博雅書院期末報告—

高效能生物資料去冗餘演算法

高效能生物資料去冗餘演算法開發小組

指導教授：羅惟正、梁美智 老師

小組組長：生物科技學系 林千珊

小組組員：生物科技學系 翁詩玟

生物科技學系 楊鈞詒

資訊工程學系 周暘典

電機工程學系 蘇啓宏

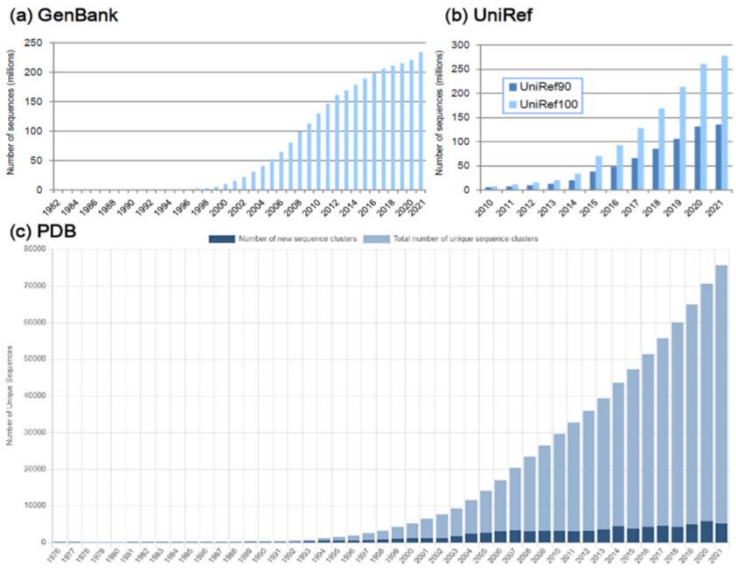
簡介(ABSTRACT)

我們相信，臺灣作為資訊科技巨頭，有實力帶領全球開創數位生物科技新紀元。我們企圖拋磚引玉，以提升大數據分析效率為切入點，開發高效能生物序列資料去冗餘演算法，盼加速全球生物科技發展，並鼓舞更多資訊團隊投入生物科技研發，提升臺灣的國際地位。大數據時代來臨，各類資料庫日漸龐大，如 GenBank, UniRef, PDB 皆以指數級成長。處理海量數據極耗資源與時間，儼然成為不具超級電腦之多數研究團隊進步的阻礙。然而，許多生物資料具高度相似性，全數分析並無必要。當前諸多頂尖生物資料去冗餘演算法仍有缺點；速度快者不夠精確，精確者的運算速度卻追趕不上資料增加的腳步。我們發現，幾乎所有序列減量演算法都將去冗餘及分群共同封裝，但很多研究只需去冗餘之資料而非分群結果。因此，本計畫欲開發一演算法，把傳統同時進行去冗餘和分群的操作做兩階段拆分，並於去冗餘階段用 Golang 程式語言實現高度平行運算，來大幅提升資料生產效率。期望在海量數據充斥的現今，能以臺灣強大資訊實力，提供全球一個快速獲取高密度原始資料並銳減資料總量之有效解決方案，推進各生物科技領域研發，共創新紀元。

研究問題動機（INTRODUCTION）

大數據時代來臨，隨著各項硬體、軟體和人工智慧科技快速增長，含生物科技在內之各項領域的基礎資料量，也都正以爆炸性速度急遽增加，連作為生物科技領域多項研究所需的重要生物巨分子，DNA、RNA、蛋白質也不例外。如圖一中所示：常用之資料庫 GenBank、 UniProt 和 Protein Data Bank (PDB)中生物分子序列數，已成指數型成長多年。

近年來，這急劇增加的基礎資料量，著實令沒有超級電腦的科研團隊難以有效率地推進研究進程，也就是說，這些不同領域的海量資料也極有可能為各相對應領域之研究帶來阻力。



圖一：常使用之序列資料庫的成長趨勢

(a) GenBank 資料庫(由 NCBI 所維護)

(b) UniRef 資料庫(為 UniProt 下設的去冗餘資料集，數字 90 和 100 分別表示任兩條序列資料間彼此相似度< 90% 及 < 100%)

(c) Protein Data Bank(PDB)蛋白質結構資料庫

事實上，由於生物數據通常具有重複性，因此在大部分生物科技領域中，是沒有必要將所有巨量資料一一放下去做分析的。如圖一中所示，雖 PDB 中總序列數呈指數增長，但相似度較低的序列卻成長緩慢。對大部分研究主題來說，這些資料量較小的低相似度序列，就已足夠供給研究所需。一些蛋白質結構預測研究中[1，2]也指出，相較用龐大且重複性高的資料，使用這些低相似度序列作為基礎資料可以協助提升整體的預測準確度。因此，如何有效率地將生物序列資料去冗餘，這個問題已漸漸不容忽視。

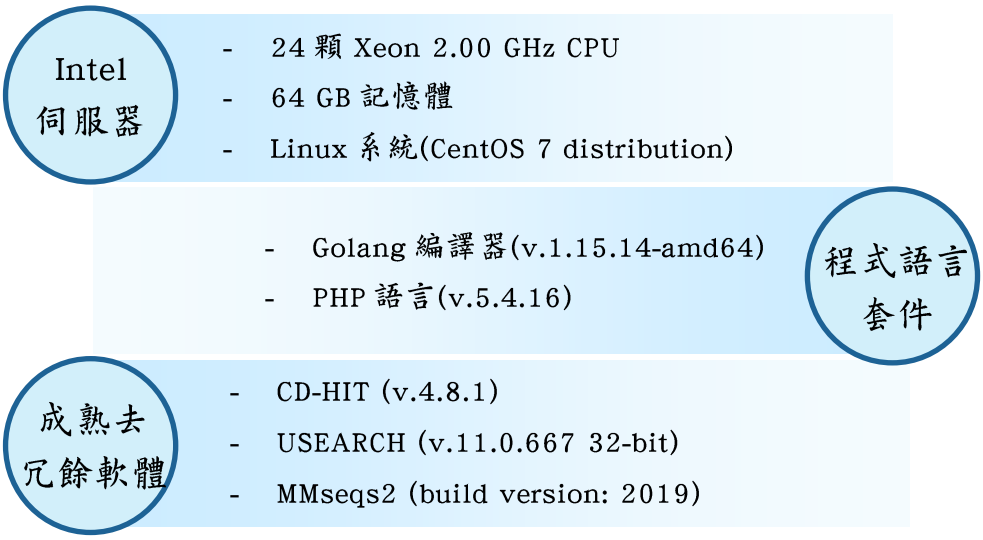
目前現有成熟序列去冗餘演算法仍有不足之處，尤其是速度和準確度無法同時兼顧。在實驗中我們發現，用成熟去冗餘演算法，將 UniProt 資料庫中約一億

條序列，放入配有 24 至 48 顆 2.0 GHz CPU 的伺服器中做計算，篩選出序列彼此間相似度<40%的結果，約耗時 4 個月，而在這段期間中，UniProt 的資料量又增加了約 30%。

如何有效將資料做去重複處理並兼顧速度和準確度，其重要性不言而喻。我們的目標是，開發出高效能生物資料去冗餘演算法，透過去除重複性序列來減低目前海量序列資料，生成精密、非重複的數據集，來提高生物技術的進步。盼我們的拋磚引玉能吸引更多人投入至生物資訊領域中，這樣，臺灣未來或許將能夠在推進各生物科技領域研發的同時，與世界各國建立牢固技術和經濟關係，攜手共創新紀元。

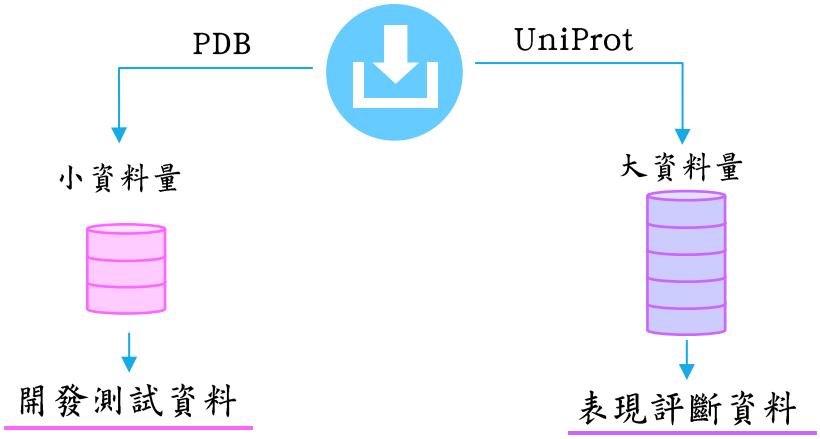
研究環境(ENVIRONMENT)

● 研究設備器材



研究皆於備有 24 顆 Xeon 2.00 GHz CPU 和 64 GB 記憶體的 Intel 伺服器上進行，使用羅老師先前建立之具有 Golang 編譯器(v.1.16.2-amd64)和 PHP 語言套件(v.5.4.16)之 Linux 操作系統(CentOS 7 distribution)進行程式寫作，並下載安裝目前成熟之序列去冗餘演算法軟體 CD-HIT (v.4.8.1) [3]、USEARCH (v.11.0.667 32-bit) [10]、MMseqs2 (build version: 2019)作為研究參考資料。

● 序列資料取得 - 蛋白質序列



下載 2022 年 1 月 3 日由 PDB (https://www.rcsb.org/) 釋出之蛋白質序列資料，資料中包含 2021 年底前存入之所有序列。使用羅老師實驗室已具備的 PHP 程式碼，對 PDB 結構資料檔進行分析，並隨機將其拆分成不同序列大小的子檔案作為演算法開發的測試資料集，如 pdb100(100 條蛋白質序列)、pdb1K(1000 條蛋白質序列)、pdb10K(10000 條蛋白質序列)..... 資料集。

下載 2015 年 12 月 9 日和 2021 年 10 月 27 日由 UniProt (https://www.uniprot.org/) 釋出之蛋白質序列資料。資料中含有三個子資料集，分別為 UniRef100、UniRef90 和 UniRef50，其數字代表了資料集中任兩條序列的最高相似程度，下載之序列資料分別為：

序列數 資料集 / 年分	UniRef100	UniRef90	UniRef50
2015	70.5 million	38.2 million	16.3 million
2021	280.5 million	138.5 million	49.9million

我們用這些巨量資料來分析判斷目前成熟演算法和我們開發之演算法的表現(執行速度、記憶體消耗、磁碟負載)差異。

研究方法(METHOD)

● 選擇欲參考之成熟演算法

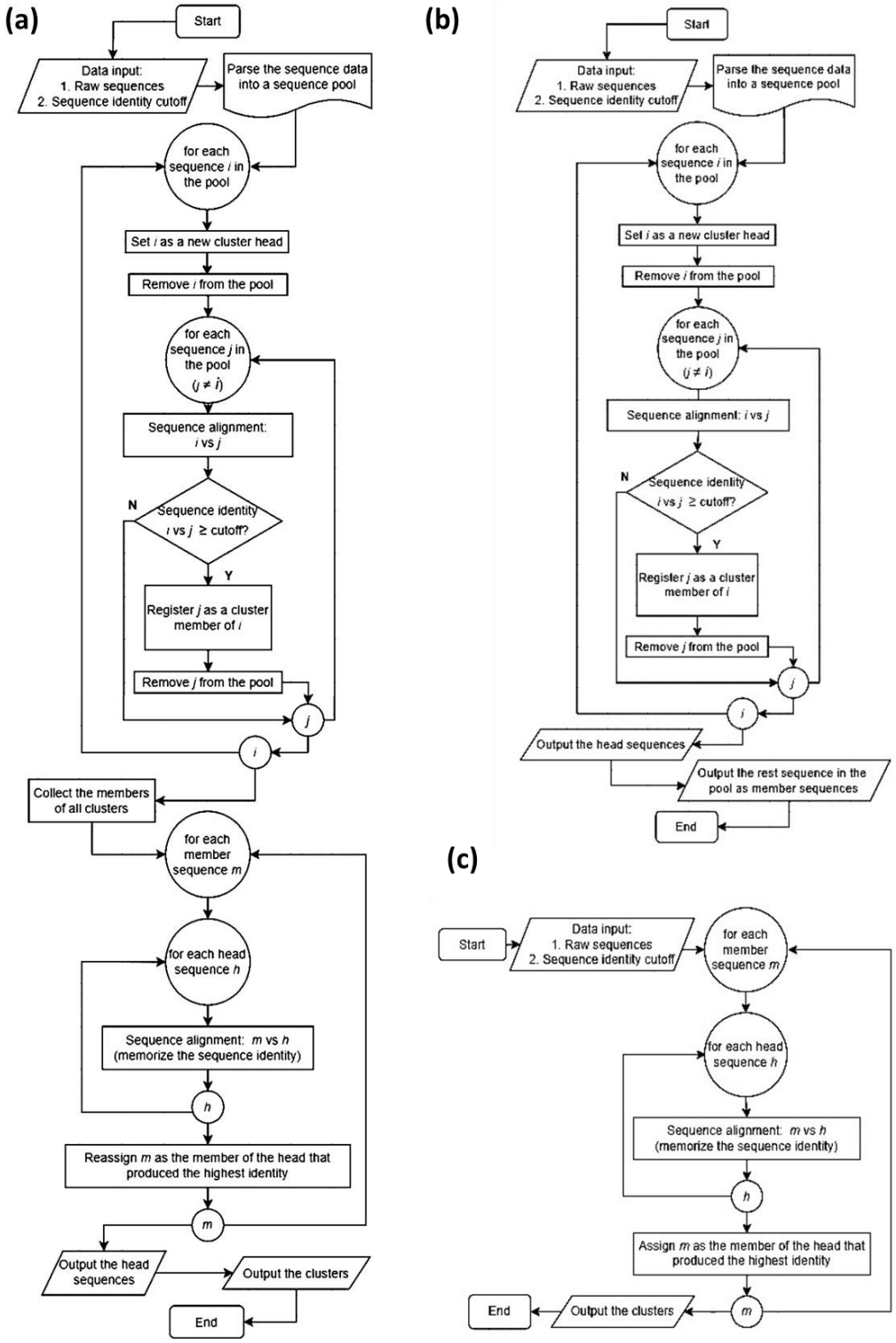
根據羅老師的觀察，這三個成熟演算法(CD-HIT、USEARCH、MMseqs2)中，CD-HIT 擁有最高的準確度及穩定性，而 MMseqs2 則展現了最高的速度。為了證實這個觀察結果，我們分別用 UniRef90-2015 資料集對這三種軟體進行測試，並觀察他們過濾出序列彼此間相似度小於 70%之資料集。如表格 1 中所示，雖然 CD-HIT 速度遠小於 MMseqs2，但其只需跑 1 次，過濾出之序列數就能維持穩定，而 MMseqs2 則需執行 20 次，產生之序列數才能到達穩定。因 32 位元的 USEARCH 有 4GB 的容量上限，無法負荷 UniRef90-2015 (14.3 GB 磁碟空間)這麼多的資料，因此我們隨機抽取 UniRef90-2015 中的資料對其進行測試，發現他最多只能夠處理約 160 萬筆資料，而且需執行 10 次其序列數才能漸趨穩定。

鑒於上述觀察結果，我們決定選擇 CD-HIT 來當作同源序列去冗餘演算法開發的主要基礎及參考。

Method	Dataset	Number of sequences	Time cost (minutes)
CD-HIT	Initial set (UniRef90-2015)	38,203,400	
	Round 1 nr set	21,270,890	1690.6 (20 threads)
	Round 2 nr set	21,270,890	1120.1 (20 threads)
MMseqs2	Initial set (UniRef90-2015)	38,203,400	
	Round 1 nr set	23,317,430	277.7 (20 threads)
	Round 2 nr set	23,221,051	189.1 (20 threads)
	Round 20 nr set	23,205,925	185.7 (20 threads)
USEARCH	Initial set (UniRef90-2015, sampled)	1,600,000	
	Round 1 nr set	1,423,331	41.9 (1 thread)
	Round 2 nr set	1,423,062	38.6 (1 thread)
	Round 10 nr set	1,422,899	40.9 (1 thread)

表一：目前成熟演算法過濾 UniRef90-2015 並產生序列間相似度小於 70%之資料集所需回合數

● 演算法流程規劃



圖二：

(a) CD-HIT 演算流程圖

(b)(c) 為將兩步驟拆分開之演算流程圖，(b)為去冗餘部份，(c)為分群部份

如圖二(a)所示，CD-HIT 比對程序中包含兩層的迴圈，我們認為可以將其拆分成兩個獨立的程式，演算流程如圖二(b)、(c)所示。第一部份負責做同源序列去冗餘(類似 CD-HIT 兩層迴圈所做功能)，第二部份則利用第一部份輸出之代表性序列結果，將資料中與代表性序列有一定相似度的序列做分群。

對於只需使用代表性序列的使用者來說，僅執行第一部份程式不只可滿足其需求，還能大幅降低計算等待時間；而對於需要代表性序列和分群結果的使用者，再多跑一隻程式即可獲得所需結果，而兩程式的傳遞途徑也會在最終開發的軟體套件中被建置。

● 演算法開發大方向

■ 使用 Golang 語言進行演算法寫作

Golang 為中/高階編譯式語言，具有豐富函式庫、執行速度快等優點，尤其是其因應大數據時代而生的強大平行運算能力，對此演算法開發帶來極大的助益。

■ 將傳統演算流程拆分成兩部分

雖然資料在兩部份程式中傳遞會導致額外的計算時間，但將同源序列去冗餘和序列分群這兩部份拆分開，不僅可以讓我們針對各部份專一性地進行優化，還能更符合使用者的需求，因此我們認為總計算時間是有低於目前成熟演算法的可能性的。

■ 一對一同源序列比對寫作方式

我們原先認為如在與 query sequence 比對前先串接欲測試序列，可以有效加速整體比對的過程。因為在先前測試中，我們初步寫了一支以傳統演算法為模板的程式來進行一對一比對演算法和一對多比對演算法的比較，如表二所示，我們發現，當原始數據含有 100 條胺基酸序列時速度較快，因此推測此方法應當可以幫助減少整體計算時間。

資料集	傳統一對一比對演算法 執行所需時間(秒)	一對多比對演算法 所需執行時間(秒)
pdb 100	1.72	1.40
pdb 1K	88.88	111.85

表二：不同比對數量演算法初期測試-不同資料集下所需之執行時間

但在後來寫作過程中，我們漸漸發現「串接欲測試序列再進行比對」這個動作不僅有些耗時，還會占用極大記憶體空間，而且在針對特定符

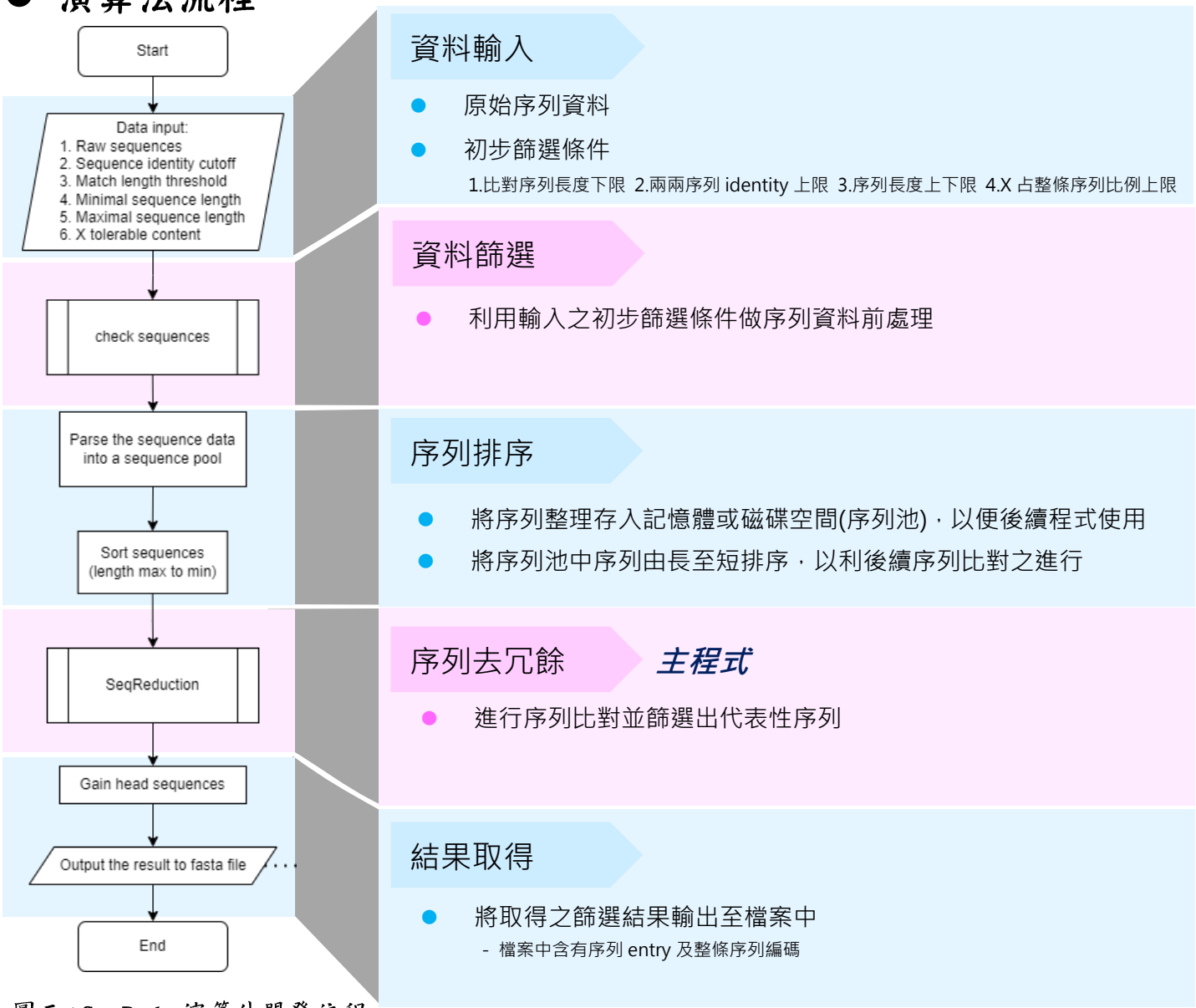
合條件之序列做移除或是結果輸出過程中，皆需多增加一些較冗的程式篇幅。因此經小組成員們和指導老師討論溝通後，我們決定改變最初的想法，回歸至傳統一對一同源序列去冗餘方式寫作，並對其進行優化。

■ 分散運算多執行緒計算

利用擁有強大平行運算能力的 Golang 語言寫作，再搭配上羅老師實驗室專利開發的分散運算方法及管理系統，來提高整體執行效率，使其之後面對海量資料計算時，能夠擁有更超群的實力脫穎而出。

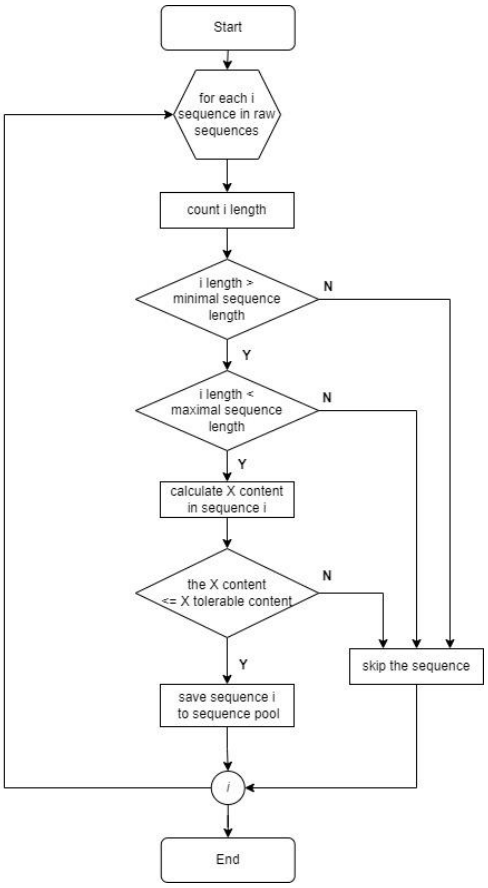
研究成果(RESULT)

● 演算法流程



圖三：SeqRedu 演算法開發流程

■ 資料篩選



對每一條輸入之序列進行篩選：

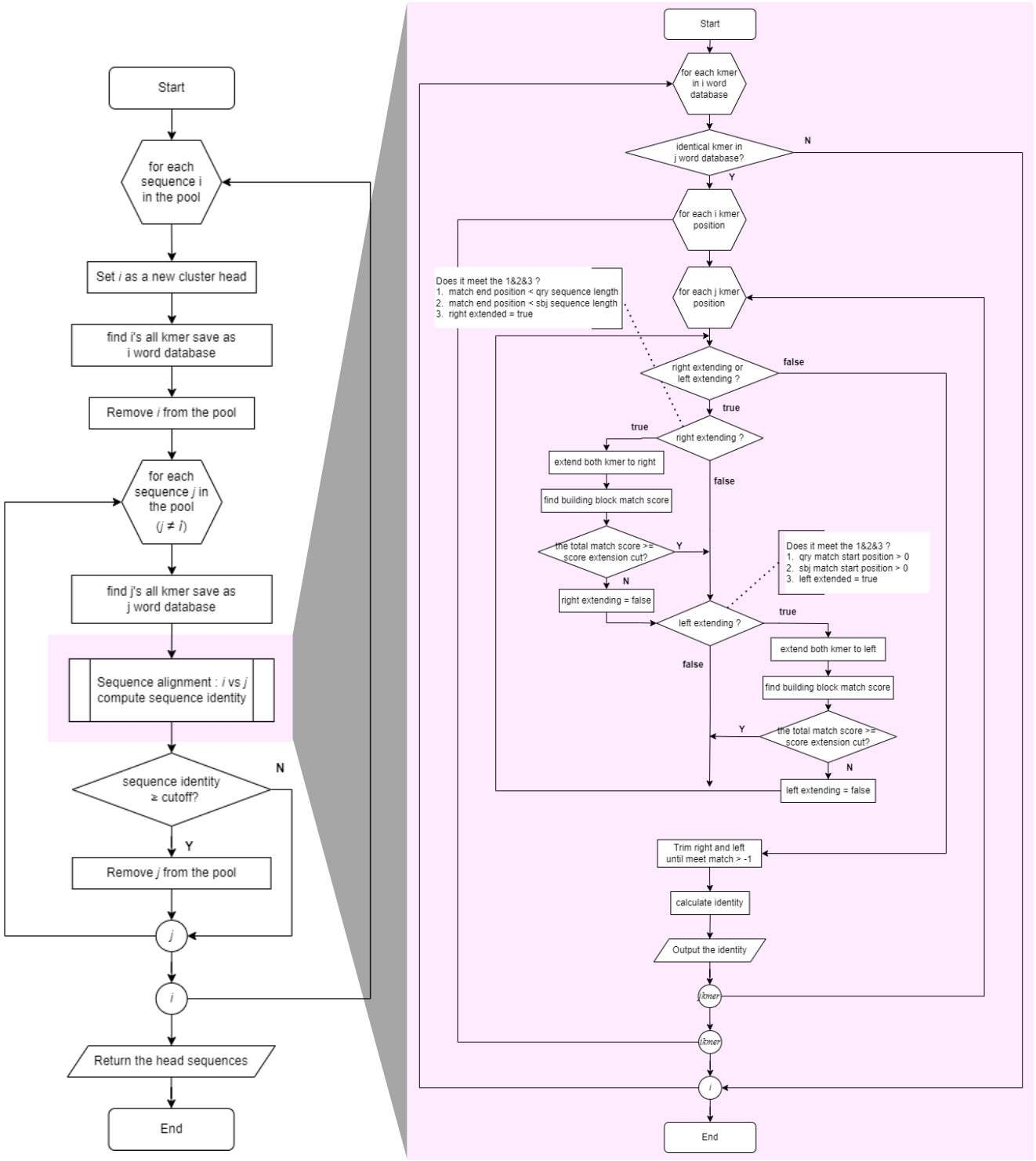
1. 確認長度是否大於使用者設定之長度下限
2. 確認長度是否小於使用者設定之長度上限
3. 計算字元 X 在整個序列中的比例確認是否小於設定值

如皆符合上面三項篩選條件，此序列將會被儲存至記憶體或磁碟空間中；如有一項不符合，則此序列會直接被跳過，不會進入下一步的計算中

圖四：SeqRedu 副程式-資料篩選流程

■ 序列去冗餘

經上游篩選並處理過後的序列會被存入序列池，並進入圖五的流程中。第一層迴圈中會先取出序列池中最開頭的序列直接當作代表性序列，計算儲存其文字 kmer(為某序列中某長度的序列短片段。如一序列：AGDEE，其長度為三的 kmer 有 AGD、GDE、DEE)，再進入第二層迴圈中。第二層迴圈運用分散運算一次進行多執行緒計算，透過另一個副程式(圖六)和其他序列進行序列比對後得出之相似度，進行篩選，如相似度大於使用者設定值，此序列將會被視為和代表性序列有一定相似度，而從序列池中被移除，等到所有執行緒都完成計算後，進入下一回合迴圈，重複直至序列池中所有序列都已比對完成。過程中如遇到已經計算過文字 kmer 的序列，將會從緩衝區中取出先前計算之結果，直接進行比對。



圖五：SeqRedu 主程式-序列去冗餘流程　　圖六：SeqRedu 主程式-相似度計算流程

■ 序列相似度計算

如圖六中所示，程式會先穿繞兩序列的文字 kmer 資料集合，如有相同的文字片段，則會透過已儲存在資料集合中的文字片段位置，往下進行 BLAST(Basic Local Alignment Search Tool)序列比對算法的計算，並切除頭尾分數較低之配對後，再計算兩序列相似度，得出結果回傳至序列去冗餘程式(圖五)，進行下一步計算。

● 演算法輸出成果

UniRef 與 PDB 我們都已寫好自動化下載、解壓縮、檔案格式統一化的基本處理程式，只不過尚未付諸實戰。因此成果展示所用之資料集為先前從 PDB 網頁下載之 2015 年資料集，並隨機加以拆分成不同序列大小的子檔案，分別為較小資料量的 pdb1K(1000 條蛋白質序列)及較大資料量的 pdb10K(1000 條蛋白質序列)。

不同 identity 有其最適宜的 kmer 長度[3]，如表三所示，成果展示之不同 identity 輸出，其相對應之 kmer 長度(word size)皆是依照此表進行調整。

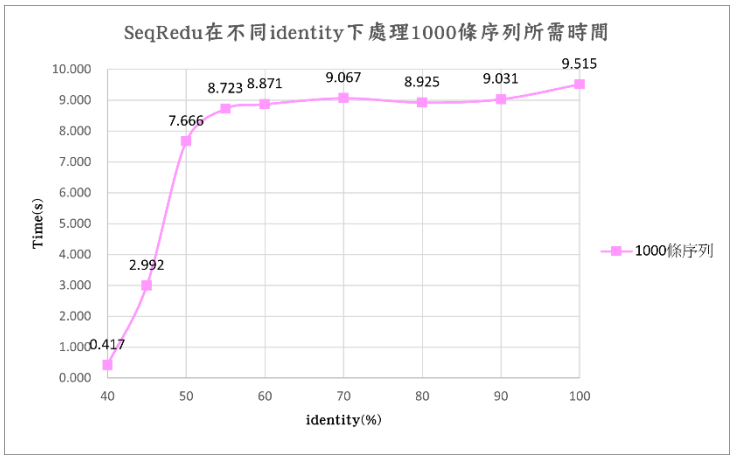
identity	40%	45%	50%	55%	60%	70%	80%	90%	100%
word size 長度	2	2	3	3	4	5	5	5	5

表三：不同 identity 和其最適宜之 kmer 長度對照表

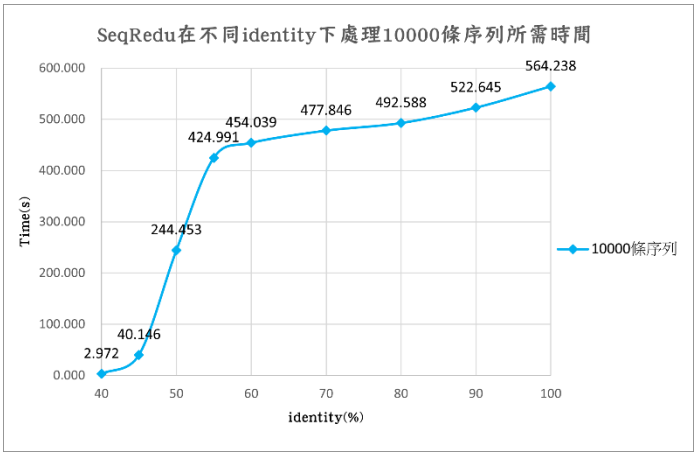
■ SeqRedu 在不同資料量和不同 identity 下所需之時間

下圖七為在不同資料量大小(序列長度為 1000 和 10000 條)、不同 identity 及不同長度 kmer 下，使用 10 個執行緒進行分散運算，連續跑 30 次的結果。

(a)



(b)

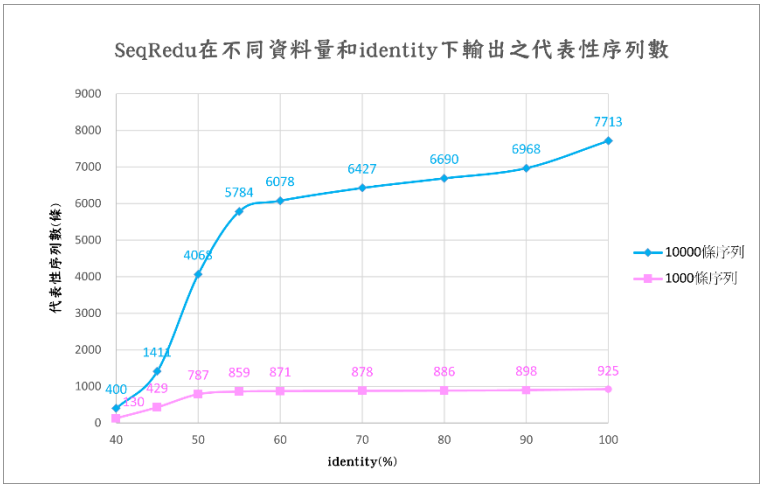


圖七：SeqRedu 處理不同資料量和 identity 所需之時間

從圖七結果可以發現，我們所開發之去冗餘演算法在低 identity 的時候，跑的速度會最快，而隨著 identity 越來越高(自左向右)，其速度會

漸成對數分布，在 identity 等於 50%左右時，其花費時間會有急遽上升現象，約在大於 60%時，上升速度漸趨平緩。

■ 演算法在不同資料量和不同 identity 下輸出之代表性序列數

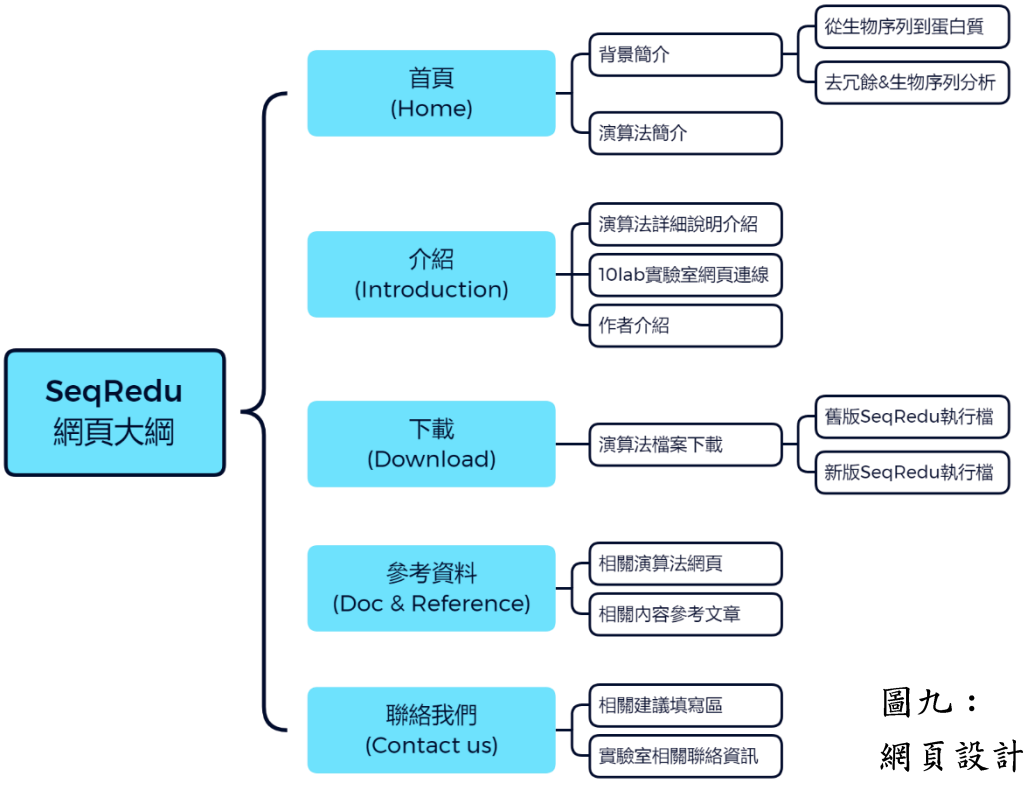


圖八：
SeqRedu 在不同資料量和 identity 下輸出之代表性序列數

從圖表結果可以發現，隨著 identity 下降(自右向左)，不同資料量的序列皆有呈現下降趨勢，在 identity 較高的部份下降較少，identity 變低時(<60%)，資料量有顯著下降的趨勢。

● 演算法網頁製作

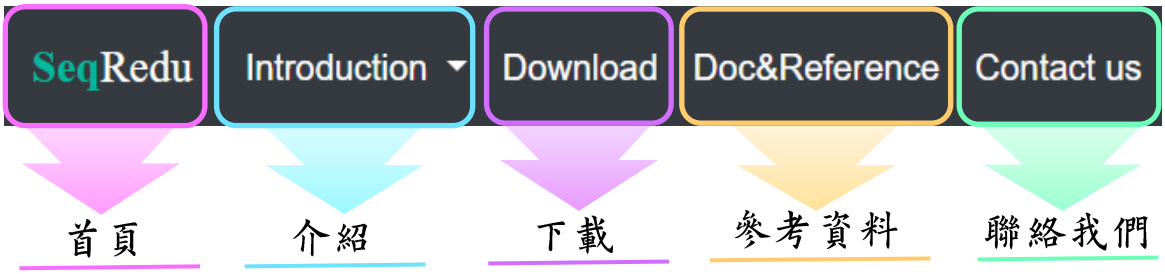
■ 網頁大綱



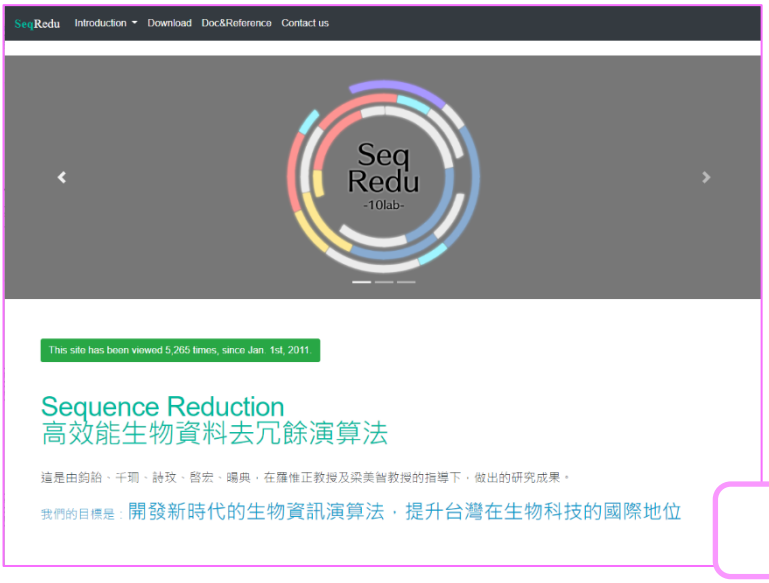
圖九：
網頁設計大綱

■ 網頁成果展示

首頁上方有相應條目，可供使用者點選



SeqRedu 首頁(Home)

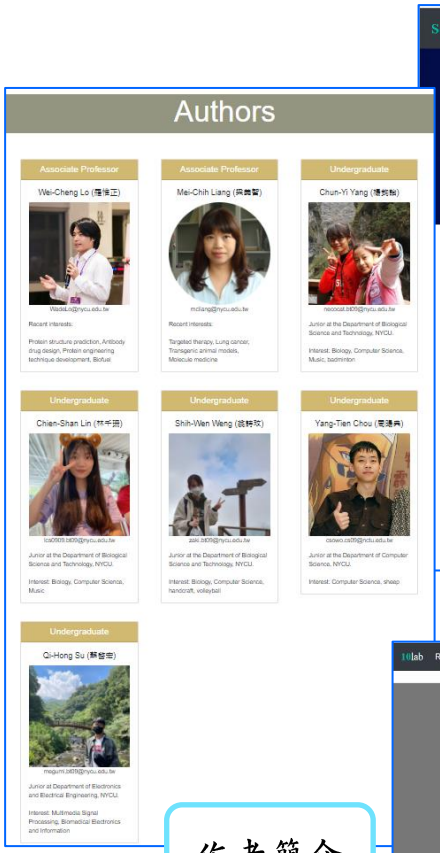


首頁

演算法概述

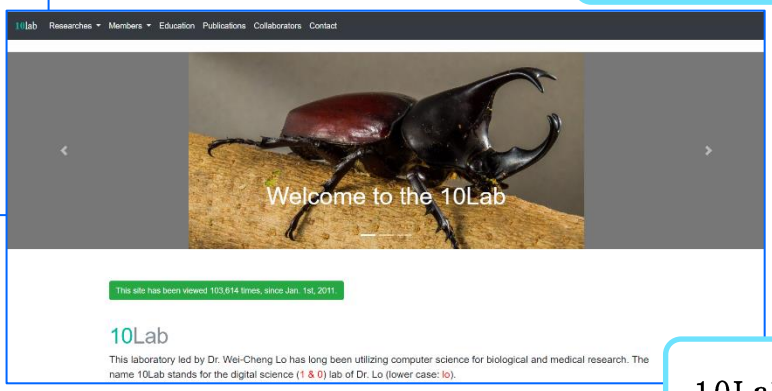
背景簡介

Introduction 介紹(Introduction)



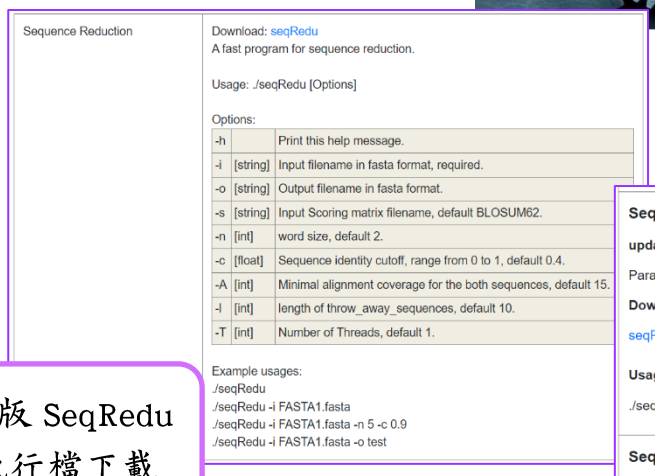
作者簡介

演算法詳細介紹



10Lab 網頁

Download 下載(Download)


Sequence Reduction

Download: seqRedu
A fast program for sequence reduction.

Usage: ./seqRedu [Options]

Options:

-h		Print this help message.
-i	[string]	Input filename in fasta format, required.
-o	[string]	Output filename in fasta format.
-s	[string]	Input Scoring matrix filename, default BLOSUM62.
-n	[int]	word size, default 2.
-c	[float]	Sequence identity cutoff, range from 0 to 1, default 0.4.
-A	[int]	Minimal alignment coverage for the both sequences, default 15.
-l	[int]	length of throw_away_sequences, default 10.
-T	[int]	Number of Threads, default 1.

Example usages:
./seqRedu
./seqRedu -i FASTA1.fasta
./seqRedu -i FASTA1.fasta -n 5 -c 0.9
./seqRedu -i FASTA1.fasta -o test

新版 SeqRedu 執行檔下載


Sequence Reduction v.2 (build date Sep, 11, 2022)

update

Parallel Processing

Download

seqReduction: A fast program for sequence reduction.

Usage

./seqReduction [Options]

Sequence Reduction v.1 (build date Jul, 1, 2022)

Download

seqReduction: A fast program for sequence reduction.

Usage

./seqReduction [Options]

舊版 SeqRedu 執行檔下載

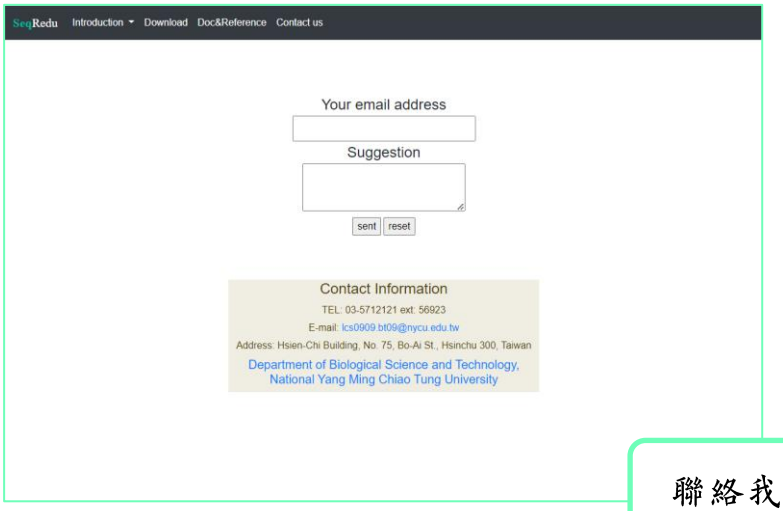
Doc&Reference 參考資料(Doc&Reference)

相關演算法網頁

相關內容參考文章

1. CD-HIT official website: https://sites.google.com/view/cd-hit
2. Juan SH, Chen TR, Lo WC: A simple strategy to enhance the speed of protein secondary structure prediction without sacrificing accuracy. PLoS One 2020, 15:e0235153.
3. Chen TR, Lo CH, Juan SH, Lo WC: The influence of dataset homology and a rigorous evaluation strategy on protein secondary structure prediction. PLoS One 2021, 16:e0254555.
4. Weizhong Li, Lukasz Jaroszewski and Adam Godzik: Tolerating some redundancy significantly speeds up clustering of large protein databases. BIOINFORMATICS, Vol. 18 no. 1 2002 Pages 77–82.
5. Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. 48, 443–453 (1970)
6. Smith, T.F. and Waterman, M.S. (1981) Identification of Common Molecular Subsequences. Journal of Molecular Biology, 147, 195-197. http://dx.doi.org/10.1016/0022-2836(81)90087-5

Contact us 聯絡我們(Contact us)



聯絡我們

結果討論分析(DISCUSSION)

● 目前成熟演算法-CD-HIT

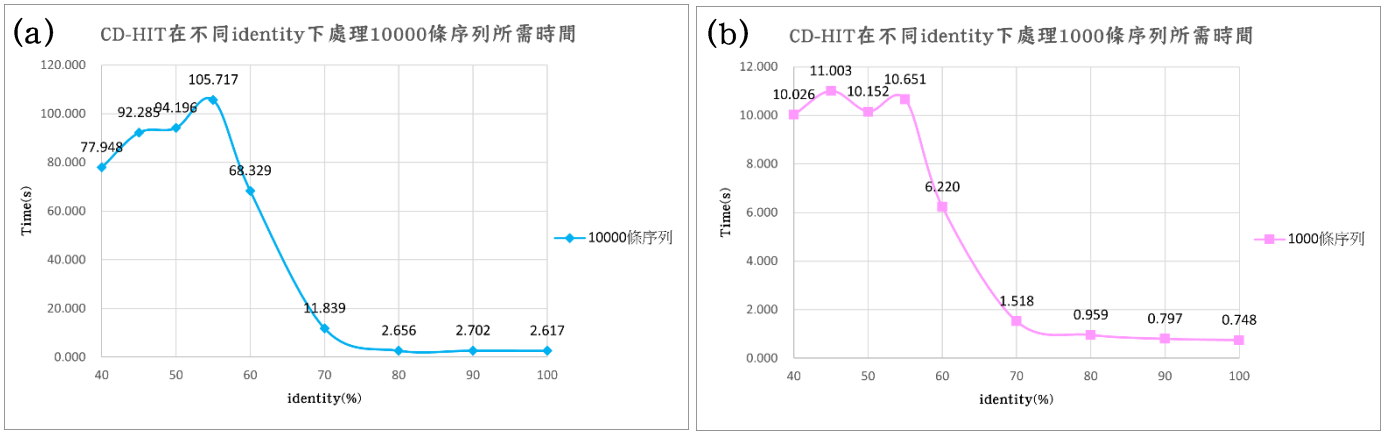
鑒於使用 UniRef90-2015 資料集對這三個成熟演算法(CD-HIT、USEARCH、MMseqs2)進行測試所觀察到之結果，我們選擇使用 CD-HIT 作為演算法開發及比較的基礎。

■ CD-HIT 演算法

演算流程詳見圖二(a)，CD-HIT 去冗餘和分群兩步驟沒有拆分開。

■ CD-HIT 在不同資料量和不同 identity 下所需之時間

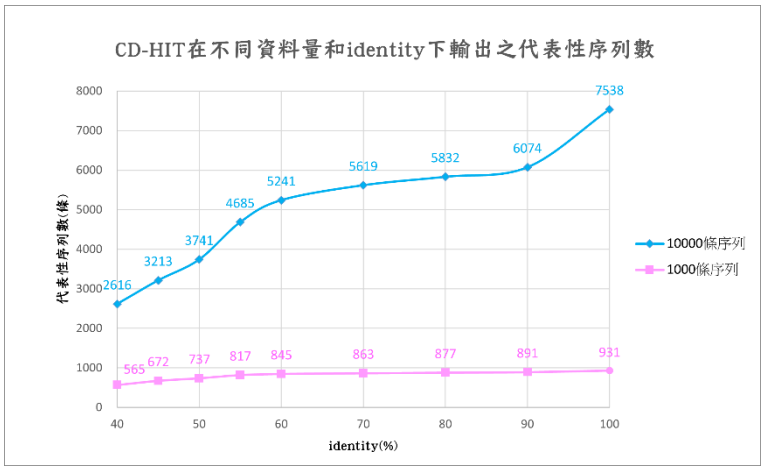
下圖十為在不同資料量大小(序列長度為 1000 和 10000 條)、不同 identity 及不同長度 kmer 下，使用 10 個執行緒進行分散運算，連續跑 30 次的結果。kmer 長度(wordsize)設定皆是依照表 3 進行調整。



圖十：CD-HIT 處理不同資料量和 identity 所需之時間

從圖十結果可以發現，CD-HIT 演算法在低 identity 的時候，花費的時間最多，然而隨著 identity 越來越高(由左至右)，其速度會逐漸加快，大致呈現與 SeqRedu 相反的對數分布(圖七)，在 identity 約等於 60%時其花費時間快速下降，約在大於 70%時，下降速度漸趨平緩。

■ CD-HIT 在不同資料量和不同 identity 下輸出之代表性序列數

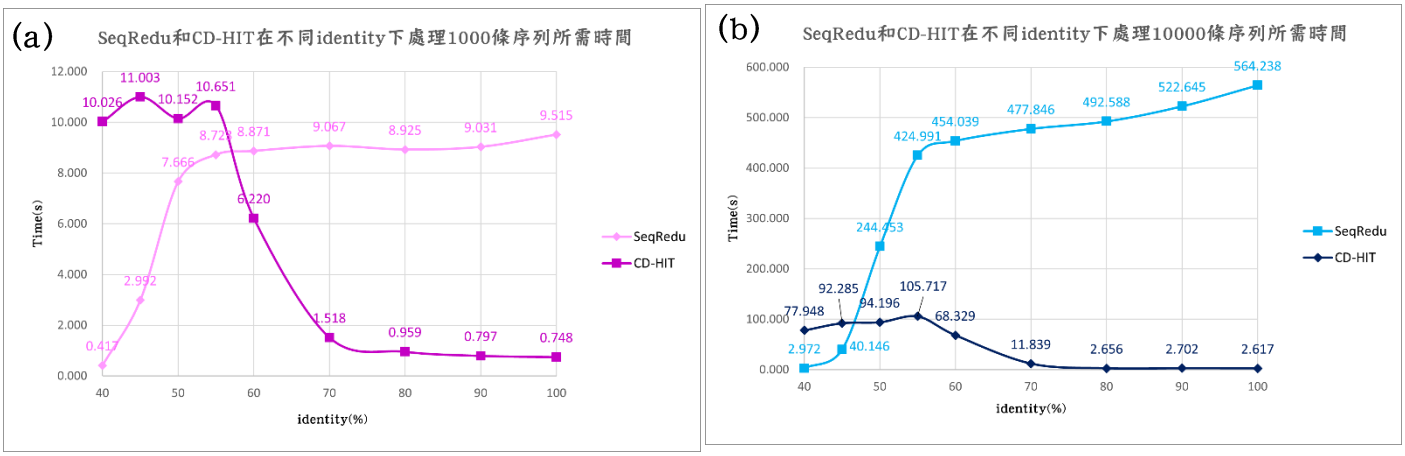


圖十一：
CD-HIT 在不同資料量和 identity 下輸出之代表性序列數

從圖表結果可以發現，隨著 identity 下降(由右至左)，不同資料量的序列皆有呈現下降趨勢，其下降的趨勢和 SeqRedu(圖八)相比則較為平緩，大致呈線性下降。1000 條序列資料集因資料量較小，故下降幅度較不如 10000 條序列資料來得明顯。

● SeqRedu 和 CD-HIT 比較

■ 兩者在不同資料量和不同 identity 下所需之時間

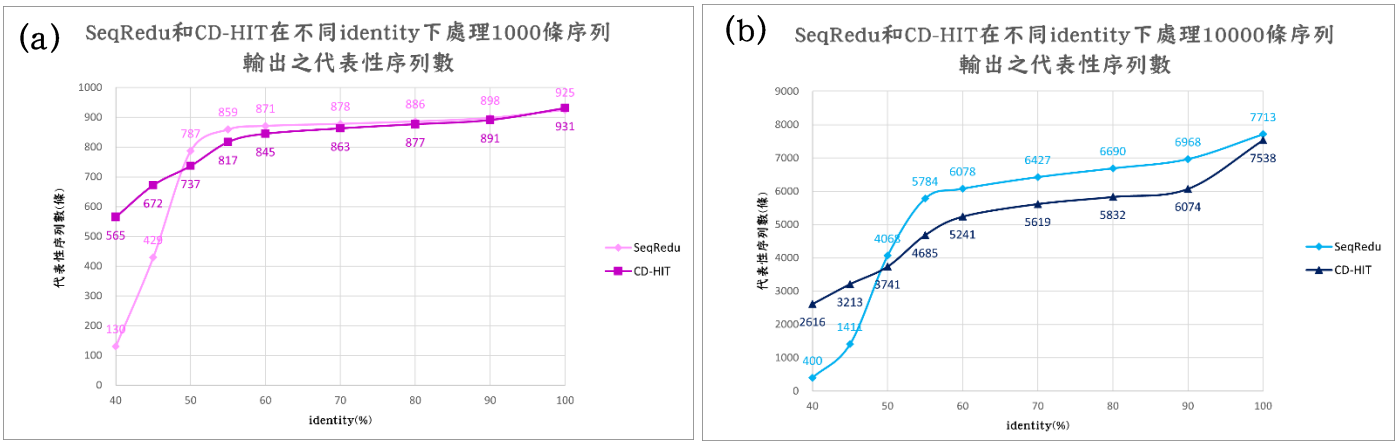


圖十二：CD-HIT 和 SeqRedu 在不同資料量和不同 identity 下所需之時間

從圖十二中可以發現。隨著處理序列數量上升(a 至 b 圖)，在低 identity 時，SeqRedu 的計算速度可以較 CD-HIT 快上許多，尤其當資料量越多時，耗費時間差距會越顯著；但在高 identity 時，SeqRedu 所耗費之時間會急遽上升，尤其在處理序列數為 10000 條(b)時，CD-HIT

計算 100% identity 只需約 2 秒，而 SeqRedu 所耗費的時間卻高達 9 分鐘左右。因此，如何優化 SeqRedu，提升其在高 identity 時的計算效能，將是我們在未來開發過程中，需要彼此充分思考討論並解決的一大關卡。

- **兩者在不同資料量和不同 identity 下輸出之代表性序列數**



圖十三：CD-HIT 和 SeqRedu 在不同資料量和不同 identity 下輸出之代表性序列數

從圖十三中可發現，在高 identity 時，兩演算法彼此輸出之代表性序列數會較相近，SeqRedu 輸出之序列數會稍多於 CD-HIT，但當 identity 降低(由右至左)時，兩個演算法在不同資料量大小下，皆會在約 50% identity 處發生交叉，當 identity 低於 50%，SeqRedu 所篩選出之序列數相較 CD-HIT 會有明顯的降低。

- **得出此結果可能原因- SeqRedu 使用 gapless alignment 演算方法**
  - **一般成熟演算法-gap alignment**

一般成熟演算法會將整個流程拆成兩大部分：

- 第一部份-篩選：

  找到 kmer 做左右延伸，直至遇到序列中兩元素得分為負分即停止，初步過濾出不符合條件的序列，減少下游程式要處理的資料量，再將這些符合的片段結果，放到下游程式做計算。

- 第二部分-dynamic programming 計算：

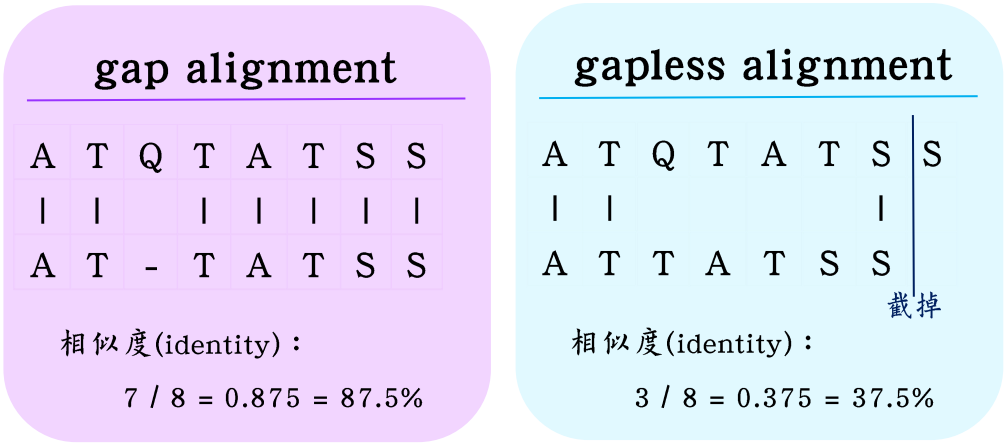
如 Needleman-Wunsch 演算法[4](global alignment)、Smith-Waterman 演算法[5](local aignment)。在計算過程中會透過開 gap 的方式(gap alignment)，來使序列元素彼此能夠匹配的更好，再藉由得分扣分的方式得出最好的結果，最後再算出相似度，並依據所得之數據進行序列去冗餘暨分群。

■ **SeqRedu 演算法- gapless alignment**

SeqRedu 則是將兩大運算(篩選、dynamic programming)合併，利用左右延伸過程算出的分數及匹配數，直接得出相似度(identity)。

SeqRedu 亦會利用 kmer 做左右延伸，但遇到兩元素得分為負分時還是會繼續算，直至序列中匹配元素得分總合為負分才停止，在計算過程中不會開 gap(gapless alignment)，程式停止後接著切除頭尾分數較低之配對，並計算兩序列相似度，最後利用輸出之相似度進行序列去冗餘暨分群。

因此，SeqRedu 較不容易算出高 identity 的結果，例如圖十四 ATQTATSS、ATTATSS 兩序列的比對，在 gap alignment(一般成熟演算法)下，其相似度為 87.5%，而在 gapless alignment(SeqRedu)下計算的相似度僅有 37.5%



圖十四：ATQTATSS、ATTATSS 兩序列在有無開 gap 之情況下的 identity

- **identity 高速度較慢原因討論**

  如圖五演算法流程所示，如相似度大於使用者設定值，此序列將會被視為和代表性序列有一定相似度，而從序列池中被移除，但因為 SeqRedu 為 gapless alignment，不容易像 CD-HIT (gap alignment) 一樣偵測到演化過程中發生長度變化之兩相似序列，使得算出序列之 identity 會偏低(圖十四)，序列不容易大於使用者設定值而被移除，因此會繼續留在序列池中被計算，所以在高相似度時計算速度會相較 CD-HIT 來得慢。

  兩序列相似度低時，SeqRedu 和 CD-HIT 兩者的計算結果差不多，但因為 SeqRedu 企圖將兩大運算(篩選、dynamic programming)合併，以單一動作去完成兩件事，以省去第二部分較耗時之 dynamic programming 計算，因此低相似度時計算速度會較 CD-HIT 來得快，也就是說低相似度時 SeqRedu 會較 CD-HIT 來得有優勢。

未來優化方向(FUTURE)

經由輸出成果和與 CD-HIT 比較結果來看，SeqRedu 需要找到一平衡點，使其在速度獲得提升的同時，還能具備一定的嚴格度，才得以符合未來使用之效能和實用性。

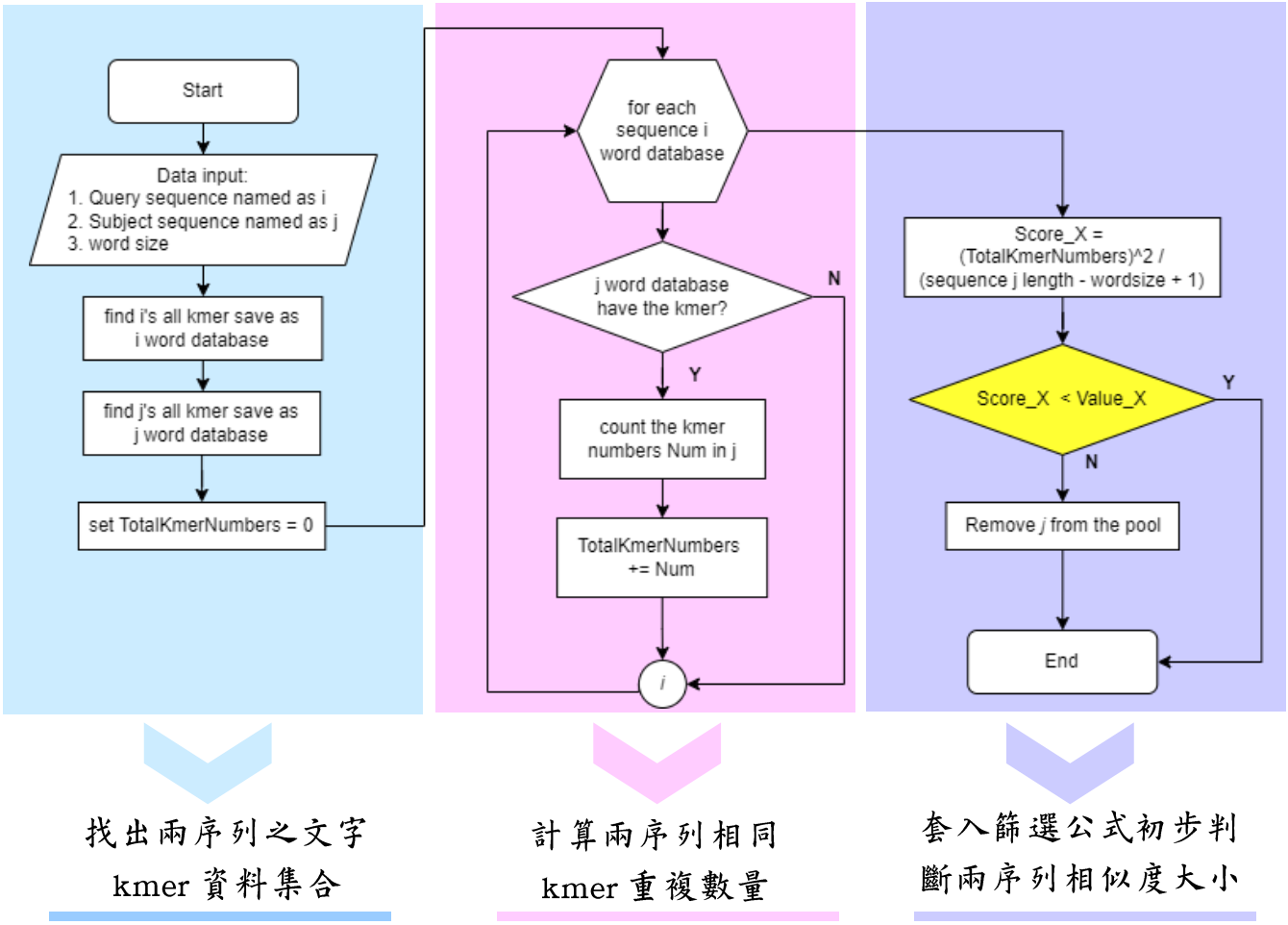
● 速度

依圖七所示，SeqRedu 在低 identity 時速度有較好的表現，但是當 identity 上升時，其計算速度會隨之下降，尤其是在 identity> 50%時，其花費的時間會急遽上升。

之後的優化方向會試著在程式前半部新增一個篩選函式，大致流程如下所示。利用老師多年寫作經驗所得之經驗公式(圖十五中黃色框起處)計算 Score_X，並期望在未來能找出一個切點(Value_X)和此篩選公式連結，使得只要 Score_X > Value_X 即表示兩序列有一定相似度，可以直接將待測序列

從序列池中移除，不用再到下游步驟中進一步做計算，以減少下游計算之資料量，盼能對整體計算速度達到提升的效果。

圖十五：未來優化方向 - 在程式前半部新增一篩選函式之流程圖



致謝(ACKNOWLEDGEMENTS)

感謝兩位指導教授，羅惟正老師和梁美智老師在整個計畫過程中給予的引導、指導、協助及建議，讓我們計劃得以順利推進的同時，還能學習如何整合不同領域之所長，跨領域學習。與指導老師和小組成員一同面對各式問題，攜手解決一道道難關的執行計畫期間，我們不僅獲得了滿滿的知識點，也看到了一個比之前更進步的自己。

經費來源(FUNDING)

- 博雅書院
  - 國立陽明交通大學「博雅學生社群 Active learning」
- 科技部
  - 台灣國家科學及技術委員會-大專生研究計畫 111-2813-C-A49 -030 -E

參考資料(REFERENCES)

1. Juan SH, Chen TR, Lo WC: A simple strategy to enhance the speed of protein secondary structure prediction without sacrificing accuracy. PLoS One 2020, 15:e0235153.
2. Chen TR, Lo CH, Juan SH, Lo WC: The influence of dataset homology and a rigorous evaluation strategy on protein secondary structure prediction. PLoS One 2021, 16:e0254555.
3. Weizhong Li, Lukasz Jaroszewski and Adam Godzik: Tolerating some redundancy significantly speeds up clustering of large protein databases. BIOINFORMATICS, Vol. 18 no. 1 2002 Pages 77–82.
4. Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. 48, 443–453 (1970)
5. Smith, T.F. and Waterman, M.S. (1981) Identification of Common Molecular Subsequences. Journal of Molecular Biology, 147, 195-197.