مجلة جامعة طرطوس للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (٩) العدد (٥) ٢٠٢٥
Tartous University Journal for Research and Scientific Studies - engineering Sciences Series Vol. (9) No. (5) 2025

تقييم أداء خوارزميات المعلوماتية الحيوية لدراسة تسلسل البروتين في البيئة التسلسلية والبيئة التفرعية باستخدام منصات الحوسبة السحابية

م. سنان غازي شاش*

(تاريخ الإيداع ٢٠٢٥/٢/٢٦ . قُبِل للنشر في ٢٠٢٥/٤/٢٣)

□ ملخّص□

التغيرات التي قد تطرأ على سلاسل البروتين أو المادة الوراثية يمكن أن تؤدي إلى ظهور أمراض متنوعة، بعضها قد يكون مهدداً للحياة، مما يجعل التشخيص المبكر أمراً حاسماً لحماية صحة المريض. وفي إطار السعي الحثيث لاكتشاف هذه التغيرات في وقت مبكر، تم تطوير خوارزميات متخصصة في المعلوماتية الحيوية، تهدف إلى تحليل سلاسل البروتينات ودراسة تسلسل الشيفرة الجينية في الحمض النووي (DNA). ومع كون البيانات البيولوجية هائلة الحجم ومعقدة، فإن استخدام الأنظمة التقليدية لمعالجة هذه الخوارزميات يصبح تحدياً حقيقياً، حيث يحتاج الأمر إلى وقت طويل لتعامل الأنظمة مع هذه البيانات الضخمة.

في هذا البحث، قمنا باستخدام خوارزمية تسلسلية من خوارزميات المعلوماتية الحيوية لاكتشاف سلاسل البروتين، ثم قمنا بتطوير هذه الخوارزمية إلى شكل تفرعي، ونفذناها عبر الحوسبة التفرعية باستخدام وحدات المعالجة الرسومية (GPU) المتوفرة على منصة Google Colab. وبعد ذلك، أجرينا مقارنة شاملة بين النتائج من حيث الدقة، زمن التنفيذ، واستهلاك الموارد المادية مقارنة بالخوارزمية التسلسلية الأصلية. الهدف من هذه المقارنة كان تحديد الحل الأمثل لاكتشاف التغيرات في السلاسل البروتينية، بحيث نحقق دقة أعلى، زمن تنفيذ أسرع، وتكلفة أقل.

كلمات مفتاحية: المعلوماتية الحيوية، الأحماض الأمينية، الأسس الآزوتية، الخلية، الجينوم، البروتين، سلاسل DNA، NCBI، BLAST، Google Colab، معالج GPU ، SW، NW .

*: ماجستير في قسم هندسة النظم الحاسوبية والالكترونية، كلية هندسة تكنولوجيا المعلومات والاتصالات، جامعة طرطوس – سورية
البريد الالكتروني: sinanshash123456789@gmail.com
رقم الموبايل: ٠٩٣٢٦٤٥١٠٠

مجلة جامعة طرطوس للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (٩) العدد (٥) ٢٠٢٥
Tartous University Journal for Research and Scientific Studies - engineering Sciences Series Vol. (9) No. (5) 2025

Performance evaluation of bioinformatics algorithms for studying protein sequences in sequencing and branching environments using cloud computing platforms

Eng. Sinan Shash*

(Received 26/2/2025 . Accepted 23/4/2025)

□ **ABSTRACT** □

Changes in protein chains or genetic material can lead to the emergence of various diseases, some of which may be life-threatening, making early diagnosis crucial to protect the patient's health. In the relentless pursuit of discovering these changes early, specialized bioinformatics algorithms have been developed, aiming to analyze protein chains and study the sequence of the genetic code in DNA. With the large volume and complexity of biological data, using traditional systems to process these algorithms becomes a real challenge, as it takes a long time for systems to deal with this huge data.

In this paper, we used a bioinformatics sequencing algorithm to detect protein sequences, then developed this algorithm into a parallel form, and implemented it via parallel computing using GPUs available on the Google Colab platform. Then, we made a comprehensive comparison of the results in terms of accuracy, execution time, and resource consumption compared to the original sequencing algorithm. The goal of this comparison was to determine the optimal solution for detecting changes in protein sequences, so that we can achieve higher accuracy, faster execution time, and lower cost.

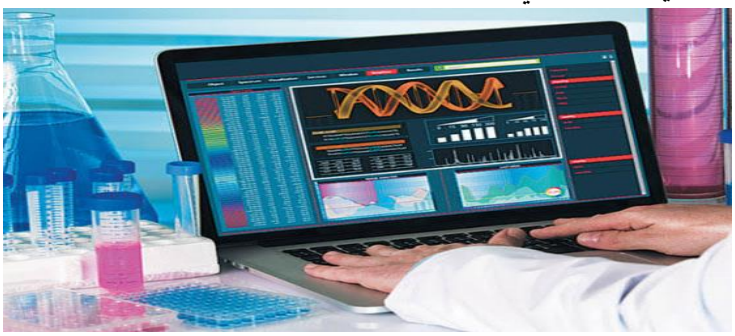
Keywords: bioinformatics, amino acids, nitrogenous bases, cell, genome, protein, DNA sequences, NCBI , BLAST , Google Colab , Processor GPU, SW ، NW.

*** Master at CESE Department, Faculty of Communication and Information Engineering Technology, Tartous University, Syria**

١. مقدمة

في العقود الأخيرة، تم تحقيق تقدم مهم في المجالات البيولوجية والطبية الحيوية، والتي تم تعزيزها من خلال التقدم المهم في التقنيات التجريبية، بالإضافة إلى ذلك، أدت التقنيات الأخرى عالية الإنتاجية لقياس التعبير الجيني أو تركيزات البروتين أو المركبات في الخلايا، إلى ثورة حقيقية في البحوث البيولوجية والطبية. كل هذه التقنيات قادرة حالياً على توليد كميات هائلة مما يسمى ببيانات omics، والتي يمكن استخدامها لتعزيز البحث العلمي في علوم الحياة وتعزيز تطوير تقنيات جديدة في الرعاية الصحية والتكنولوجيا الحيوية والمجالات ذات الصلة[1].

المعلوماتية الحيوية هي مجال متعدد الأوجه يشارك في تطوير الأساليب والتقنيات والأدوات البرمجية للبيانات البيولوجية. من أجل فهم وتحليل البيانات البيولوجية بشكل أفضل، تدمج المعلوماتية الحيوية مجالات علوم الكمبيوتر وعلوم الطبيعة والرياضيات معاً. إنه يتعامل مع جمع المعلومات ونمذجتها ومعالجتها لتحليلها وتصور البيانات البيولوجية ويساعد بدوره في إنشاء خوارزميات وأدوات جديدة[2].



الشكل(١) شكل توضيحي يبين استخدام الحاسب في تحليل المعلومات البيولوجية.

أجرى الباحثون في المرجع [٤] دراسة تم فيها نقل برنامج MEGADOCK، المستخدم في التنبؤ بتفاعل البروتين–بروتين (protein–protein docking)، إلى منصة Microsoft Azure كنموذج لبيئة حوسبة سحابية عالية الأداء (HPC). تم إعداد بيئة حوسبة سحابية متوازية تضمنت ما يصل إلى ١٦٠٠ نواة لوحدة المعالجة المركزية و٩٦٠ وحدة معالجة رسومات، باستخدام أربعة أنواع متماثلة من وحدات المعالجة المركزية ونوعين من وحدات معالجة الرسومات، ثم تم تقييم كفاءة الأداء في الحوسبة المتوازية.

قدَّم الباحثون في المرجع [٥] دراسة اقترحوا فيها مجموعة من الأساليب والأدوات التي يمكن تصنيفها إلى ثلاثة فئات رئيسية: تخزين البيانات الضخمة، وتصميم الخوارزميات الفعالة، والحوسبة المتوازية. هدفت هذه الدراسة إلى استكشاف تقنيات البرمجة المتوازية الشائعة لمعالجة تسلسل الجينوم. تم تقديم ثلاثة نماذج رئيسية للحوسبة المتوازية بناءاً على بنى الأجهزة المستخدمة، حيث صُنّف كل نموذج إلى نوعين أو ثلاثة أنواع، وتم تحليل ميزاتها بالتفصيل. بعد ذلك، تم تناول دور الحوسبة المتوازية في معالجة تسلسل الجينوم من خلال أربعة تطبيقات رئيسية: محاذاة تسلسل الجينوم، واستدعاء تعدد أشكال النيوكليوتيدات الأحادية، والمعالجة المسبقة لتسلسل الجينوم، واكتشاف الأنماط والبحث. كما تم استعراض التحديات الحالية والاتجاهات المستقبلية في تقنيات الحوسبة المتوازية.

استعرض الباحثون في المراجع [٦]–[١٦] استخدام منصة Hadoop في تطبيقات المعلوماتية الحيوية الهيكلية، حيث يُعد Hadoop إطاراً حديثاً ضمن الحوسبة الموزعة، مصمماً لتحليل كميات هائلة من بيانات

البروتين. يلعب هذا الإطار دوراً أساسياً في دراسة الإنتاجية العالية لرسو البروتين الترابطي، وتجميع مجمعات البروتينات الترابطية، والمحاذاة الهيكلية. كما تم تسليط الضوء على عدد من التطبيقات التي تعتمد على Hadoop لتحقيق تحليلات عالية الإنتاجية وقابلة للتوسع، بالإضافة إلى استعراض بعض تطبيقات المعلوماتية الحيوية التي تسهم في فهم مبادئ تنفيذ الخوارزميات باستخدام نمط البرمجة MapReduce.

٢. أهمية البحث وأهدافه

تتمثل أهمية هذا البحث في إجراء تحليل دقيق لخوارزميات المعلوماتية الحيوية وتطبيق بيئة الحوسبة التفرعية لتحقيق نتائج أسرع مع الحفاظ على نفس مستوى الدقة. كما يكمن جوهر البحث في تحويل الخوارزميات المكتوبة بأسلوب تسلسلي إلى صيغة تفرعية، مما يتيح تنفيذها في بيئات الحوسبة الموزعة، بالإضافة إلى استخدام منصة Google Colab السحابية للعمل عليها. كما يتناول البحث دراسة تأثير زيادة عدد النياسب(Threads) على أداء الخوارزمية، مع التركيز على تأثير ذلك على الدقة، زمن التنفيذ، وتكاليف المعالجة.

٣. طرائق البحث ومواده

٣,١. المعلوماتية الحيوية

يُعد علم المعلوماتية الحيوية مجالاً بحثياً حديث النشأة ومتعدد التخصصات، حيث يجمع بين علوم الحاسوب، والرياضيات، والعلوم البيولوجية بهدف تحليل وتفسير الظواهر البيولوجية المختلفة. يهتم هذا المجال بمعالجة كميات هائلة من البيانات البيولوجية الناتجة عن المشاريع العلمية الضخمة، مثل مشروع الجينوم البشري. ويُستخدم علم المعلوماتية الحيوية بشكل أساسي ومتزايد في مجالين رئيسيين من العلوم البيولوجية: علم الجينوم (Genomics)، الذي يركز على دراسة التركيب والوظيفة للمجموعة الكاملة من جينات الكائن الحي، وعلم البروتين (Proteomics)، الذي يتناول دراسة التركيب والوظيفة للمجموعة الكاملة من بروتينات الكائن الحي [٨].

بالإضافة الى ذلك فإن علم المعلوماتية الحيوية يستخدم في فروع كثيرة من العلم البيولوجي مثل علم خرائط التمثيل الغذائي وشبكات الاتصال الخلوية والتفاعلات البيوكيميائية وغيرها، وتهدف كل هذه النواحي في علم المعلوماتية الحيوية إلى فهم الأنظمة البيولوجية المعقدة.

وقد أدى التوسع الكبير في هذا المجال إلى ارتباطه بمجموعة كبيرة من العلوم الأخرى في محيط الكائن الحي ولينتج مايسمى بالنظام البيولوجي المتكامل"Integrated System Biology" حيث أن النظام البيولوجي المتكامل يعدّ وسيلة لحل وتفسير العديد من التساؤلات البيولوجية المعقدة، وهو يتكون من اندماج علم الجينوم وعلم البروتين وعلم المعلوماتية الحيوية والعديد من العلوم الأخرى المتعلقة بها، وذلك لشرح وتفسير كيفية عمل المسار الحيوي في الخلية والتعرف على كيفية تفاعل الجينات والبروتينات والعناصر الأخرى الداخلية والخارجية المرتبطة بالمسار الحيوي مع بعضها البعض ومع البيئة المحيطة [8].

٣,٢. أهداف المعلوماتية الحيوية

الغاية الأساسية من المعلوماتية الحيوية هي تحقيق فهم أعمق للخلية الحية وآلية عملها على المستوى الجزيئي، من خلال تحليل البيانات الأولية لتسلسل الجزيئات والبنية الهيكلية، كما تتيح أبحاث المعلوماتية الحيوية إمكانية استخلاص رؤى جديدة وتقديم منظور شامل حول الخلية.

يُعد تحليل بيانات التسلسل أداة فعالة لفهم وظائف الخلية، نظراً لأن تدفق المعلومات الوراثية يتبع ما يُعرف بـ"**العقيدة المركزية**" في علم الأحياء، حيث يتم نسخ الحمض النووي (DNA) إلى الحمض النووي الريبي (RNA)،

ثم يُترجم الأخير إلى بروتينات[٧]. وتُعتبر البروتينات العنصر الأساسي في تنفيذ الوظائف الخلوية، حيث تعتمد قدراتها في النهاية على تسلسلها الجزيئي. لذلك، أثبتت الدراسات أن استخدام تحليل التسلسل إلى جانب النهج الهيكلي يمكن أن يكون وسيلة مثمرة لفهم الوظائف البيولوجية المختلفة.

٣,٢,١. المادة الوراثية DNA:

تُعتبر المادة الوراثية أحد المركبات الجزيئية الحيوية الأساسية في الجسم (الشكل(2))، حيث تحتوي كل خلية على الحمض النووي الذي يُوجد داخل نواتها، والذي يميز كل فرد عن غيره من الكائنات الحية. تحمل المادة الوراثية المعلومات والتعليمات المسؤولة عن التطور والنمو والتكاثر، حيث يتم توارث المعلومات الجينية من الوالدين إلى الأبناء، بحيث يُورَّث نصفها من الأب والنصف الآخر من الأم. تلعب التغيرات الجينية في المادة الوراثية دوراً رئيسياً في تحديد الصفات الفردية، مثل لون العيون، كما أنها قد تكون سبباً في الإصابة ببعض الأمراض[٧]. باختصار، تحتوي المادة الوراثية على الشيفرة الجينية الفريدة لكل شخص، بالإضافة إلى المعلومات اللازمة لبناء البروتينات الضرورية لوظائف الجسم المختلفة.

الشكل(٢) شريط الـ DNA.

٣,٣. تطبيقات المعلوماتية الحيوية

لم تعد المعلوماتية الحيوية مقتصرة على أبحاث البيولوجيا الجينومية والجزيئية الأساسية فحسب، بل باتت تلعب دوراً محورياً في العديد من مجالات التكنولوجيا الحيوية وعلوم الطب الحيوي. فقد أسهمت بشكل كبير في تطوير الأدوية المعتمدة على المعرفة، وتحليل الحمض النووي لأغراض قانونية، إضافة إلى تطبيقاتها الواسعة في التكنولوجيا الحيوية الزراعية[9].

تتيح معرفة الهياكل ثلاثية الأبعاد للبروتينات تصميم جزيئات ترتبط بدقة وخصوصية عالية بالموقع المستهدف على البروتين، ويؤدي هذا النهج المعتمد على المعلوماتية إلى تقليل الوقت والتكلفة اللازمين لتطوير أدوية أكثر فاعلية وأقل سمية مقارنةً بالأساليب التقليدية القائمة على التجربة والخطأ. وفي مجال الطب الشرعي، أصبحت نتائج التحليل الجزيئي للسلالة معترفاً بها كدليل في المحاكم الجنائية[5][9].

٣,٤. خوارزميات المعلوماتية الحيوية

غالباً ما تقدم الخوارزميات تنبؤات غير دقيقة تفتقر إلى المعنى عند وضعها في سياق بيولوجي، حيث يمكن أن تؤدي الأخطاء في محاذاة التسلسل إلى تأثيرات غير مرغوبة على نتائج التحليل الهيكلي أو تطور السلالات. كما تعتمد دقة الحسابات على قوة الحوسبة المتاحة، مما يجعل بعض الخوارزميات الأكثر شمولاً ودقة غير قابلة للتطبيق بسبب بطء معدل معالجتها. لذلك، غالباً ما يتم اللجوء إلى خوارزميات أسرع وأقل دقة، مما يفرض مفاضلة ضرورية بين الدقة والجدوى الحسابية، أو كما يُقال في أوساط البرمجة، بين الأناقة والبساطة في تصميم الخوارزميات. لذا، من المهم أخذ احتمالية حدوث أخطاء في برامج المعلوماتية الحيوية بعين الاعتبار.

الخوارزميات المستخدمة في المعلوماتية الحيوية [10-13]:

❖ **خوارزمية Smith-Waterman algorithm**

تعتمد خوارزمية Smith–Waterman على محاذاة التسلسل المحلي، حيث تهدف إلى تحديد المناطق المتشابهة بين سلسلتين من متواليات الحمض النووي أو البروتين، بدلاً من مقارنة التسلسل بأكمله. وتعد هذه الخوارزمية فعالة عند التعامل مع التتابعات المتباعدة أو التي تحتوي على مجالات متعددة، والتي قد تكون مختلفة في الأصل.

سنعتمد على هذه الخوارزمية في هذا البحث.

❖ **خوارزمية Needleman-Wunsch algorithm**

تُستخدم خوارزمية Needleman-Wunsch في المعلوماتية الحيوية لمحاذاة تسلسلات البروتين أو النوكليوتيدات. وتعتمد على تقسيم المشكلات الكبيرة، مثل محاذاة التسلسل الكامل، إلى مشكلات أصغر، ثم استخدام حلول هذه المشكلات الجزئية لإيجاد الحل الأمثل للمشكلة الأساسية. تُعرف أيضاً باسم خوارزمية المطابقة المثلى أو تقنية المحاذاة العامة، وتهدف إلى تحديد جميع المحاذاة الممكنة التي تحقق أعلى درجة تطابق.

٣,٥. الحوسبة التفرعية وبيئة الحوسبة الموزعة

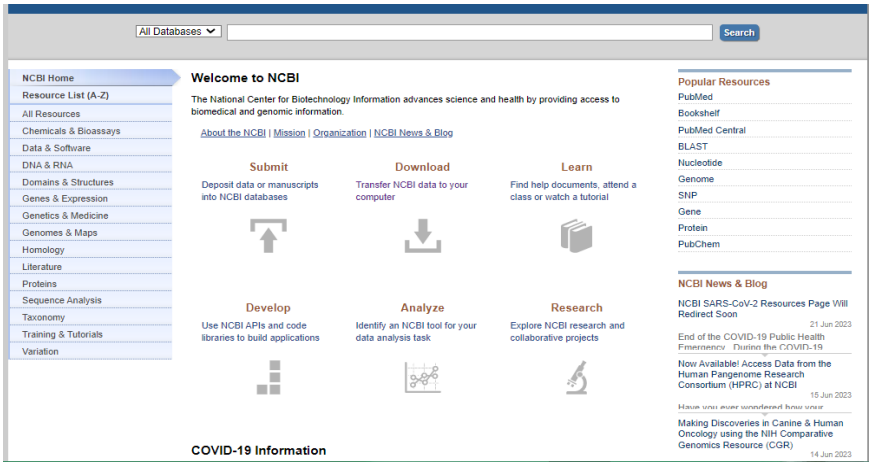
يُعد هذا المجال أحد فروع هندسة الحاسبات، ويختص بدراسة الأنظمة الموزعة (Distributed Systems)، وهي أنظمة برمجية تتكون من مجموعة من الحواسيب المتصلة عبر شبكة مشتركة. يتم التواصل والتفاعل بين هذه الأجهزة من خلال تمرير الرسائل (Passing Messages)، حيث تتعاون مكوناتها لتحقيق هدف مشترك. وتُعرف برامج الحاسوب التي تعمل ضمن هذه الأنظمة بالبرامج الموزعة.

في الحوسبة الموزعة، يتم تقسيم المشكلة إلى مشاكل صغيرة، ثم يتم توزيعها على الحواسيب ليتم حل هذه المشكلة كما هو موضح في الشكل(٣).

الشكل(٣) مثال على التحضير للعمل في البيئة الموزعة.

٣,٦. موقع NCBI

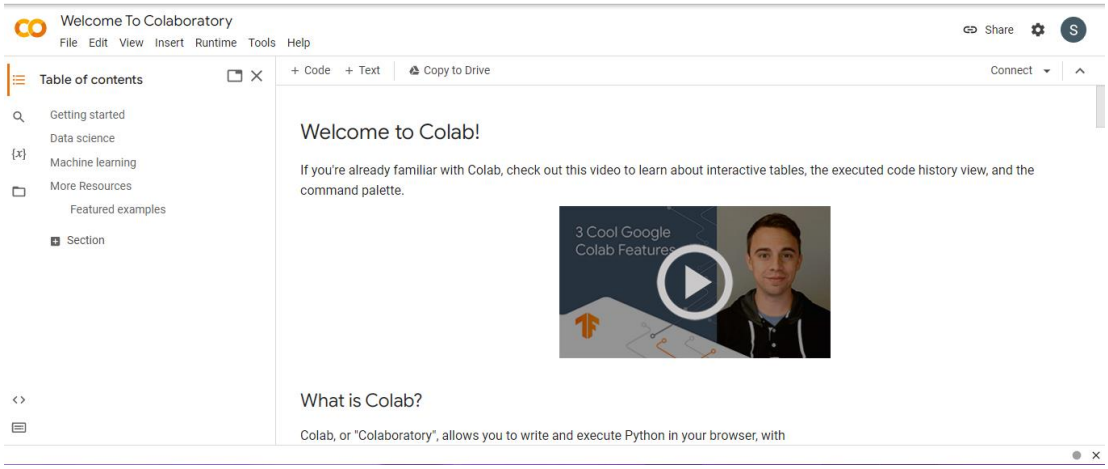
يعد مركز المعلوماتية الحيوية الوطني (NCBI - National Center for Biotechnology Information) جزءاً من المكتبة الوطنية للطب في الولايات المتحدة، التابعة لمعاهد الصحة العالمية. يضم المركز قاعدة بيانات ضخمة تشمل التسلسلات الجينية المخزنة في بنك الجينات، بالإضافة إلى فهرس للأبحاث والمقالات في الطب الحيوي. كما يوفر معلومات شاملة حول تسلسلات DNA وRNA والبروتين لمختلف الكائنات، وجميع هذه البيانات متاحة عبر الإنترنت[17]. يبين الشكل (٤) واجهة هذه البيئة.



الشكل(٤) واجهة موقع NCBI.

٣,٧. Google Colab

يُعد Google Colab نسخة سحابية مجانية من Jupyter Notebook يعمل على خوادم Google[11] Cloud، مما يتيح للمستخدمين الوصول إلى ميزات متقدمة مثل وحدات معالجة الرسومات (GPU)، والذاكرة الإضافية، والأجهزة الخلفية القوية. يوفر Colab بيئة متكاملة لتنفيذ جميع الوظائف التي يتم عادةً تنفيذها على Jupyter Notebook المثبّت محلياً، دون الحاجة إلى إعدادات معقدة. كما يوفر جميع الأدوات اللازمة لبدء المشاريع البرمجية في علم البيانات والتعلم العميق بسهولة وفعالية[12]، واجهة Google Colab تظهر كما في الشكل (5).



الشكل(5) واجهة موقع Google Colab.

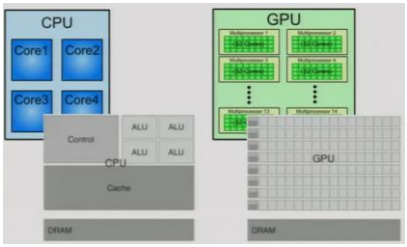
٣,٨. معالج GPU:

وحدة معالجة الرسوميات (Graphics Processing Unit - GPU) هي معالج متوازي مصمم لتسريع العمليات الحسابية المرتبطة بالرسوميات، خاصة في تنفيذ عمليات الفاصلة العائمة اللازمة لعرض الرسوم ثلاثية الأبعاد. وتتميز GPU بقدرتها الفائقة على التعامل مع المهام الحسابية المكثفة بشكل أسرع وأكثر كفاءة من وحدة المعالجة المركزية (CPU)، مما يجعلها ضرورية في العديد من المجالات مثل الألعاب، التصميم الغرافيكي، التعلم العميق، والحوسبة المتقدمة.

تم تصميم وحدات معالجة الرسومات لتسريع عرض الرسومات ثلاثية الأبعاد في الزمن الحقيقي، مثل تطبيقات الألعاب .لذا تزايدت الطلبات على معالجات رسومية أكثر سرعة خصوصاً بعد ظهور تقنيات العرض المتطورة مثل شاشات4K وظهور ألعاب الواقع الافتراضي.

مع التطور السريع في الذكاء الاصطناعي (AI) والتعلم الآلي (ML) وإنترنت الأشياء (IoT)[15]، ازدادت الحاجة إلى معالجة كميات هائلة من البيانات بكفاءة وسرعة عالية. ونظراً لعدم قدرة الحواسيب التقليدية على تنفيذ هذه المهام بفعالية، أصبح تحسين أداء الحوسبة ضرورة ملحّة. وقد تم تلبية هذه المتطلبات من خلال الحوسبة عالية الأداء (HPC)، التي تتيح تنفيذ مليارات العمليات الحسابية المعقدة بسرعة فائقة وبشكل متوازٍ، مما يعزز كفاءة التطبيقات الحديثة في مختلف المجالات[3].

تختلف نواة GPU بشكل جوهري عن نواة CPU من حيث التصميم والوظيفة. فبينما تتميز نواة CPU بمنطق تحكم معقد مصمم لتنفيذ البرامج التسلسلية بكفاءة، تمتلك نواة GPU بنية أبسط تركز على تشغيل المهام المتوازية بشكل فعال. ولهذا، تحتوي GPU على آلاف النوى الصغيرة التي توفر أداءاً عالياً في معالجة البيانات المتوازية، مما يجعلها أكثر كفاءة من CPU متعددة الأنوية (Multicore CPUs) في تنفيذ العمليات الحسابية الكثيفة. كما في الشكل (٦). توفر وحدة المعالجة المركزية متعددة الأنوية (Multicore CPU) ذاكرة تخزين مؤقتة كبيرة، وتنفذ تعليمات بطول x86 كاملة على كل نواة، مما يجعلها مثالية لمعالجة المهام التسلسلية بكفاءة. في المقابل، تم تصميم نوى وحدة معالجة الرسومات (GPU) لتكون أصغر حجماً وأكثر تركيزاً على حسابات النقطة العائمة لتعزيز الإنتاجية. وعند تنفيذ المهام، تُعالج الأجزاء المتسلسلة من حمل العمل على CPU، بينما تُنفَّذ الأجزاء كثيفة الحوسبة بالتوازي عبر آلاف النوى داخل GPU، مما يتيح أداءاً أعلى في العمليات الحسابية المكثفة.[3].



الشكل(٦) بنية معالج GPU و معالج CPU.

٤. خوارزمية Smith - Waterman

خوارزمية Smith-Waterman هي واحدة من أهم الخوارزميات المستخدمة في الاصطفاف المحلي، وقد طُورت عام ١٩٨١ على يد Temple F. Smith و Michael S. Waterman، وتعرف اختصاراً بـ SW. وتُستخدم على نطاق واسع في علم الأحياء الحاسوبي وتحليل التسلسلات الجزيئية. تعتمد هذه الخوارزمية على البرمجة الديناميكية في عملية الاصطفاف المحلي لمحاذاة أجزاء محددة من سلسلتين (DNA، RNA، أو البروتينات)، حيث يتم منح درجات موجبة للمحارف المتطابقة، وتُسند قيمة صفر لعدم التطابق، دون استخدام درجات سالبة. أما المسار الخلفي التتبعي (Tracing-Back)، فيتم وفق نهج مماثل للبرمجة الديناميكية[10].

تُستخدم خوارزمية Smith-Waterman لمحاذاة التسلسل المحلي، مما يساعد في العثور على مناطق التشابه بين سلسلتين من متواليات الحمض النووي أو البروتينات، دون الحاجة إلى مقارنة التسلسلات بالكامل. وتُعد هذه الطريقة مثالية عند التعامل مع تتابعات متباعدة أو تحتوي على مجالات متعددة ذات أصول مختلفة.

٤,١. آلية عمل الخوارزمية

يمكن شرح آلية عمل الخوارزمية على النحو الآتي:

١. يتم أولاً إنشاء مصفوفتين ثنائيتي البعد بحجم (N+1, M+1)، حيث M يمثل طول السلسلة الأولى، وN يمثل طول السلسلة الثانية. يُملأ الصف الأول بأحرف السلسلة الأولى، بينما يُملأ العمود الأول بأحرف السلسلة الثانية.

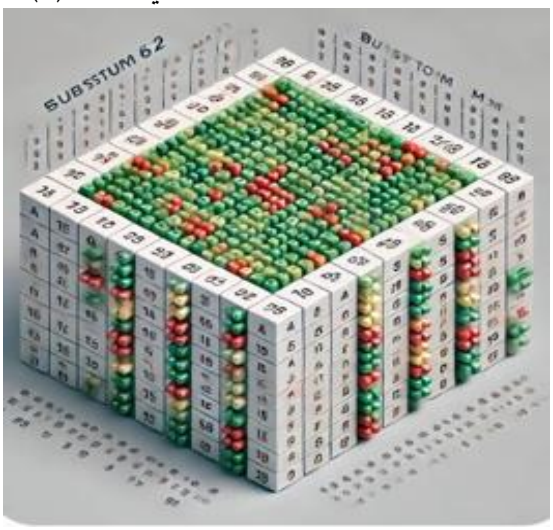
- المصفوفة الأولى: تُستخدم لحساب الدرجات، حيث يتم تحديد أفضل مسار ممكن للوصول إلى كل نقطة تصطف فيها أزواج الحروف المتقابلة.
- المصفوفة الثانية: وهي مصفوفة التتبع، وتساعد في اختيار أفضل مسار لاسترجاع الاصطفاف الأمثل. كما هو موضح في الشكل (٧).

	-	G	A	T	A
-	-	-	-	-	-
G	-	-	-	-	-
C	-	-	-	-	-
A	-	-	-	-	-
T	-	-	-	-	-

	-	G	A	T	A
-	0	0	0	0	0
G	0	0	0	0	0
C	0	0	0	0	0
A	0	0	0	0	0
T	0	0	0	0	0

الشكل(٧) مصفوفة الدرجات، مصفوفة التتبع.

- يبدأ مسار الاصطفاف وينتهي على امتداد القطر الرئيسي للمصفوفة.
- يتم الانطلاق من الموقع ذي الدرجة الأعلى، ثم يُتابع المسار بشكل قطري وإلى اليسار حتى الوصول إلى أول مربع يحتوي على القيمة صفر، مع إدراج الفجوات عند الحاجة.
- بهذه الطريقة، يتم استخراج جزء من الاصطفاف الأمثل الذي يحقق أعلى الدرجات.
- تُهيّأ المصفوفة بوضع القيمة صفر في السطر الأول والعمود الأول، مما يؤدي إلى تجاهل الأعمدة التي تحتوي على فجوات.
- يتم احتساب الدرجات باستخدام مصفوفات الاستبدال، مثل BLOSUM62، لضمان التقييم الأمثل للتطابقات والاختلافات بين التسلسلات[14]. المبينة في الشكل (٨).



الشكل(٨) مصفوفة الاستبدال BLOSUM62.

- في الاصطفاف المحلي باستخدام خوارزمية Smith-Waterman، يتم حساب درجة كل خلية بنفس الطريقة المستخدمة في الاصطفاف الشامل (Needleman-Wunsch)، ولكن مع تعديل مهم: إضافة القيمة ٠ لمنع القيم السالبة.
- معادلة حساب درجة كل خلية :

$$D(i,j) = max \begin{cases} 0 \\ diag = D(i-1,j-1) + s(x_i, y_j) & 1 < i \leq n \\ up = D(i-1,j) + g & 1 < j \leq m \\ left = D(i,j-1) + g \end{cases} \quad (1)$$

حيث:

- $s(x_i,y_j)$ هي درجة تطابق x_i مع y_j وتحتسب من أحد مصفوفات الاستبدال مثلBLOSUM62
- g هي قيمة الفجوة الخطية، أي g هي جزاء إضافة فجوة $(-, y_j)$ أو $(x_i, -)$
- $D(i\text{-}1\ ,\ j\text{-}1)$ هي قيمة الدرجة القطرية السابقة.
- $D(i\text{-}1\ ,\ j)$ قيمة الدرجة من الصف السابق.
- $D(i\ ,\ j\text{-}1)$ قيمة الدرجة من العمود السابق.

- تُبنى مصفوفة التتبع بشكل متزامن مع مصفوفة الدرجات، حيث يتم تحديثهما تدريجياً في كل خطوة من خطوات الحساب، مما يضمن توافقاً دقيقاً بين القيم المحسوبة ومسار المحاذاة.
- بعد اكتمال المصفوفات، يتم استخراج المحاذاة المثلى من مصفوفة التتبع، والتي تحدد المسار الأمثل وفقاً لأعلى الدرجات، مما يساعد في الحصول على أفضل تقابل بين التسلسلين قيد التحليل.

٤,٢. تحضير سلسلة البروتين

لأجل الحصول على سلسلة البروتين من موقع NCBI (National Center for Biotechnology Information)، يوجد ثلاث طرق يمكن تلخيصها على الشكل التالي:

١. البحث عن تسلسل البروتين يدوياً عبر NCBI:

يتم وفق الخطوات الآتية:

- الانتقال إلى موقع NCBI Protein.
- البحث عن البروتين المطلوب.
- اختيار التسلسل الصحيح.
- عرض وتحميل التسلسل.

٢. استخدام NCBI Entrez Direct للحصول على تسلسل البروتين برمجيا:

يتم ذلك في بيئة Linux أو عبر واجهة الأوامر، باستخدام الأمر التالي:

esearch -db protein -query "اسم البروتين أو رقم التعريف" | efetch -format fasta

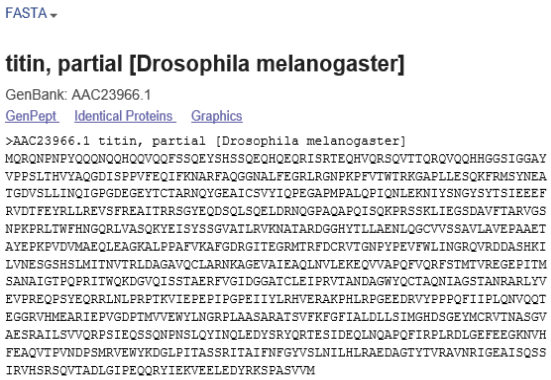
٣. استخدام API الخاص بـ NCBI:

يتم ذلك عن طريق جلب تسلسل البروتين باستخدام NCBI E-utilities API عبر رابط مباشر مثل:

https://www.ncbi.nlm.nih.gov/sviewer/viewer.fcgi?id=<Protein_ID>&db=protein&report=fasta

بحيث يتم استبدل <Protein_ID> برقم تعريف البروتين المطلوب.

البروتين الذي اعتمدنا عليه في دراستنا هو بروتين تيتين (Titin)، وهو أكبر بروتين معروف في جسم الإنسان، ويلعب دوراً حاسماً في عمل العضلات. يُعرف أيضاً باسم "كونيكتين" (Connectin)، وهو ضروري لوظيفة العضلات الهيكلية والقلبية، حيث يساعد في تنظيم مرونة العضلات أثناء الانقباض والانبساط، وبعبارة أخرى تينين هو العمود الفقري للمرونة العضلية في جسم الإنسان. بالنقر على FASTA نحصل على سلسلة البروتين كما هو موضح في الشكل (٩).


FASTA

titin, partial [Drosophila melanogaster]

GenBank: AAC23966.1

GenPept Identical Proteins Graphics

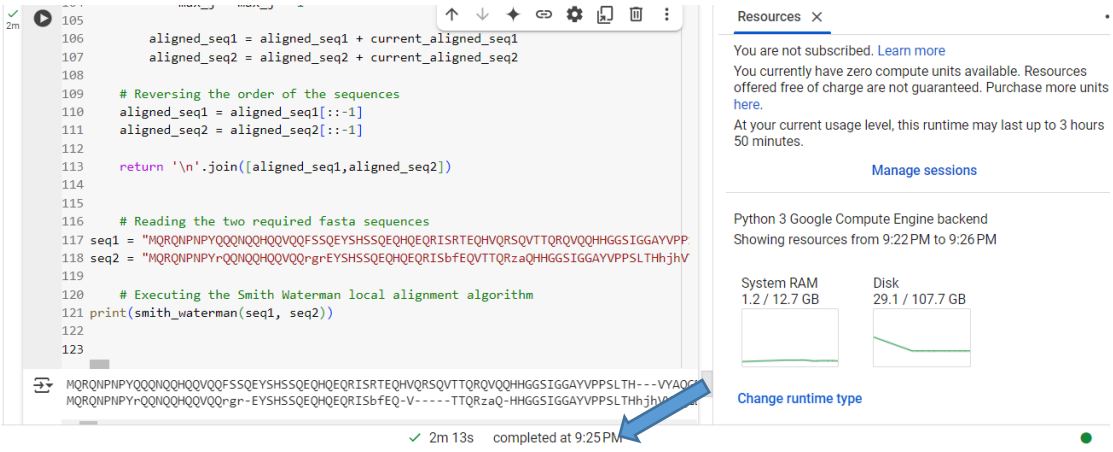
>AAC23966.1 titin, partial [Drosophila melanogaster]
MQRQNPNPYQQQNQQHQQVQQFSSQEYSHSSQEQHQEQRISRTEQHVQRSQVTTQRQVQQHHGGSIGGAY
VPPSLTHVYAQGDISPPVFEQIFKNARFAQGGNALFEGRLRGNPKPFVTWTRKGAPLLESQKFRMSYNEA
TGDVSLLINQIGPGDEGEYTCTARNQYGEAICSVYIQPEGAPMPALQPIQNLEKNIYSNGYSYTSIEEEF
RVDTFEYRLLREVSFREAITRRSGYEQDSQLSQELDRNQGPAQAPQISQKPRSSKLIEGSDAVFTARVGS
NPKPRLTWFHNGQRLVASQKYEISYSSGVATLRVKNATARDGGHYTLLAENLQGCVVSSAVLAVEPAAET
AYEPKPVDVMAEQLEAGKALPPAFVKAFGDRGITEGRMTRFDCRVTGNPYPEVFWLINGRQVRDDASHKI
LVNESGSHSLMITNVTRLDAGAVQCLARNKAGEVAIEAQLNVLEKEQVVAPQFVQRFSTMTVREGEPITM
SANAIGTPQPRITWQKDGVQISSTAERFVGIDGGATCLEIPRVTANDAGWYQCTAQNIAGSTANRARLYV
EVPREQPSYEQRRLNLPRPTKVIEPEPIPGPEIIYLRHVERAKPHLRPGEEDRVYPPPQFIIPLQNVQQT
EGGRVHMEARIEPVGDPTMVVEWYLNGRPLAASARATSVFKFGFIALDLLSIMGHDSGEYMCRVTNASGV
AESRAILSVVQRPSIEQSSQNPNSLQYINQLEDYSRYQRTESIDEQLNQAPQFIRPLRDLGEFEEGKNVH
FEAQVTPVNDPSMRVEWYKDGLPITASSRITAIFNFGYVSLNILHLRAEDAGTYTVRAVNRIGEAISQSS
IRVHSRSQVTADLGIPEQQRYIEKVEELEDYRKSPASVVM

الشكل(٩) سلسلة البروتين.

٤,٣. تنفيذ خوارزمية Smith-Waterman على معالج تسلسلي

تُعد خوارزمية Smith-Waterman من الخوارزميات التسلسلية المصممة للتشغيل على المعالجات التقليدية، حيث تعتمد على مبدأ البرمجة الديناميكية لمقارنة السلاسل الحيوية واستخراج المحاذاة المحلية الأكثر تطابقاً. في هذه المرحلة، تم تعديل الخوارزمية لإدراج سلاسل البروتين الخاصة بالدراسة، مما أتاح تحليلاً أكثر دقة للبيانات الحيوية.

تم تنفيذ الخوارزمية باستخدام Google Colab، حيث تم الاستفادة من وحدة GPU لتسريع العمليات الحسابية وتحسين أداء المحاذاة. بعد تشغيل الخوارزمية باستخدام البيانات المعدلة، ظهرت النتائج كما في الشكل(١٠):


```
105
106         aligned_seq1 = aligned_seq1 + current_aligned_seq1
107         aligned_seq2 = aligned_seq2 + current_aligned_seq2
108
109     # Reversing the order of the sequences
110     aligned_seq1 = aligned_seq1[::-1]
111     aligned_seq2 = aligned_seq2[::-1]
112
113     return '\n'.join([aligned_seq1,aligned_seq2])
114
115
116     # Reading the two required fasta sequences
117 seq1 = "MQRQNPNPYQQQNQQHQQVQQFSSQEYSHSSQEQHQEQRISRTEQHVQRSQVTTQRQVQQHHGGSIGGAYVPP
118 seq2 = "MQRQNPNPYrQQNQQHQQVQQrgrEYSHSSQEQHQEQRISbfEQVTTQRzaQHHGGSIGGAYVPPSLTHhjhV
119
120     # Executing the Smith Waterman local alignment algorithm
121 print(smith_waterman(seq1, seq2))
122
123
MQRQNPNPYQQQNQQHQQVQQFSSQEYSHSSQEQHQEQRISRTEQHVQRSQVTTQRQVQQHHGGSIGGAYVPPSLTH---VYAQG
MQRQNPNPYrQQNQQHQQVQQrgr-EYSHSSQEQHQEQRISbfEQ-V-----TTQRzaQ-HHGGSIGGAYVPPSLTHhjhV
```

Resources

You are not subscribed. Learn more

You currently have zero compute units available. Resources offered free of charge are not guaranteed. Purchase more units here.

At your current usage level, this runtime may last up to 3 hours 50 minutes.

Manage sessions

Python 3 Google Compute Engine backend

Showing resources from 9:22 PM to 9:26 PM

System RAM 1.2 / 12.7 GB

Disk 29.1 / 107.7 GB

Change runtime type

✓ 2m 13s completed at 9:25 PM

الشكل(١٠) نتائج تنفيذ الخوارزمية.

نلاحظ أن تنفيذ هذه الخوارزمية **استغرق زمن قدره 2m 13s (دقيقتين وثلاث عشرة ثانية).**

٤,٤. تطوير نسخة تفرعية من خوارزمية Smith-Waterman على معالج GPU

تطوير نسخة تفرعية من خوارزمية Smith-Waterman على معالج GPU يتطلب مراعاة التوازي الكثيف واستغلال البنية المعمارية للمعالجات الرسومية. نوضح فيما يلي المراحل الأساسية لهذا التطوير:

٤,٤,١ البحث والتحليل

استيعاب آلية عمل الخوارزمية الأصلية من خلال تحليل تعقيدها الحسابي ومتطلبات استهلاك الذاكرة لضمان كفاءة التنفيذ. وتحديد التحديات المحتملة عند نقلها إلى GPU، مثل قيود الوصول إلى الذاكرة العالمية والمشتركة، والتوازي على مستوى الكتل والنياسب. ثم دراسة التحسينات السابقة التي تم تطويرها باستخدام تقنيات مثل CUDA, OpenCL, HIP، مع التركيز على استراتيجيات تحسين الأداء، مثل تقليل التأخير في نقل البيانات والاستفادة من الذاكرة المشتركة وتقنيات إعادة استخدام القيم المحسوبة.

٤,٤,٢ تحديد أهداف التطوير

- هل الهدف هو تقليل زمن التنفيذ، تقليل استهلاك الذاكرة، أو تحقيق توازن بين الاثنين؟
- تحديد حجم التسلسلات البيولوجية المستهدفة والقيود على حجم الذاكرة في GPU.
- تحديد معمارية المعالج الرسومي (NVIDIA، AMD، أو Intel Arc) لضبط التنفيذ بناءً على قدراته.

٤,٤,٣ تصميم النسخة التفرعية للخوارزمية

قمنا بتطوير النسخة التفرعية من خوارزمية Smith-Waterman التسلسلية وفق الخطوات التالية:

١. استيراد مكتبة Threading لتمكين التنفيذ المتوازي.

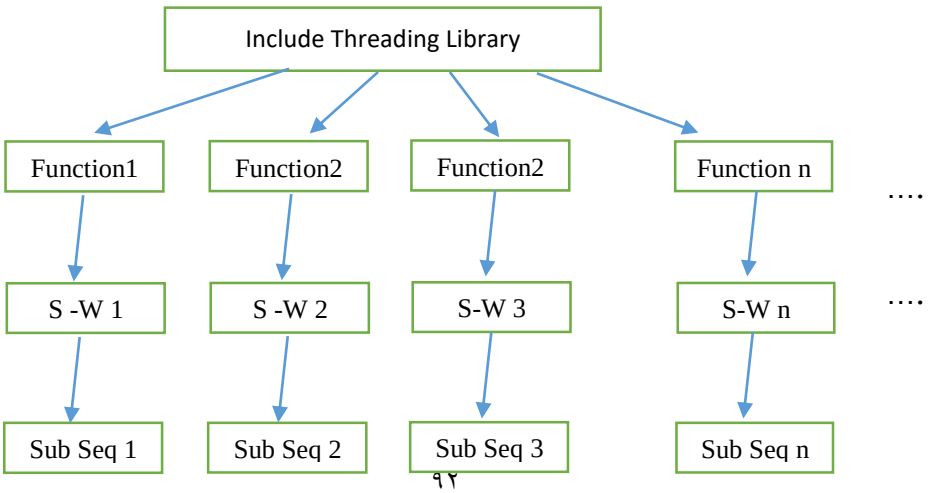
٢. إنشاء مجموعة من التوابع بعدد الأنوية أو الخيوط (Threads) التي سيتم التشغيل عليها.

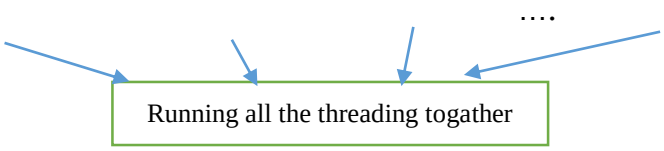
٣. تنفيذ خوارزمية Smith-Waterman داخل كل تابع بحيث يعمل كل Thread على معالجة جزء معين من البيانات.

٤. تقسيم السلسلة المدروسة إلى أجزاء أصغر وتوزيعها على النياسب Threads لضمان توزيع متوازن للحمل الحسابي.

٥. تشغيل جميع النياسب Threads بالتوازي لضمان الاستفادة القصوى من موارد المعالج وتقليل زمن التنفيذ.

ويمكن تلخيص هذه الخطوات وفق المخطط الصندوقي التالي(الشكل(١١)):





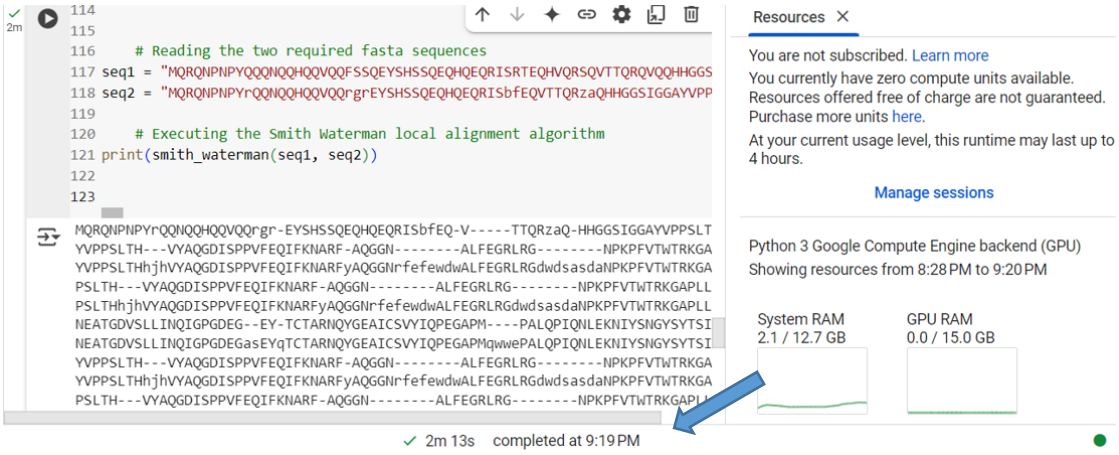
الشكل(١١) المخطط الصندوقي لتطوير خوارزمية Smith-Waterman التسلسلية إلى خوارزمية تفرعية

٤,٥. تنفيذ خوارزمية Smith-Waterman على معالج تفرعي

بعد الانتهاء من تطوير النسخة التفرعية للخوارزمية، سيتم اختبارها على معالج GPU داخل Google Colab باستخدام مجموعة من الاختبارات التقييمية لقياس كفاءتها في التنفيذ التفرعي.

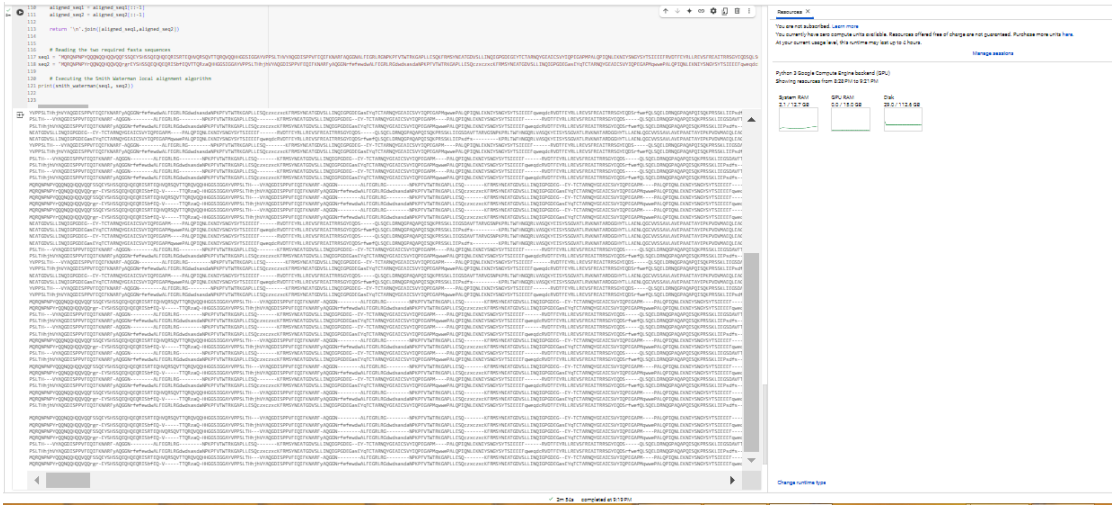
٤,٥,١ تنفيذ خوارزمية Smith-Waterman التسلسلية على معالج تفرعي Nothreads

تم إعداد بيئة العمل في Google Colab لتفعيل استخدام معالجات GPU في العمليات الحسابية، ثم قمنا بإدخال خوارزمية Smith-Waterman التسلسلية وإجراء المحاكاة، فظهرت النتائج كما في الشكل(١٢):



الشكل(١٢) نتائج تنفيذ الخوارزمية

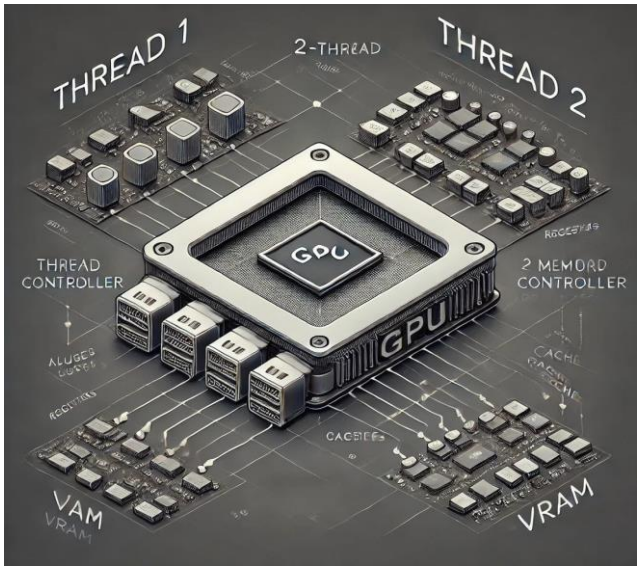
نلاحظ أن تنفيذ هذه الخوارزمية **استغرق زمن قدره 2m 13s (دقيقتين وثلاث عشرة ثانية)**، وهو نفس الزمن المستغرق عند تنفيذ هذه الخوارزمية على معالج تسلسلي CPU، مع ملاحظة أنه عند تنفيذ هذه الخوارزمية في بيئة تفرعية، قام معالج GPU بتقسيم حمل المعالجة إلى عدة Threads تعمل بشكل مستقل ومتوازٍ. ومع ذلك، نظراً لأن الخوارزمية كُتبت بطريقة تسلسلية، لم يكن بالإمكان تنفيذ العمليات بالتوازي، مما أدى إلى معالجة كل Thread بشكل متتابع، بدلاً من توزيع المهام على الأنوية المتعددة للمعالج الرسومي بكفاءة. الشكل(١٣) يظهر العدد الضخم من النياسب التي تم توزيع الحمل عليها:



الشكل(١٣) نتائج تنفيذ الخوارزمية

٤,٥,٢ تنفيذ خوارزمية Smith-Waterman على معالج تفرعي باستخدام نيسبين متوازيين tow threads

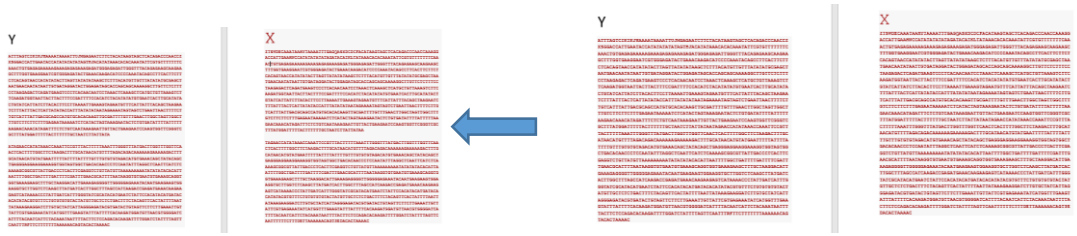
قمنا بإعداد بيئة العمل في Google Colab للاستفادة من معالجات GPU مع تشغيلها باستخدام مسارين (2Thread)بشكل متوازٍ وفق الشكل(١٤):



الشكل(١٤) معالج GPU يعمل على 2Thread

تمكنا من تحسين آلية الخوارزمية السابقة لتصبح قادرة على العمل باستخدام معالج الرسوميات (GPU) من خلال الاستفادة من ٢threads، عبر اتباع الخطوات التالية:

- تحضير البيانات للعمل باستخدام ٢threads عن طريق تقسيم كل سلسلة من السلسلتين السابقتين. (الشكل (١٥)).



الشكل(١٥) تحضير السلاسل.

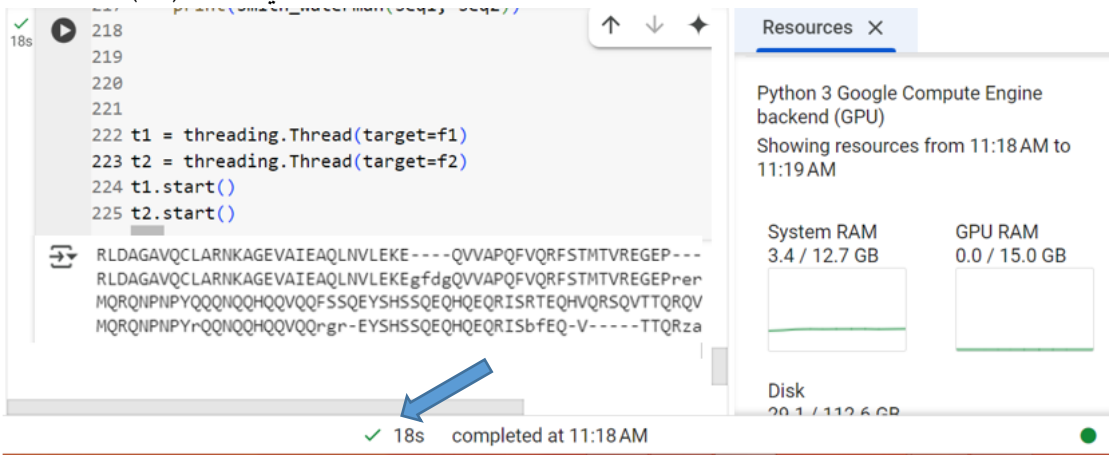
- تم تضمين مكتبة Threading في الكود:

```
import numpy as np
import threading
```

- ثم إعطاء الأمر لتنفيذ التابعين بشكل متوازي:

```
t1 = threading.Thread(target=f1)
t2 = threading.Thread(target=f2)
t1.start()
t2.start()
```

- إدخال السلاسل بعد التعديل وتنفيذ الخوارزمية، ظهرت النتيجة كما في الشكل (١٦).

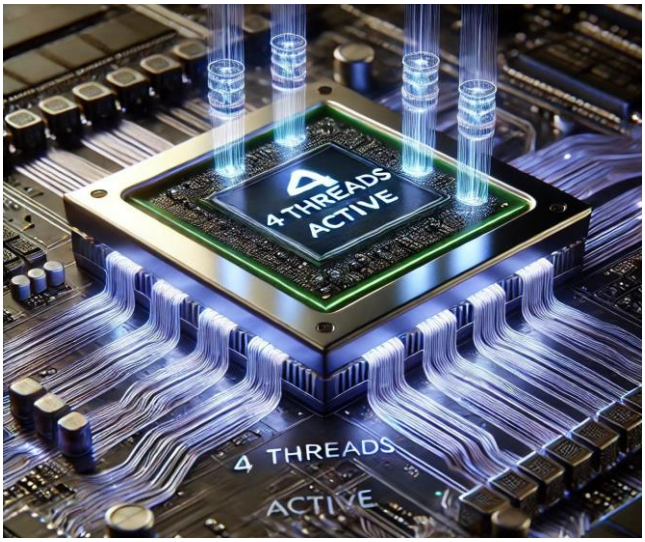


الشكل(١٦) نتائج تنفيذ الخوارزمية.

نلاحظ أن تنفيذ هذه الخوارزمية **استغرق زمن قدره 18s**، أي ما يعادل $1/7$ من زمن التنفيذ للخوارزمية السابقة، أما بالنسبة للنتائج التي ظهرت فهي مختلفة عن نتائج الخوارزمية السابقة.

٤,٥,٣ تنفيذ خوارزمية Smith-Waterman على معالج تفرعي باستخدام ٤ نياسب متوازية 4 threads

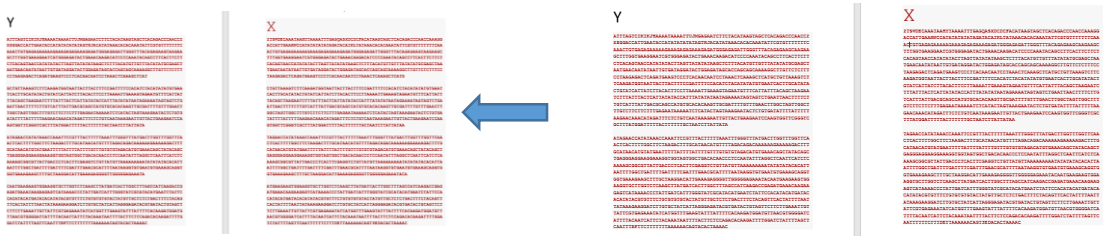
قمنا بإعداد بيئة العمل في Google Colab للاستفادة من معالجات GPU مع تشغيلها باستخدام 4 نياسب (4Thread)بشكل متوازٍ وفق الشكل(١٧):



الشكل(١٧) معالج GPU يعمل على 4Thread

تمكنا من تحسين آلية الخوارزمية السابقة لتصبح قادرة على العمل باستخدام معالج الرسوميات (GPU) من خلال الاستفادة من threads٤، عبر اتباع الخطوات التالية:

- تحضير البيانات للعمل باستخدام 4threads عن طريق تقسيم كل سلسلة من السلسلتين السابقتين. (الشكل (١٨)).



الشكل(١٨) تحضير السلاسل.

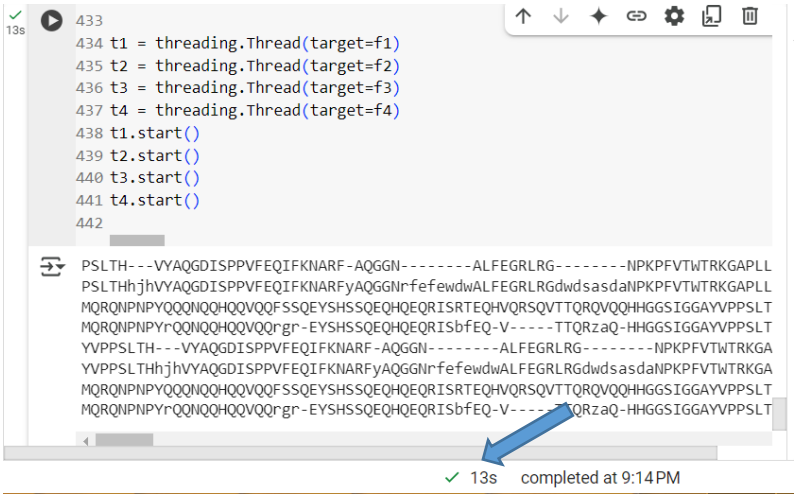
- تم تضمين مكتبة Threading في الكود:

```
4 import numpy as np
5 import threading
```

- ثم إعطاء الأمر لعمل التوابع الأربعة بشكل متوازي

```
440 t1 = threading.Thread(target=f1)
441 t2 = threading.Thread(target=f2)
442 t3 = threading.Thread(target=f3)
443 t4 = threading.Thread(target=f4)
444 t1.start()
445 t2.start()
446 t3.start()
447 t4.start()
```

- إدخال السلاسل بعد التعديل وتنفيذ الخوارزمية فظهرت النتيجة كما في الشكل (١٩).



الشكل(١٩) نتائج تنفيذ الخوارزمية.

نلاحظ أن تنفيذ هذه الخوارزمية استغرق **زمن قدره 13s**، أي ما يعادل $^2/_3$ من زمن التنفيذ للخوارزمية المنفذة على 2thread تقريباً، أما بالنسبة للنتائج التي ظهرت فهي مختلفة عن الخوارزميتين السابقتين.

الجدول (١) يوضح مقارنة حول أهم النتائج التي توصلنا إليها من تنفيذ خوارزمية S-W وفق الخيارات الثلاث السابقة.

الجدول (١) يوضح مقارنة حول أهم النتائج التي توصلنا إليها من تنفيذ خوارزمية SW .

خوارزمية Smith-Waterman	1 Thread	2 Thread	4 Thread
زمن التنفيذ	2m 13s	18 s	13 s
نوع المعالج المستخدم	معالجCPU/معالج GPU	معالج GPU/ معالج بنواتين	معالجGPU / معالج بأربع أنوية
الأداء	أداء ضعيف	أداء متوسط	أداء ممتاز وسريع
نتيجة التنفيذ	مختلفة عن الخوارزميتين	مختلفة عن الخوارزميتين	مختلفة عن الخوارزميتين
الكلفة	في هذه الخوارزمية، تم استخدام معالج واحد فقط، مما يعني أن الربح كان في تحسين العتاد الصلب، ولكن زمن التنفيذ كان ضعيفًا للغاية.	في هذه الخوارزمية، كان زمن التنفيذ والعتاد الصلب المستخدم مقبولين عند مقارنتهما بالتنفيذ التسلسلي.	في هذه الخوارزمية، كان الربح في زمن التنفيذ، ولكن على حساب العتاد الصلب المستخدم.

من الجدول(١) نلاحظ أن تنفيذ الخوارزمية على البيئة التفرعية باستخدام 2Thread خفَّض زمن التنفيذ بنسبة ٨٦%، كما أن تنفيذها على البيئة التفرعية باستخدام 4Thread خفَّض زمن التنفيذ بنسبة ٩٠%، لكن هذا الزمن نحصل عليه على حساب الزيادة الكبيرة والمكلفة في حجم العتاد الصلب، لذلك لابد من الأخذ بعين الاعتبار حجم العتاد الصلب المستخدم، فبالمقارنة نجد أن تنفيذ الخوارزمية على 2Thread هو الخيار الأفضل بما يضمن لنا أفضل زمن تنفيذ مع أقل حجم للعتاد الصلب.

بالإضافة إلى ذلك فإن تنفيذ خوارزمية Smith-Waterman على البيئة التفرعية نتج عنه تشوه في النتائج، وكلما تم تفريع الخوارزمية على معالجات أكثر كلما حصلنا على تشوه أكبر في النتائج، وبالتالي تنفيذ هذه الخوارزمية على البيئة التفرعية غير فعال ولا ينصح به.

سبب الحصول على تشوه في النتائج، أن مربعات المصفوفة يتم ملئها بالاعتماد على قيم المربعات المجاورة لها(المربع المجاور الأعلى، والمربع المجاور على اليمين، والمربع المجاور القطري)، وبما أن مسار التتبع الخلفي في هذه الخوارزمية يبدأ من الموقع الأعلى درجة ويستمر قطرياً وإلى اليسار حتى يصل إلى المربع الذي يحوي القيمة صفر، وبالتالي فإن عملية تقطيع السلاسل لتوزيعها على الحواسيب في البيئة الموزعة سيؤدي إلى تغير في قيم المربعات واختلاف مربع انطلاق مسار التتبع الخلفي وأيضاً اختلاف موضع توقفه، والذي يلحق به اختلاف في مجالات المقارنة وتغير في أبعاد وأشكال نتائج السلاسل المقارنة،(الشكل(٢٠)، الشكل(٢١)).

	gap	H	G	W	A	G
gap	0	0	0	0	0	0
P	0	0	0	0	0	0
H	0	8	0	0	0	0
S	0	0	8	0	1	0
W	0	0	0	19	11	3
G	0	0	6	11	19	17

	gap	H	G	W	A	G
gap	0	0	0	0	0	0
P	0	0	0	0	0	0
H	0	8	0	0	0	0
S	0	0	8	0	1	0
W	0	0	0	19	11	3
G	0	0	6	11	19	17

الشكل(٢٠) نتائج تنفيذ خوارزمية S-W دون تقسيم السلاسل

أفضل تقابل محلي هو:

HSWG

HGWA

	gap	H	G	W
gap	0	0	0	0
P	0	0	0	0
H	0	8	0	0
S	0	0	8	0

	gap	A	G
gap	0	0	0
W	0	0	10
G	0	8	0

الشكل(٢١) نتائج تنفيذ خوارزمية S-W بعد تقسيم السلاسل

أفضل تقابل محلي بعد التقسيم هو:

HS W

HG G

وبمقارنة النتائج السابقة نلاحظ وجود تشوه ظاهر في الخرج.

عند مقارنة هذه النتائج بما توصل إليه الباحثون في المرجع [١٨] من خلال تنفيذ خوارزمية Needleman-Wunsch ضمن بيئة تفرعية، تبيّن أن النتائج جاءت على النحو الآتي:

الجدول (٢) أهم ما تم التوصل إليه من تنفيذ خوارزمية NW في البيئة التفرعية

خوارزمية Needleman–Wunsch	زمن التنفيذ	عدد المعالجات	نتيجة التنفيذ
NO Thread	20 s	معالج واحد	مطابقة للخوارزميتين
2 Thread	5 s	معالجين اثنين	مطابقة للخوارزميتين
4 Thread	1.52 s	أربع معالجات	مطابقة للخوارزميتين

من خلال مقارنة الجدولين (١) و(٢)، يتبيّن أن خوارزمية Smith-Waterman تحقق سرعة أعلى في التنفيذ عند العمل ضمن بيئة تفرعية، متفوقة بذلك على خوارزمية Needleman-Wunsch من حيث الأداء الزمني. غير أن هذه الزيادة في السرعة تأتي على حساب دقة النتائج، إذ أدّى التنفيذ التفرعي لخوارزمية Smith–Waterman إلى حدوث تشوه في المخرجات، في حين حافظت خوارزمية Needleman-Wunsch على نتائجها الدقيقة دون أي تشويه.

٥. الاستنتاجات والتوصيات

من خلال تقييم أداء خوارزمية Smith-Waterman تم التوصل الى الاستنتاجات الآتية:

١- الزمن المستغرق لتنفيذ الخوارزمية التسلسلية على معالج CPU هو نفس الزمن المستغرق عند تنفيذ هذه الخوارزمية على معالج GPU.

٢- عند تنفيذ الخوارزمية التسلسلية في بيئة تفرعية، قام معالج GPU بتقسيم حمل المعالجة إلى عدة Threads تعمل بشكل مستقل ومتوازٍ. ولكن نظراً لأن الخوارزمية كُتبت بطريقة تسلسلية، لم يكن بالإمكان تنفيذ العمليات بالتوازي، مما أدى إلى معالجة كل Thread بشكل متتابع، بدلاً من توزيع المهام على الأنوية المتعددة للمعالج الرسومي بكفاءة.

٣- يمكن إضافة أي سلسلة إلى الخوارزمية بغض النظر عن حجمها.

٤- تم تقليص زمن تنفيذ الخوارزمية الثانية ليصل إلى $1/7$ من الزمن الذي استغرقته الخوارزمية الأولى.

٥- كما تم تقليص زمن التنفيذ في الخوارزمية الثالثة بمقدار $2/3$ من الزمن الذي استغرقته الخوارزمية الثانية.

٦- أظهرت البيئة الموزعة أداءً أسرع بكثير في تنفيذ الخوارزمية مقارنة مع البيئة التسلسلية.

٧- أدى تنفيذ خوارزمية Smith-Waterman في البيئة الموزعة إلى تشوه في النتائج.

يمكننا تطوير بحثنا باتجاه إجراء اختبار للخوارزمية السابقة بسلاسل مختلفة الطول وتنفيذ المهمة عدة مرات ومراقبة زمن التنفيذ، لمعرفة أثر تغير طول السلسلة على مدى فعالية الخوارزمية للتوصل الى أفضل أداء للخوارزمية من حيث طول السلسلة المستخدم وزمن التنفيذ ودقة النتائج وحجم الهاردوير.

كما يمكن زيادة تفريع الخوارزمية الى 8Thread و16Thread و دراسة تأثيرها على فاعلية الخوارزمية.

يمكن أيضاً الاستفادة من التحسينات التي أجريناها على الخوارزمية التسلسلية لجعلها متوافقة مع البيئة التفرعية، وتطبيق هذه التطويرات على خوارزميات أخرى في مجال المعلوماتية الحيوية.

المراجع العلمية

[1] Miguel Rocha, Pedro G. Ferreira, "Bioinformatics Algorithms", ACADIMIC PRESS, An imprint of Elsevier, 2018.

[2] K. G. Srinivasa et al; " Statistical Modelling and Machine Learning Principles for Bioinformatics Techniques, Tools, and Applications ", Springer, ISSN 2524-7573 (electronic), 2020.

[3] Dr. Hasan Albustani, Eng. Noor khder, ”Analysis of the training time of convolutional neural networks on high-performance processors”, Tartous University Journal for Research and Scientific Studies - engineering Sciences Series, 2022.

[4] Masahito Ohue et al; "High-Performance Cloud Computing for Exhaustive Protein–Protein Docking", Tokyo Institute of Technology, Japan, (2020).

[5] You Zou et al; "Parallel computing for genome sequence processing", OXFORD, Briefings in Bioinformatics,2021.

[6] Jamie J Alnasir et al; "The Application of Hadoop in Structural Bioinformatics",NLM, Briefings in Bioinformatics,2020.

[7] Ahmed Alzohairy;"Introduction to bioinformatics and genomics", Zagazig University,2013.

[8] Marketa Zvelebil, Jeremy O. Baum; "Understanding Bioinformatics", Garland Science, Taylor & Francis Group, LLC, 2008.

[9] P.M. Selzer et al; "Applied Bioinformatics", Springer, Verlag Berlin Heidelberg, 2008.

[10] Miguel Rocha, Pedro G. Ferreira; "Bioinformatics Algorithms Design and Implementation in Python", ACADEMIC PRESS.

[11] Aisling O’Driscoll, et al ;" 'Big Data', Hadoop and Cloud Computing in Genomics", Journal of Biomedical Informatics 46 (2013) 774–781.

[12] Dr. Poornima Naik, et al ;"Conceptualizing Python in Google COLAB ",2022.

[13] FN Muhamad, et al; " Performance Analysis Of Needleman-Wunsch Algorithm (Global) And Smith-Waterman Algorithm (Local) In Reducing Search Space And Time For Dna Sequence Alignment", 1st International Conference on Green and Sustainable Computing (ICoGeS) 2017, IOP Publishing, Series: Journal of Physics: Conf. Series 1019 (2018) 012085.

[14] STEVEN HENIKOFF, et al ;"Amino acid substitution matrices from protein blocks", Proc. Natl. Acad. Sci. USA, Vol. 89, pp. 10915-10919, November 1992, Biochemistry.

[15] Jamal Bzai, et al ;" Machine Learning-Enabled Internet of Things (IoT): Data, Applications, and Industry Perspective ", MDPI ,2022.

[16] Paul Hodor, et al ;" cl-dash: rapid configuration and deployment of Hadoop clusters for bioinformatics research in the cloud ", OXFORD, 2015.

[17] Eric W Sayers, et al ;" Database resources of the National Center for Biotechnology Information in 2025", OXFORD, 2024.

[18] Dr. Hasan Albustani, Eng. Sinan Shash, ”Implementation of bioinformatics algorithms to study protein sequences using parallel computing and cloud computing platforms”, Tartous University Journal for Research and Scientific Studies - engineering Sciences Series, 2023.