Dev.* 31(2):249-260.
[17]CROCHEMORE, M., LECROQ, T., 1996, Pattern matching and text compression algorithms, in *CRC Computer Science and Engineering Handbook*, A. Tucker ed., Chapter 8, pp 162-202, CRC Press Inc., Boca Raton, FL.
[18]KNUTH D.E.**,** MORRIS (Jr) J.H.**,** PRATT V.R., 1977, Fast pattern matching in strings, *SIAM Journal on Computing* 6(1):323-350.
[19]Knuth,D.E., 1998, vol. 3 of *The Art of Computer Programming,* 2nd edn, Addison Wesley.
[20]BOYER R.S.**,** MOORE J.S., 1977, A fast string searching algorithm. *Communications of the ACM*. 20:762-772.
[21]COLE, R., 1994, Tight bounds on the complexity of the Boyer-Moore pattern matching algorithm, *SIAM Journal on Computing* 23(5):1075-1091.
[22]SUNDAY D.M., 1990, A very fast substring search algorithm, Communications of the ACM . 33(8):132-142.
[23]Ricardo Baeza-Yates, Gaston H. Gonnet, October 1992, vol. 35,A new approach to text searching, *Communications of the ACM*, 10, 74-82.
[24]Hoare, C. A. R. (1962), Quicksort, *The Computer Journal* 5, 10-15.

表(二):BF-A、KR-A、LSSA 與 IJSA(未經區間切割、經位元區間切割和經均勻區間切割)時間複雜度比較表

		時間複雜度		
BF-A		$O((L_{\max}-m)\times L_{\max}\times m\times N)$		
		編碼階段	排序階段	搜尋階段
KR-A		$O((L_{\max}-m)^2\times N)$	-	$O((L_{\max}-m)^2\times N)$
LSSA	未經區間切割	$O((L_{\max}-m)^2\times N)$	$O(N\times(L_{\max}-m)\log(L_{\max}-m))$	$O(N\times(L_{\max}-m))$
LIJSA	位元區間切割法	$O((L_{\max}-m)^2\times N)$	$O(N\times(L_{\max}-m)\log(L_{\max}-m))$	$O(N\times(L_{\max}-m))$ $O(N\times(L_{\max}-m)/\lvert I_{\max}\rvert)$
	均勻切割法	$O((L_{\max}-m)^2\times N)$	$O(N\times(L_{\max}-m)\log(L_{\max}-m))$	$O(N\times(L_{\max}-m))$ $O(N\times(L_{\max}-m)/\lvert I_{\max}\rvert)$
備註	m：樣本長度、N：序列個數、$L_{\max}$：序列集合中擁有最大長度之序列的長度、$\lvert I_{m\_\max}\rvert$：序列集合中擁有最大長度之序列的區間切割數量			