ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

TẠ VĂN NHÂN

ÁP DỤNG PHƯƠNG PHÁP

DÓNG HÀNG TRÌNH TỰ

CHO BÀI TOÁN DỰ ĐOÁN BIẾN THỂ GEN

Chuyên ngành: Khoa học dữ liệu

Mã số: 8904468.01QTD

LUẬN VĂN THẠC SỸ KHOA HỌC

NGƯỜI HƯỚNG DẪN KHOA HỌC
PGS.TS. NGUYỄN THỊ HỒNG MINH

Hà Nội - Năm 2021

MỞ ĐẦU

Giải trình tự DNA đang ngày càng trở nên nhanh chóng và kinh tế. Tuy nhiên, để ráp các trình tự thu được dựa trên bộ gen tham chiếu và tìm kiếm các biến thể, chúng ta cần có những hệ thống đủ mạnh để xử lý và phân tích dữ liệu. Phương pháp dóng hàng trình tự là một giải pháp hữu hiệu cho vấn đề này. Đã có nhiều kết quả nghiên cứu liên quan tới phương pháp cũng như phát triển công cụ dóng hàng trình tự được công bố. Tuy nhiên vấn đề về thời gian thực hiện, mức độ chính xác và phạm vi áp dụng của các kĩ thuật dóng hàng vẫn còn là những chủ đề cần được phát triển.

Mục đích của luận văn là nghiên cứu sâu kĩ thuật dóng hàng trình tự, đề xuất cải tiến để tăng hiệu quả về thời gian của thuật toán cũng như khả năng triển khai trên các hạ tầng tính toán phổ dụng. Đồng thời áp dụng thuật toán để khám phá trong phạm vi rộng hơn các biến thể gen so với một số nghiên cứu trước đây, khám phá mức độ ảnh hưởng của các biến thể đến chức năng của Protein.

Cụ thể trong nghiên cứu này, chúng tôi phát triển thuật toán dóng hàng dựa trên chuyển dạng Burrows-Wheeler và thuật toán Smith-Waterman. Trong đó, các mã giả được viết chi tiết để có thể triển khai bằng các ngôn ngữ lập trình khác nhau. Chúng tôi sử dụng ngôn ngữ Go với kỹ thuật song song và đồng thời để triển khai thuật toán dóng hàng trình tự dựa trên chuyển dạng Burrows-Wheeler, các chương trình được triển khai có thể chạy trên các hệ thống tính toán hiệu năng cao nhiều bộ xử lí, và cũng có thể chạy trên các máy tính cá nhân với khả năng tận dụng tất cả các logic proccessor của bộ xử lí. Kết quả thực nghiệm thuật toán bằng chương trình của chúng tôi được so sánh với kết quả nhận được từ công cụ BWA-MEM nhằm kiểm nghiệm tính chính xác của thuật toán mà chúng tôi phát triển. Đồng thời, việc thử nghiệm này cũng giúp hiểu rõ hơn về các tham số cho phù hợp với dữ liệu để sử dụng thuận lợi các công cụ dóng hàng trên các hệ thống đã có.

Trong chương 1, luận văn giới thiệu một số kiến thức cơ sở về sinh học phân tử, tin sinh học, các công nghệ giải trình tự. Từ những kiến thức cơ sở đó, những nghiên cứu sâu về phương pháp dóng hàng trình tự và những đề xuất cải tiến được trình bày chi tiết trong chương 2, bao gồm cả phần phương pháp và phần

thực nghiệm. Cuối cùng, trong chương 3 trình bày những kết quả áp dụng các phương pháp và công cụ dóng hàng để tìm biến thể gen của bệnh tâm thần phân liệt (Schizophrenia), một hội chứng rối loạn tâm thần nghiêm trọng có liên quan đến nhiều gen với yếu tố di truyền cao. Dữ liệu được tiền xử lý và khớp với bộ gen tham chiếu sử dụng thuật toán dựa trên chuyển dạng Burrows-Wheeler. Sau đó, thuật toán dóng hàng Smith-Waterman sắp xếp lại các Haplotype ở một số vùng hoạt động giúp kết quả dóng hàng ban đầu chính xác hơn. Các quá trình được triển khai trên nền tảng Galaxy và máy chủ Linux 64CPUs. Kết quả những biến thể tìm được trên các gen sẽ được so sánh với một số kết quả nghiên cứu của một số nhà khoa học và tổ chức đã được công bố.

Lời cảm ơn

Trước hết, tôi xin được tỏ lòng biết ơn và gửi lời cám ơn chân thành đến PGS.TS. Nguyễn Thị Hồng Minh, người trực tiếp hướng dẫn luận văn, đã tận tình chỉ bảo và định hướng giúp tôi tìm ra hướng nghiên cứu, tiếp cận thực tế, tìm kiếm tài liệu, xử lý và phân tích số liệu, giải quyết vấn đề. Tôi cũng xin được gửi lời cảm ơn đến NCS. Nguyễn Hà Linh (KU Leuven) đã giúp giải đáp cho tôi một số vấn đề chuyên môn về Tin sinh học, trong đó có một phần nội dung đã được đăng trên kỉ yếu Hội thảo quốc gia năm 2020.

Ngoài ra, trong quá trình học tập, nghiên cứu và thực hiện đề tài tôi còn nhận được rất nhiều sự quan tâm, góp ý và hỗ trợ quý báu. Tôi xin bày tỏ lòng biết ơn sâu sắc đến Quý thầy cô giảng viên Khoa Toán-Cơ-Tin học, trường Đại học Khoa học tự nhiên, Đại học Quốc gia Hà Nội đã tận tình truyền đạt những kiến thức chuyên môn sâu và rộng trong suốt quá trình tôi học tập tại trường.

Lời cảm ơn xin được gửi tới bạn bè cùng lớp thạc sĩ Khoa học dữ liệu khóa 1 đã luôn chia sẻ, trao đổi kiến thức, và thông tin đến tôi những điều cần thiết.

Xin chân thành cảm ơn các thầy cô, các kĩ thuật viên Trung tâm Động lực Thủy khí Môi trường, Trường Đại học Khoa học Tự nhiên đã giúp đỡ, tạo điều kiện để tôi có thể sử dụng hệ thống máy chủ của Trung tâm.

Tôi cũng xin gửi lời cảm ơn đến Edoardo Giacopuzzi và các cộng sự đã công khai trên NCBI dữ liệu của 7 mẫu bệnh tâm thần phân liệt mà tôi sử dụng trong phần thực nghiệm của đề tài.

Cuối cùng, nhưng trên tất cả là sự biết ơn tới Gia đình, Bố, Mẹ, Vợ, Em trai và các Con yêu thương luôn đồng hành, động viên và tạo điều kiện về mọi mặt để tôi được tham gia khóa đào tạo và hoàn thành nghiên cứu này.

Học Viên
Tạ Văn Nhân

Mục lục

Danh mục các chữ viết tắt

Từ	Tiếng Anh	Tiếng Việt
BWA	Burrows-Wheeler aligner	Dóng hàng Burrows-Wheeler
BWT	Burrows-Wheeler transform	Chuyển dạng Burrows-Wheeler
DNA	Deoxyribonucleic acid	Axit deoxyribonucleic
DP	Depth of coverage	Độ sâu bao phủ
LCS	The longest common subsequence	Chuỗi con chung dài nhất
RNA	Ribonucleic acid	Axit ribonucleic
mRNA	Messenger RNA	RNA thông tin
NGS	Next genration sequencing	Giải trình tự thế hệ tiếp theo
SNP	Single nucleotide polymorphism	Đa hình đơn Nucleotide
INDEL	Insertion or deletion of bases	Thêm hoặc xóa các bazơ
SA	Suffix arrays	Mảng hậu tố
SRA	Sequence read archive	Lưu trữ trình tự
SWA	Smith-Waterman aligner	Dóng hàng Smith-Waterman
PCR	Polymerase chain reaction	Phản ứng chuỗi Polymerase
TSP	Targeted sequencing panels	Giải trình tự nhắm mục tiêu
UTR	Untranslated region	Vùng không dịch mã
WES	Whole - exome sequencing	Giải trình tự exome
WGS	Whole - genome sequencing	Giải trình tự toàn hệ gen

Danh sách hình vẽ

Danh sách bảng

CHƯƠNG 1

KIẾN THỨC CƠ SỞ

Trái đất của chúng ta được hình thành từ cách đây khoảng 4.5 tỷ năm, nó vẫn là một hành tinh nóng bỏng cho đến 4 tỷ năm trước mà không một sinh vật nào có thể tồn tại [Haz]. Thật may mắn, trái đất dần hạ nhiệt, đó là điều kiện cần để sự sống đầu tiên xuất hiện cách đây 3.8 tỷ năm [Coo]. Sau đó, các tế bào giống như vi khuẩn đơn giản được gọi là tế bào nhân sơ (prokaryotic) và các tế bào có nhân (nucleus) được gọi là tế bào nhân thực (eukaryotic) xuất hiện. Hai loại tế bào này đã chứa một phân tử mang theo thông tin di truyền tên là DNA. Rất lâu sau đó, các sinh vật đa bào như thực vật có cơ thể phức tạp mới xuất hiện. Đặc biệt, các bằng chứng khảo cổ học đã chứng minh cho sự tồn tại những sinh vật đầu tiên giống con người trên trái đất cách đây từ 6 đến 7 triệu năm [CF]. Trải qua hàng triệu năm tiếp theo, các sinh vật ấy đã tiến hóa thành con người hiện đại, đây cũng chính là đối tượng được nghiên cứu trong luận văn này.

1.1 Một số khái niệm về sinh học phân tử và di truyền

Ở thời điểm đầu của sinh học phân tử và di truyền học, các nhà sinh hóa tách một cơ thể sống, một tế bào thành các phần và nghiên cứu chúng một cách độc lập, do đó họ không thấy được sự tác động lẫn nhau giữa các thành phần. Ngược lại, các nhà di truyền học lại nghiên cứu các chức năng sinh học bằng cách loại bỏ một thành phần nào đó. Nói cách khác, họ nghiên cứu trên những cá thể đột biến mà thành phần khác biệt được quy định bởi một hoặc nhiều gen. Tuy nhiên, có rất nhiều gen trong hệ gen người, các nhà di truyền học gặp khó khăn trong việc xác định các gen đó là gì, chúng kết hợp với nhau như thế nào. Thành tựu lớn nhất của các nhà sinh hóa là phân tách được protein, còn của các nhà di truyền học là tìm ra được cấu trúc của DNA. Ngày nay, cả hai phương pháp nghiên cứu được kết hợp lại với nhau một cách thông minh để khám phá những bí mật của bộ gen con người.

1.1.1 Các phân tử của một tế bào

Một tế bào gồm có các phân tử nhỏ có chức năng mang theo năng lượng hoặc truyền tín hiệu như: adenosine triphosphate (ATP), epinephrine, chất dẫn truyền thần kinh (neurotransmitters), nước, đường, acid béo, amino acid, và đơn phân tử nucleotide. Các phân tử này có thể tồn tại độc lập hoặc liên kết với nhau tạo thành các đại phân tử. Hai đại phân tử được biết đến trong sinh học là protein và nucleic acid:

Protein là một chuỗi dài được tạo thành bởi liên kết từ hàng trăm đến hàng nghìn amino acid. Sự đa dạng của protein bắt nguồn từ các cách kết hợp khác nhau của 20 loại amino acid mà trình tự của chúng xác định cấu trúc 3 chiều độc đáo của mỗi protein. Các cấu trúc riêng của protein quy định các chức năng cụ thể của chúng như: kháng thể, enzyme, điều khiển, thành phần cấu trúc, vận chuyển, dự trữ [1].

Deoxyribonucleic acid (DNA) là vật chất di truyền ở người và hầu hết các sinh vật khác mang thông tin (dưới dạng bộ ba mã di truyền) về cách thức, thời gian, và vị trí để sản xuất mỗi loại protein. Gần như mọi tế bào trong cơ thể người đều có cùng một DNA mà thường nằm trong nhân tế bào (nơi nó được gọi là DNA nhân), nhưng một lượng nhỏ DNA cũng có thể được tìm thấy trong ty thể (nơi nó được gọi là DNA ty thể hoặc mtDNA) và lục lạp [2].

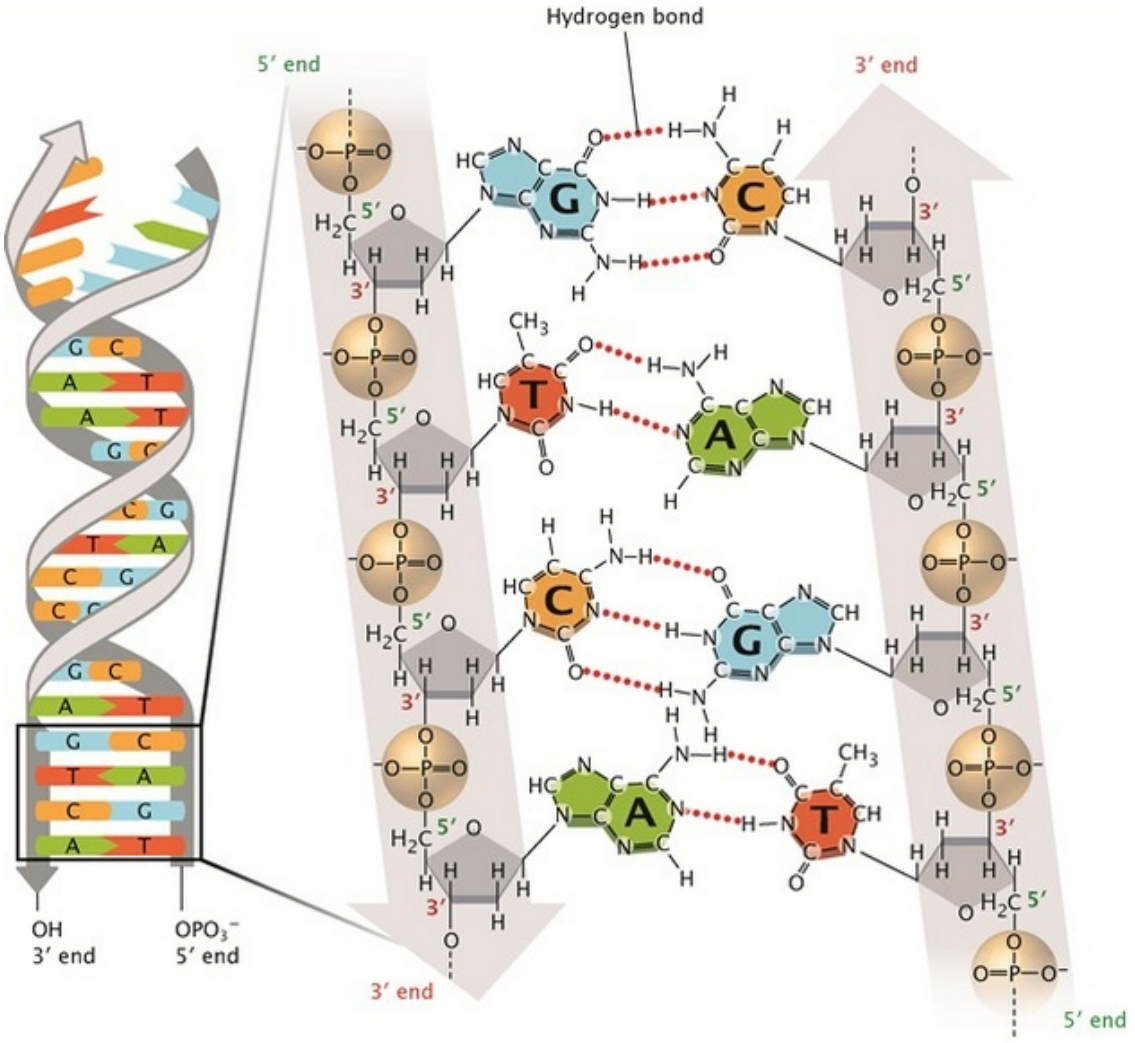
Cấu trúc ba chiều của DNA bao gồm hai sợi xoắn dài được cuộn quanh một trục tạo thành một chuỗi xoắn kép [WC53]. Các sợi DNA gồm có các đơn phân nucleotide được cấu tạo từ đường (deoxyribose), một nhóm phosphate, và một trong bốn bazơ hữu cơ mạch vòng là adenine (A), guanine (G), cytosine (C) và thymine (T). Chính vì vậy người ta còn gọi các nucleotide là các bazơ. Trên các phân tử đường, nhóm phosphate gắn với carbon $5'$ của nucleotide này liên kết cộng hóa trị với nhóm thế -OH gắn với carbon $3'$ của nucleotide tiếp theo tạo nên đường trục xoắn của một sợi. Mỗi chuỗi xoắn kép DNA lại được tạo thành bởi liên kết hidro giữa các cặp bazơ nhô ra khỏi đường trục xoắn: A của sợi này với T của sợi kia, G của sợi này với C của sợi kia. Những liên kết này làm cho hai sợi DNA trở thành dạng bổ sung ngược của nhau (Hình 1.1).

Gen là một đơn vị chức năng của DNA. Gen thường bao gồm hai phần: vùng mã hóa (coding region) xác định trình tự amino acid của protein, vùng điều hòa (regulatory region) kiểm soát thời gian và tế bào mà protein được tạo thành [Lod+07]. Hiện nay, các nhà khoa học đã tìm ra khoảng 21000 gen mã hóa protein chiếm 2% hệ gen con người, 98% còn lại là vùng điều hòa biểu hiện gen và những chức năng bí ẩn chưa được khám phá [Per+18].

Ribonucleic acid (RNA) là một phân tử polymer được liên kết bởi các nucleotide. RNA tương đối giống DNA về cấu trúc và cấu tạo, chúng chỉ khác

[1] https://ghr.nlm.nih.gov/primer/howgenswork/protein
[2] https://ghr.nlm.nih.gov/primer/basics/dna

nhau ở hai điểm: RNA có dạng sợi đơn gập vào chính nó trong khi DNA có dạng sợi xoắn kép; T trong DNA được thay thế bằng Uracil (U) trong RNA.



Hình 1.1: Chuỗi xoắn kép DNA gồm hai sợi xoắn dài được cuộn quanh một trục, sợi này là dạng bổ sung ngược của sợi kia. Trên các phân tử đường, nhóm phosphate gắn với carbon $5^{'}$ của nucleotide này liên kết cộng hóa trị với nhóm thế -OH gắn với carbon $3^{'}$ của nucleotide tiếp theo tạo nên đường trục xoắn của một sợi (màu xám). Hai sợi DNA lại được liên kết với nhau bởi các liên kết hidro giữa các cặp bazơ nhô ra khỏi đường trục xoắn: hai liên kết hidro nối T với A, ba liên kết hidrô nối G với C.[3]

[3] https://www.nature.com/scitable/topicpage/discovery-of-dna-structure-and-function-watson-397/

1.1.2 Luận thuyết trung tâm

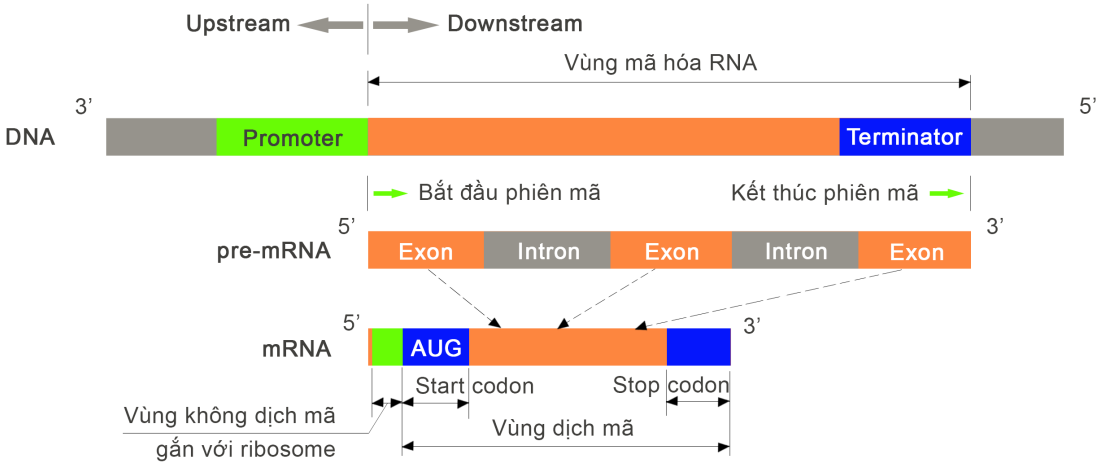
Luận thuyết trung tâm đưa ra các quy tắc chung cho việc chuyển giao thông tin giữa DNA, RNA và protein [Cri70]. Tế bào chuyển đổi thông tin được mã hóa trong DNA để tạo ra protein thông qua hai quá trình: phiên mã (transcription) và dịch mã (translation). Trong quá trình phiên mã, vùng mã hóa của gen được sao chép thành một phiên bản RNA. Đối với tế bào nhân thực, sản phẩm RNA ban đầu được xử lý thành các phân tử RNA thông tin nhỏ hơn (mRNA), các mRNA này sau đó di chuyển đến tế bào chất. Tại đây, cỗ máy phân tử phức tạp mang tên ribosome chứa trong nó cả mRNA và protein thực hiện quá trình dịch mã. Ribosome lắp ráp và liên kết các amino acid với nhau theo một thứ tự chính xác do mRNA quy định [Lod+07].

Các amino acid được tạo thành từ các bộ ba mã hóa mà các phần tử là một trong 4 nucleotide A, C, G, U. Ta có tất cả $4^3 = 64$ bộ ba như thế, nhiều bộ ba có thể đại diện cho một amino acid. Ví dụ UUU và UUC cùng đại diện cho Phenylalanine, một α-amino acid không phân cực và trung hòa về điện. Sợi đơn DNA khuôn mẫu đánh dấu sự phiên mã tại các trình tự có độ dài từ 100 đến 1000 bp được gọi là promoter, các trình tự này đứng trước vùng mà hóa RNA và không tham gia vào quá trình phiên mã tạo thành mRNA (xem hình 1.2). Vùng bắt đầu từ promoter về phía $3'$ gọi là vùng thượng nguồn (upstream), vùng sau promoter về phía $5'$ gọi là vùng hạ nguồn (downstream). Vùng hạ nguồn chứa vùng mã hóa RNA có độ dài từ 20000 đến 30000 bp. Trên thực tế, sản phẩm phiên mã đầu tiên của gen là pre-mRNA bao gồm cả các đoạn intron (đoạn không mã hóa amino acid) và các đoạn exon (đoạn mã hóa amino acid). Tế bào sẽ loại bỏ các đoạn intron để sản xuất ra các mRNA trưởng thành trước khi tiến hành dịch mã. mRNA đã bao gồm các codon, nhưng không phải tất cả các trình tự của mRNA đều tham gia quá trình dịch mã. Vùng gần $5'$ nằm giữa nucleotide đầu tiên và codon bắt đầu AUG được gọi là vùng không dịch mã (untranslated region, viết tắt là UTR) hoặc trình tự dẫn hướng (leader sequence). Ở người, vùng $5'$ UTR có chiều dài khoảng 170 nucleotides gắn với protein của ribosome, nơi đây còn chứa các trình tự điều hòa. Bộ ba đầu tiên AUG mã hóa cho Methionine có thể là một amino acid của sản phẩm protein, cũng có thể bị loại bỏ. mRNA kết thúc quá trình dịch mã tại codon kết thúc, nó có thể là một trong những bộ ba UAA, UAG, và UGA mà không mã hóa cho một amino acid nào.

1.1.3 Nhiễm sắc thể

Phần lớn DNA nằm trong nhân của tế bào nhân thực được gấp nhiều lần thành hình dạng các nhiễm sắc thể và được tái tạo trong quá trình phân chia tế bào. Mỗi nhiễm sắc thể chứa một phân tử DNA liên kết với một số protein nhất định. Khi tế bào phân chia, một cỗ máy phân chia gồm nhiều protein (được gọi

là replisome) tách DNA trong nhiễm sắc thể thành hai sợi, mỗi sợi được dùng làm khuôn để tổng hợp nên sợi bổ sung ngược của nó. Kết quả là chúng ta có hai chuỗi DNA mới giống chuỗi ban đầu. Ngoại trừ tế bào trứng và tinh trùng, mỗi tế bào người có 23 cặp nhiễm sắc thể, một nửa được di truyền từ bố, nửa còn lại được di truyền từ mẹ, do đó có hai bản sao của một gen được di truyền từ bố và mẹ (lưỡng bội, hay diploid). Riêng đối với các nhiễm sắc thể giới tính, nữ giới sở hữu hai nhiễm sắc thể X trong khi nam giới sở hữu một nhiễm sắc thể X và một nhiễm sắc thể Y. Đối với tế bào trứng và tinh trùng, quá trình phân chia được gọi là meiosis, nó trải qua hai giai đoạn phân bào chia một tế bào ban đầu thành 4 tế bào con trong đó mỗi tế bào con chỉ chứa một bản sao của mỗi nhiễm sắc thể (đơn bội, hay haploid).



Hình 1.2: Sợi đơn DNA khuôn mẫu đánh dấu quá trình phiên mã tại vùng promoter, vùng này không tham gia quá trình phiên mã. Vùng từ promoter đến $3'$ gọi là vùng thượng nguồn (upstream), vùng ngay sau promoter đến $5'$ gọi là vùng hạ nguồn. Vùng hạ nguồn chứa vùng mã hóa RNA mà kết thúc tại đoạn terminator. Các pre-mRNA sau khi được phiên mã từ gen vẫn chứa các đoạn exon (đoạn mã hóa amino acid) và các đoạn intron (đoạn không mã hóa amino acid). Tế bào sẽ loại bỏ các đoạn intron để tạo ra các mRNA trưởng thành trước khi tiến hành dịch mã. Trong mRNA, đoạn $5'$UTR được gắn với ribosome, bao gồm các trình tự điều hòa và không tham gia quá trình dịch mã. Vùng dịch mã chứa các codon, trong đó codon bắt đầu luôn là AUG, codon kết thúc có thể là UAA, UAG, hoặc UGA không mã hóa cho một amino acid nào.

1.1.4 Đột biến

Đột biến là sự thay đổi trình tự các nucleotide xảy ra trong quá trình nhân đôi của DNA gây ra sự thay đổi về chức năng của protein. Các đột biến có thể có lợi, có hại, hoặc không ảnh hưởng đến sinh vật. Đột biến có thể di truyền nếu chúng nằm trong các tế bào có khả năng góp phần hình thành con cái, các tế bào này còn được gọi là tế bào mầm (germ-line cell). Ví dụ tế bào chứng, tinh trùng và các tế bào tiền thân của chúng là những tế bào mầm. Ngược lại, đột biến không di truyền nếu chúng nằm trong các tế bào không đóng góp vào sự hình thành con cái như các tế bào cơ thể, loại tế bào này được gọi là tế bào soma (somatic cell). Tuy nhiên, các đột biến trong tế bào soma vẫn có thể tích lũy theo thời gian gây ra bệnh cho người, chẳng hạn căn bệnh ung thư.

Một số khái niệm liên quan đến đột biến mà sẽ được nhắc đến nhiều trong các phần tiếp theo đó là allele, kiểu gen, kiểu hình và biến thể:

- Allele: là các phiên bản khác nhau của gen mã hóa cho các phiên bản khác nhau của protein.
- Kiểu gen (genotype), kiểu hình (phenotype): kiểu gen là tập hợp các allele mã hóa các đặc điểm được biểu hiện về mặt thể chất của sinh vật như: màu mắt, chiều cao..., các biểu hiện này còn được gọi là kiểu hình. Ví dụ kiểu gen biểu hiện màu mắt của một cá thể có hai allele tương ứng với hai gen cùng quy định màu mắt nằm trên một cặp nhiễm sắc thể. Allele 1 nhận được từ bố biểu hiện màu mắt xanh, allele 2 nhận được từ mẹ biểu hiện màu mắt đen. Kiểu hình tương ứng với kiểu gen này là màu mắt xanh. Có thể nói thêm rằng kiểu hình là biểu hiện của kiểu gen cộng với môi trường.
- Biến thể (variant): là vùng của gen với các khác biệt về trình tự. Bộ gen con người có 3 tỷ cặp bazơ tương ứng với 6 tỷ nucleotide, trong đó 99,9% trình tự giống nhau, chỉ có khoảng 6 triệu nucleotide khác nhau giữa các cá thể. Các biến thể phát sinh trong quá trình nhân đôi của DNA với tỷ lệ một trên một tỷ cặp bazơ. Như vậy, tại mọi thời điểm một tế bào phân chia và DNA được sao chép xuất hiện trung bình 3 nucleotide khác biệt.

Một gen đột biến có thể chứa một hoặc nhiều biến thể, để đánh giá tác động của đột biến gen ta cần xác định các loại biến thể và mức độ ảnh hưởng của chúng đến cơ thể sống.

1.1.5 Bệnh liên quan đến gen

Ở cấp độ loài, sự thay đổi trong trình tự DNA có thể được di truyền giúp loài đó thích nghi với môi trường sống, cũng có thể gây ra sự tuyệt chủng. Ở cấp độ cá thể, một số đột biến cải thiện thể chất, hầu hết các đột biến không ảnh hưởng đến các chức năng sinh học, và một số không phù hợp. Bệnh có thể

được định nghĩa là những thay đổi không thích hợp ảnh hưởng đến các cá thể trong một quần thể. Các thay đổi ấy gây ra tình trạng bất thường, suy giảm chức năng sinh lý [Pev15].

Trên phương diện tin sinh học, chúng ta nghiên cứu bệnh ở cấp độ phân tử như DNA, RNA và protein; ở cấp độ hệ thống như: bào quan, nội tạng; ở cấp độ sinh vật như: kiểu hình lâm sàng, mô hình động vật, tổ chức và cơ sở. Tại mỗi cấp độ đều có những cơ sở dữ liệu lớn phục vụ cho việc nghiên cứu. Trong đó, một số rối loạn liên quan đến gen được liệt kê như sau:

- Rối loạn đơn gen chỉ do một gen đột biến gây ra, hay còn gọi là bệnh Mendel. Ví dụ về rối loạn đơn gen như: thiếu máu hồng cầu hình liềm (Sickle cell anemia), múa giật (Huntington), ung thư vú BRCA1 và BRCA2, máu khó đông A (Hemophilia A).
- Rối loạn phức tạp xuất hiện do sự kết hợp các đột biến trong nhiều gen cộng với sự ảnh hưởng của môi trường và hành vi làm tăng nguy cơ bệnh. Ví dụ về rối loạn phức tạp như: tâm thần phân liệt (schizophrenia), hen suyễn (asthma), trầm cảm (depression), tiểu đường (diabetes), cao huyết áp, béo phì.
- Rối loạn hệ gen gây ra những bất thường trên các nhiễm sắc thể ở quy mô lớn. Một số rối loạn bộ gen liên quan đến những thay đổi quy mô lớn như thể dị bội (aneuploidies) trong đó một bản sao nhiễm sắc thể bị thừa (tam nhiễm, tiếng Anh là trisomy) hoặc bị mất (đơn bội, tiếng Anh là monosomy), hai bản sao bị thừa (tetrasomy) hoặc bị mất (nullsomy). Ví dụ về bệnh liên quan đến thể dị bội như: hội chứng Patau (trisomies 13), hội chứng Edwards (trisomies 18), hội chứng Down (trisomies 21). Ngoài ra, còn có những rối loạn hệ gen liên quan đến các nhiễm sắc thể giới tính như: hội chứng Klinefelter (47, XXY), hội chứng XYY (47, XYY), hội chứng tam bội X (47, XXX).

1.2 Các công nghệ giải trình tự DNA

1.2.1 Giải trình tự Sanger

Công nghệ giải trình tự đầu tiên được Frederick Sanger cùng các cộng sự nghiên cứu từ năm 1977 và được thương mại hóa vào năm 1986. Phương pháp giải trình tự Sanger dựa trên việc sao chép DNA trong ống nghiệm (vitro replication). Trên thực tế, bộ gen người đầu tiên đã được giải trình trong một thập kỷ nhờ công nghệ giải trình tự Sanger. Công nghệ này có thể chia ra làm 4 bước chính (xem hình 1.3) như sau:

- Bước 1: Gia nhiệt để tách sợi xoắn kép DNA thành hai sợi đơn. Điều này được thực hiện do liên kết hidro giữa hai sợi dễ bị phân tách bởi nhiệt độ.

- Bước 2: Bơm protein DNA polymerase và bổ sung các nucleotide mới. DNA polymerase giúp liên kết các nucleotide để tạo nên sợi bổ sung ngược của sợi khuôn mẫu ban đầu.

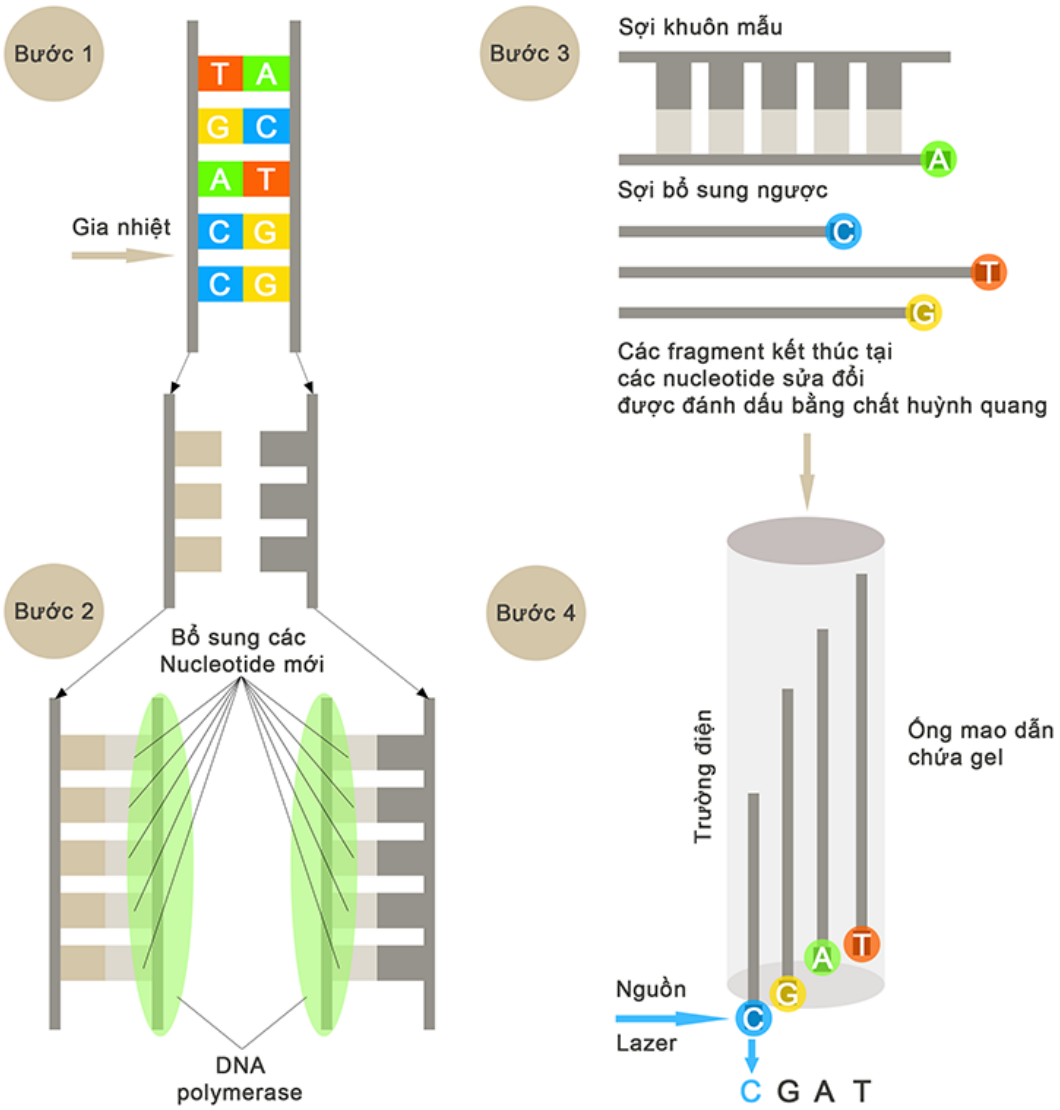
- Bước 3: Để ngăn các sợi bổ sung ngược tiếp tục kéo dài, người ta thêm vào các nucleotide chỉnh sửa được đánh dấu bởi chất huỳnh quang có các màu khác nhau. Khi mỗi chuỗi bổ sung ngược được liên kết bởi một nucleotide chỉnh sửa thì DNA polymerase sẽ không thể kiên kết thêm các nucleotide vào nó, chuỗi bổ sung ngược dừng lại tại đây. Cả sợi khuôn mẫu và sợi bổ sung ngược sau đó lại được tách ra như bước 1, mỗi sợi lại trở thành một sợi khuôn mẫu để tổng hợp một DNA mới. Sau mỗi chu trình, số lượng các DNA tăng lên gấp đôi, nhưng chiều dài các sợi bổ sung ngược ngắn hơn sợi khuôn mẫu, chúng được gọi là các đoạn (fragment). Chiều dài các đoạn không xác định do vị trí các nucleotide chỉnh sửa là ngẫu nhiên.

- Bước 4: Các đoạn DNA mang điện tích âm được cho qua một ống nghiệm chứa gel trong trường điện, các fragment có chiều dài ngắn hơn (nhỏ hơn) sẽ trôi xuống cuối ống nghiệm trước. Ở đây, một nguồn lazer được sử dụng để chiếu vào các fragment, vì các nucleotide kết thúc được đánh dấu bởi huỳnh quang nên máy có thể xác định nó là loại nucleotide nào. Quá trình này diễn ra liên tiếp cho đến khi máy giải trình tự đọc được một chuỗi trình tự.

Đối với công nghệ giải trình tự Sanger, xác xuất tổng hợp các đoạn dài là nhỏ. Chiều dài trình tự DNA lớn nhất chỉ là 1000 bp, các trình tự DNA có chiều dài lớn hơn sẽ được chia thành các phần nhỏ hơn và được giải trình tự một cách tách biệt. Kết quả được xuất ra từ máy giải trình tự Sanger là các trình tự ngắn (short read). Mặc dù giải trình tự Sanger chậm hơn giải trình tự thế hệ mới (Next genration Sequencing, viết tắt là NGS) nhưng có độ chính xác cao, do đó người ta vẫn sử dụng công nghệ này để giải trình tự nhắm mục tiêu đến gen, và để kiểm tra kết quả của các phương pháp giải trình tự mới.

1.2.2 Giải trình tự thế hệ tiếp theo (NGS)

Công nghệ giải trình tự thế hệ tiếp theo có thể giải trình tự DNA với tốc độ rất nhanh và giá thành ngày càng giảm, nhờ đó đã tạo ra những thành tựu khoa học ấn tượng và các ứng dụng sinh học mới. Cho đến năm 2008, các công cụ giải trình tự thế hệ tiếp theo có thể tạo ra một lượng dữ liệu trong 24h tương đương với vài trăm máy giải trình Sanger nhưng chỉ cần một người duy nhất vận hành [Sch08]. Dựa trên nguyên tắc giải trình tự Sanger, phương pháp giải trình tự thế hệ tiếp theo đã có những cải tiến lớn. Trong mỗi chu kỳ, nếu như công nghệ giải trình tự Sanger chỉ cho phép thao tác trên từng fragment thì công nghệ NGS có thể thao tác song song trên nhiều fragment.

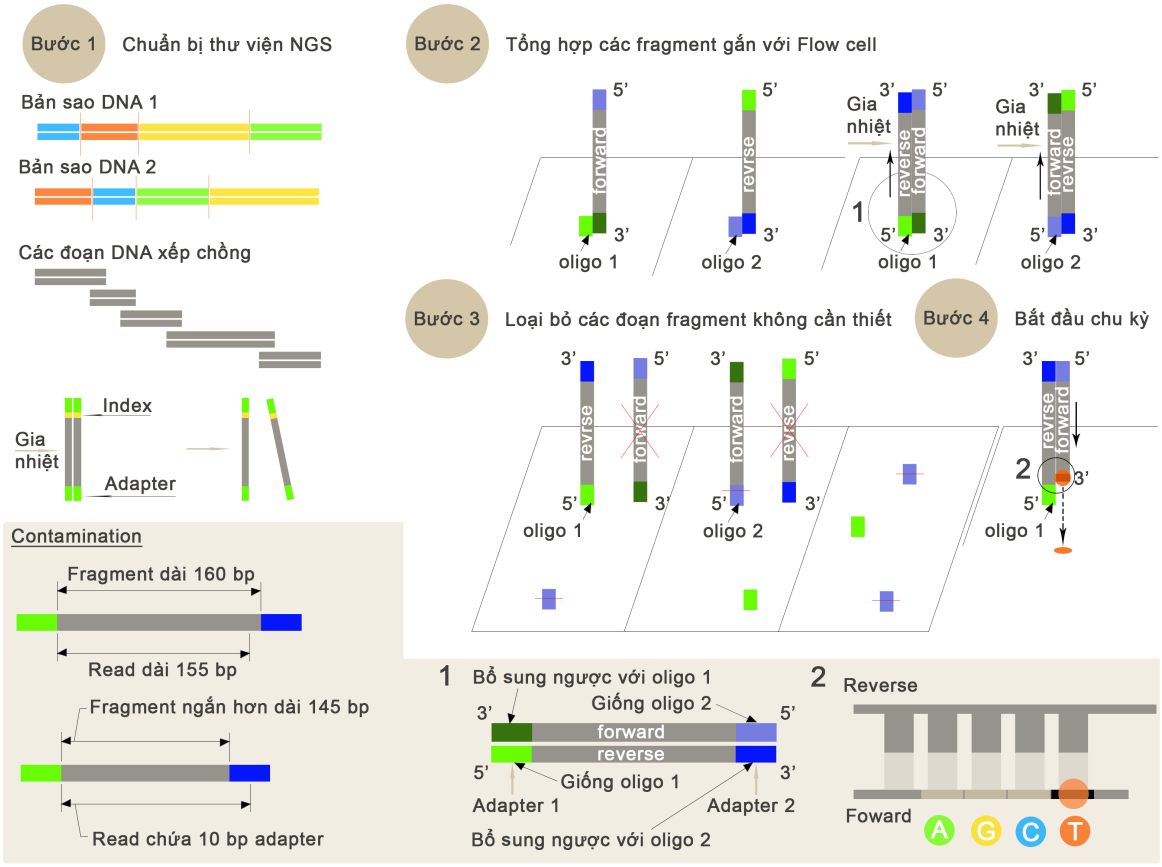


Hình 1.3: Phương pháp giải trình tự Sanger bao gồm 4 bước. Bước 1: gia nhiệt để tách DNA thành 2 sợi đơn. Bước 2: bơm protein DNA polymerase và thêm các nucleotide mới để tổng hợp các DNA mới. Bước 3: Để các sợi tổng hợp không tiếp tục kéo dài, người ta bổ sung các nucleotide chỉnh sửa, các nucleotide này sẽ liên kết vào cuối mỗi chuỗi bổ sung ngược. Các nucleotide chỉnh sửa được đánh dấu bằng chất huỳnh quang với các màu khác nhau. Bước 4: các đoạn DNA mang điện tích âm được cho qua một ống nghiệm chứa gel trong trường điện, đoạn nào ngắn hơn sẽ trôi xuống đáy ống nghiệm trước. Cuối cùng, nguồn laser được thiết kế chiếu vào phía đáy ống nghiệm để xác định loại nucleotide chỉnh sửa. Quá trình này diễn ra liên tiếp cho đến khi máy giải trình tự đọc được một chuỗi trình tự.

Hiện nay, công nghệ giải trình tự thế hệ tiếp theo đang được phát triển bởi nhiều tổ chức, do đó có một số phương pháp với những điểm khác biệt. Sau đây, chúng ta sẽ tìm hiểu các bước chính trong một phương pháp tiêu biểu của công nghệ NGS (xem hình 1.4).

Bước 1: chuẩn bị thư viện NGS. Để chuẩn bị thư viện NGS, trước hết cần chuẩn bị các thư viện DNA tương ứng với mỗi bản sao của DNA mà sau đó được cắt ra thành các đoạn DNA nhỏ hơn. Trong quá trình này, các đoạn trình tự chỉ số (index) được thêm vào đầu các đoạn DNA giúp phân loại nó thuộc thư viện nào. Các trình tự index này cũng sẽ gắn với các fragment sau khi được tách ra từ các đoạn DNA bởi quá trình gia nhiệt. Như vậy, mỗi thư viện NGS sẽ có một trình tự chỉ số khác nhau. Sự phân biệt các thư viện NGS bằng các trình tự chỉ số là cần thiết khi chúng được kết hợp với nhau trên một flow cell để giải trình tự các fragment, quá trình này được gọi là multiplexing. Các trình tự chỉ số cũng được giải trình tự, chúng được sử dụng để tách các trình tự fragment của các thư viện khác nhau thành các tệp khác nhau, quá trình này được gọi là demultiplexing. Mặt khác, các đoạn nucleotide nhân tạo có dạng bổ sung ngược với các Oligo cũng được thêm vào cuối mỗi đoạn DNA, chúng được gọi là các adapter. Trình tự nhận được từ một fragment được gọi là read, một read không bao gồm trình tự chỉ số và adapter. Nếu một read chứa một phần adapter, người ta nói read đó bị nhiễm bẩn adapter. Xét một ví dụ về sự nhiễm bẩn adapter (Xem hình 1.4, contamination). Giả sử read có chiều dài cố định là 155 bp. Nếu fragment ban đầu có chiều dài 160 bp thì read này không chứa adater ở cuối. Nếu một vài fragment có chiều dài nhỏ hơn, chẳng hạn 145 bp, tức là read thu được sẽ phải bao gồm 10 bp adapter ở cuối. Để hạn chế nhiễm bẩn adapter ta cần chọn các fragment có chiều dài tương đương, bước này còn gọi là chọn cỡ (size selection). Miền giá trị của cỡ thay đổi theo yêu cầu của từng máy giải trình tự.

Bước 2: tổng hợp các fragment gắn với flow cell (một miếng kính được chia thành nhiều làn). Trong bước này, các chu kỳ giải trình tự chưa được tiến hành mà mục đích là để tạo ra các fragment gắn với flow cell. Trước tiên, các fragment được chuẩn bị từ các thư viện NGS được đưa vào flow cell, chúng sẽ liên kết với các oligo (được gắn cố định trên flow cell) nhờ các adapter. Sau đó, protein DNA polymerase và các nucleotide mới được bơm vào, quá trình tổng hợp các đoạn bổ sung ngược được bắt đầu từ vị trí các oligo, nói cách khác các oligo chính là một phần của các đoạn bổ sung ngược. Các đoạn xoắn kép DNA sau khi tạo thành sẽ được gia nhiệt để tách thành hai đoạn xoắn đơn với các chiều thuận (forward) và nghịch (reverse). **Bước 3: loại bỏ các fragment không cần thiết.** Đây là bước xử lý trước khi bắt đầu các chu kỳ giải trình tự, bước này ảnh hưởng đến loại read được xuất ra từ máy giải trình tự. Mục đích của quá trình tổng hợp là tạo ra các fragment chỉ cùng một chiều hoặc cả hai chiều. Dù trong trường hợp nào, các fragment của cùng một chiều cũng phải được tổng hợp tách biệt. Vì vậy, ta cần loại bỏ các fragment trong thư viện NGS



Hình 1.4: **Bước 1: chuẩn bị thư viện NGS**. Các DNA được cắt ra thành những đoạn nhỏ hơn mà được thêm vào các adapter ở cuối. Chúng được gia nhiệt để tách thành các đoạn DNA chứa fragment, adapter và index. **Bước 2: tổng hợp các fragment gắn với flowcell**. Đưa các fragment vào flow cell, các fragment thuận (từ $5'$ đến $3'$) được liên kết với các oligo 1 bởi adapter bổ sung ngược với nó, tương tự với các fragment nghịch (từ $3'$ đến $5'$) được liên kết với các oligo 2. DNA polymerase và các nucleotide mới sau đó được bơm vào flow cell, các fragment nghịch sẽ được tổng hợp từ oligo 1, các fragment thuận được tổng hợp từ oligo 2. **Bước 3: loại bỏ các fragment không cần thiết**. Các đoạn DNA mới tạo thành được gia nhiệt để tách thành các đoạn sợi xoắn đơn. Các fragment không gắn các oligo thì không liên kết được với flow cell và bị loại bỏ. Các fragment thuận được loại bỏ nhờ việc cắt các oligo 2. Cuối cùng ta thu được các fragment có chiều nghịch chứa oligo gắn với flowcell để sử dụng cho quá trình tổng hợp các fragment có cùng chiều thuận. **Bước 4: thực hiện các chu kỳ giải trình tự**. Các fragment bổ sung ngược được tổng hợp theo chiều thuận với các nucleotide chỉnh sửa được bơm vào. Mỗi khi một nucleotide chỉnh sửa được liên kết vào đoạn bổ sung ngược, máy ảnh sẽ chụp lại nucleotide này, nhờ màu lưu huỳnh đánh dấu mà loại nucleotide được xác định. Sau đó, nucleotide được đảo ngược về nucleotide bình thường, một chu kỳ mới lại bắt đầu.

ban đầu và các fragment có chiều thuận hoặc nghịch gắn với oligo. Không mất tính tổng quát, giả sử ta cần tổng hợp các fragment có chiều thuận, do đó cần dữ lại các fragment có chiều nghịch. Vì sau khi gia nhiệt, các fragment của thư viện NGS ban đầu không còn liên kết với các oligo, nên chúng dễ dàng bị loại bỏ. Các fragment có chiều thuận gắn với oligo thì vẫn liên kết chặt với flow cell, để loại bỏ chúng máy giải trình tự sẽ tiến hành cắt bỏ loại oligo gắn với các fragment này. Cuối cùng, ta nhận được các fragment chỉ có một chiều nghịch được gắn chặt với flow cell.

Bước 4: thực hiện các chu kỳ giải trình tự. Như đã đề cập ở khác biệt thứ nhất giữa giải trình tự thế hệ tiếp theo với giải trình tự Sanger, các nucleotide chỉnh sửa trên đoạn tổng hợp ngược sẽ liên tiếp được đảo ngược về các nucleotide thường. Trước khi đảo ngược, một máy ảnh sẽ chụp lại các nucleotide đó, chúng được phân biệt nhờ màu huỳnh quang đánh dấu. Các chu kỳ diễn ra liên tục cho đến khi máy giải trình tự đọc được một trình tự của đoạn xoắn đơn DNA.

Như vậy, hai cải tiến chính của giải trình tự thế hệ tiếp theo so với giải trình tự Sanger là:

- Nucleotide chỉnh sửa sau khi được đọc sẽ được chuyển đổi thành nucleotide thường. Quá trình tổng hợp sợi bổ sung ngược lại tiếp tục với một nucleotide chỉnh sửa khác. Quá trình diễn ra liên tục, một chu kỳ được kết thúc bằng một lần đọc nucleotide, số lượng chu kỳ chính bằng số nucleotide được đọc (xem hình 1.4, 2). Quá trình này còn được gọi là giải trình tự bằng phương pháp tổng hợp.
- Thay vì sử dụng ống nghiệm, người ta sử dụng một miếng kính được gọi là flow cell được chia thành nhiều làn. Không có trường điện hay các DNA được tích điện âm, thay vào đó là các adapter và oligo. Với phương pháp này hàng nghìn fragment được tổng hợp đồng thời và tách biệt (xem hình 1.4, bước 3).

Từ cơ chế hoạt động của công nghệ NGS, ta rút ra một số khái niệm sau:

- Thư viện DNA (DNA library): là DNA được lấy ra từ một cá thể để chuẩn bị cho việc giải trình tự. DNA sau đó được cắt ra thành các đoạn nhỏ hơn. Mỗi bản sao của một DNA thuộc một thư viện nào đó, các bản sao sau khi được cắt ra sẽ tạo thành các đoạn xếp chồng (overlapping patterns) mà sau đó có thể dùng thuật toán để ghép nối chúng lại với nhau (xem hình 1.4, bước 1).
- Oligo: là một đoạn ngắn của sợi đơn DNA nhân tạo được gắn với flow cell. Oligo liên kết với fragment thông qua các adapter, từ đó cố định vị trí của các đoạn sợi đơn DNA trong quá trình tổng hợp (xem hình 1.4, bước 2).

- Adapter: là các đoạn nucleotide nhân tạo được thêm vào cuối mỗi đoạn DNA, các đoạn adapter có dạng bổ sung ngược với các Oligo, do đó chúng có thể liên kết với các Oligo. Sau quá trình gia nhiệt, các đoạn DNA tách ra thành các đoạn sợi đơn, mỗi đoạn sợi đơn sẽ chứa các adapter ở cuối (xem hình 1.4, bước 1).
- PCR (polymerase chain reaction): là kỹ thuật tạo nhiều bản sao DNA nhờ sử dụng enzyme DNA polymerase.
- Thư viện NGS (NGS library): bao gồm các đoạn sợi đơn DNA của một cá thể được gắn adapter (xem hình 1.4, bước 1).
- Cụm (cluster): bao gồm một fragment mà oligo là một phần trình tự của nó được gắn với flow cell và đoạn tổng hợp ngược của nó trong quá trình tiếp diễn các chu kỳ. Các fragment có chứa olio được tổng hợp từ các fragment trong thư viện NGS được liên kết với các oligo bởi các adapter.

1.2.3 Các loại trình tự nhận được từ máy giải trình tự

Như đã đề cập trong công nghệ NGS, các fragment được tổng hợp tách biệt theo cùng một chiều. Dựa vào chiều và khoảng cách các read ta có ba loại trình tự có thể được xuất ra từ máy giải trình tự:

- Single-end reads: toàn bộ các read được tổng hợp theo cùng một chiều thuận hoặc nghịch.
- Paired-end reads: hai read của cùng một đoạn DNA được tổng hợp theo cả hai chiều thuận và nghịch, chúng có thể có hoặc không có trình tự chung (overlapping sequence). Khi kết hợp hai read thuận và nghịch của cùng một đoạn DNA có thể tạo ra một read dài hơn single-end read. Hơn nữa, loại dữ liệu này tạo thuận lợi hơn cho việc lắp ráp trình tự và tìm các biến thể.
- Mate pairs: nếu hai read được tổng hợp từ cùng một đoạn DNA không có trình tự chung thì chiều dài của fragment sẽ bằng tổng chiều dài của hai read, đoạn trình tự ở giữa (inner distance), và adapter. Nếu không tính đến adapter ta có một insert size. Mate pairs là dạng đặc biệt của paired-end reads khi insert size dài cỡ 2000 bp đến 5000 bp.

1.3 Các bài toán tin sinh học

1.3.1 Một số bài toán phổ biến

Trong những năm gần đây, chúng ta đã nghe nói nhiều đến liệu pháp gen (gen therapy) hoặc chỉnh sửa gen (genome editing). Để thực hiện được điều đó, các

nhà khoa học đã không ngừng khám phá những thông điệp còn ẩn dấu trong các trình tự sinh học. Ví dụ, tần số xuất hiện của một mẫu trình tự trong một đoạn DNA, cơ chế nhân đôi DNA, và quá trình khử amin cho chúng ta manh mối về vùng bắt đầu sao chép của DNA (gọi là oriC) [Com15]. Bài toán định vị oriC không chỉ giúp ta hiểu cách tế bào tái tạo mà còn giúp con người giải quyết các vấn đề y sinh khác nhau. Một ứng dụng của nó là phương pháp trị liệu gen, người ta cấy một bộ gen nhỏ được biến đổi gen (được gọi là vector virus) vào thành tế bào. Trong nông nghiệp, vectơ virus mang gen nhân tạo đã được sử dụng để tạo ra cà chua chịu sương giá và ngô kháng thuốc trừ sâu. Năm 1990, trong bài báo "Sự bắt đầu", bác sỹ William French Anderson đã công bố liệu pháp gen lần đầu tiên được thực hiện thành công trên người khi nó cứu sống của một bé gái bốn tuổi bị rối loạn suy giảm miễn dịch kết hợp nghiêm trọng [And90]. Một ví dụ khác, quá trình phiên mã DNA thành các mRNA cho chúng ta tín hiệu về việc xác định các gen quy định chức năng của protein. Quá trình này đóng vai trò cốt yếu trong bài toán phân tích biểu hiện gen.

Kể từ khi Watson và Crick tìm ra cấu trúc xoắn kép của DNA, rất nhiều bài toán đã được đặt ra với mục đích giải mã bộ gen người và áp dụng các kiến thức thu được vào thực tiễn. Trong tài liệu "Giới thiệu về tin sinh học" [4] của Hồ Tú Bảo, một số bài toán về tin sinh học đã được phân chia như sau:

- Phân tích cấu trúc 3D:

  So sánh cấu trúc protein

  Dự đoán cấu trúc protein

  Mô hình hóa cấu trúc RNA

- Phân tích đường chuyển hóa:

  Đường trao đổi chất (metabolic pathway)

  Mạng điều hòa (reguratory networks)

- Phân tích trình tự:

  Dóng hàng trình tự (sequence alignment)

  Dự đoán chức năng và cấu trúc

  Tìm gen (gen finding)

- Phân tích biểu hiện:

  Phân tích biểu hiện gen

  Phân cụm gen

[4] http://www.jaist.ac.jp/ bao/Writings/IntroBioinformaticsV.pdf

Một số bài toán nêu trên có thể đồng thời được sử dụng trong các công nghệ sinh học hiện đại, điển hình là công nghệ chỉnh sửa gen. Công nghệ chỉnh sửa gen CRISPR-Cas9 được thiết kế để cắt những đoạn DNA tại vị trí mong muốn [Jin+12]. Phương pháp này đã được áp dụng cho việc chỉnh sửa đột biến gen gây bệnh ở phôi người [Ma+17], và có thể được sử dụng để chữa các bệnh liên quan đến gen trong tương lai.

1.3.2 Bài toán dự đoán ảnh hưởng của biến thể gen

1.3.2.1 Một số cách tiếp cận và hạn chế

Phân tích liên kết (Linkage analysis): là bản đồ ánh xạ liên kết hiển thị thông tin di truyền liên quan đến các nhóm liên kết (cặp nhiễm sắc thể) trong bộ gen [Pev15]. Đơn vị ánh xạ là centiMorgans (cM), dựa trên tần số tái tổ hợp giữa các điểm đánh dấu đa hình như SNPs hoặc vi tế bào (1 cM bằng một sự kiện tái tổ hợp trong 100 meioses). Đối với bộ gen người, tỉ lệ tái tổ hợp khoảng từ 1 đến 2 cM/Mb. Trong các nghiên cứu liên kết, vị trí của gen bệnh được xác định dựa trên mối liên kết của nó với các vị trí đánh dấu di truyền trên nhiễm sắc thể. Phương pháp này được áp dụng thành công cho các bệnh đơn gen, tiêu biểu là bệnh Huntington [Gus89]. Tuy nhiên, phân tích liên kết còn tồn tại một số hạn chế [ADL08]:

- Các gen bệnh chưa được biết trước gây khó khăn cho việc khoanh vùng vị trí của chúng trên các nhiễm sắc thể. Đặc biệt đối với các bệnh phổ biến có liên quan đến nhiều gen, phương pháp phân tích liên kết chưa cho thấy sự hiệu quả.

- Thường có nhiều allele gây bệnh trong một gen, do đó cần một phả hệ đủ lớn để nghiên cứu.

Nghiên cứu liên kết toàn hệ gen (Genome-Wide Association Studies): cung cấp một cách tiếp cận mạnh mẽ dựa vào SNP microarrays có từ vài trăm nghìn đến vài triệu SNP trên một mảng duy nhất. Trong thiết kế dựa trên gia đình, điểm đánh dấu được đo lường ở các cá nhân bị ảnh hưởng (probands) và các cá nhân không bị ảnh hưởng để xác định sự khác biệt về tần suất của các biến thể [Ott01]. Tuy nhiên, cỡ mẫu càng lớn thì sự thống kê càng chính xác. Trong các thiết kế dựa trên dân số, một số lượng lớn trường hợp bị bệnh và không bị bệnh được nghiên cứu (thường là hàng trăm hoặc hàng nghìn trong mỗi nhóm). Các nghiên cứu về sự liên kết trên toàn hệ gen liên quan đến hàng nghìn cá thể có hoặc không có một kiểu hình nhất định đã được sử dụng để xác định các allele phổ biến có ảnh hưởng nhỏ. GWAS thành công trong việc đưa ra bằng chứng mạnh mẽ về sự tương quan thường xuất hiện trong các vùng xa các gen mã hóa protein, đó có thể là các vùng điều hòa. Mặc dù mảng SNP tạo ra

một dữ liệu lớn các biến thể nhưng giải trình tự exome và hệ gen còn cho phép chúng ta khám phá nhiều biến thể hơn [Pev15].

1.3.2.2 Giải trình tự bộ gen người

Ngày nay, sự phát triển vượt bậc của các công nghệ mới đã giúp giảm rất nhiều chi phí cũng như thời gian giải trình tự. Thêm vào đó là sự cải tiến mạnh mẽ các công cụ tính toán y sinh làm việc với dữ liệu giải trình tự giúp phát hiện các đột biến gen ngày càng nhanh chóng và chính xác. Tùy vào mục đích mà ta tiến hành giải trình tự toàn hệ gen (WGS), giải trình tự exome (WES), hay giải trình tự nhắm mục tiêu (TSP). WGS có thể cho chúng ta thêm thông tin về các vùng không mã hóa của DNA (introns), các thay đổi trong vùng này có thể liên quan đến sự kiện phiên mã của DNA cũng như làm mất chức năng của protein. Trên thực tế, WES thường được sử dụng để giải trình tự các vùng mã hóa. Do vùng exome chỉ chiếm 2% toàn hệ gen, nên ngoài việc giảm chi phí giải trình tự, ta còn có thể tăng độ sâu bao phủ (DP). DP được tăng lên đồng nghĩa với tăng độ tin cậy trong phát hiện các thay đổi nucleotide với tần số thấp. Ngoài ra, ta còn có thể giải trình tự nhắm mục tiêu với chi phí thấp nhất. Ở đây, một số gen cụ thể hoặc các vùng mã hóa trong gen được lựa chọn nếu đã biết trước chúng có thể chứa các đột biến góp phần vào cơ chế sinh bệnh. Phương pháp này thường được sử dụng trong việc chuẩn đoán bệnh hoặc phát hiện các rối loạn đơn gen.

GWAS và Linkage chỉ sử dụng thông tin từng phần, trong khi giải trình tự bộ gen người phân tích toàn bộ mối quan hệ giữa biến thể gen và kiểu hình đang được áp dụng thành công cho các bệnh phức tạp [KB13]. Nếu mảng SNP trong GWAS tạo ra dữ liệu chỉ vài trăm nghìn đến vài triệu biến thể thì WGS và WES cho phép khám phá nhiều hơn các biến thể. Giống như hàng nghìn mẫu bình thường và mẫu bệnh đã được GWAS nghiên cứu trong những năm gần đây, số lượng lớn mẫu tương tự hiện đang được giải trình tự cho các rối loạn phức tạp như tâm thần phân liệt, rối loạn lưỡng cực và tự kỷ.

1.4 Dóng hàng trình tự

Bài toán mà chúng ta quan tâm là dự đoán sự ảnh hưởng của biến thể gen đến kiểu hình. Các phương pháp còn được chia ra làm ba nhóm chính dựa trên: thông tin tiến hóa hay còn gọi là bảo thủ trình tự (sequence conservation); cấu trúc 3D của protein; các thuộc tính khác nhau được tính toán trực tiếp từ trình tự amino acid [Kha+15]. Như đã đề cập đến sự hạn chế của một số cách tiếp cận và sự ưu việt của giải trình tự bộ gen, chúng ta tập trung vào nhóm phương pháp bảo thủ trình tự. Trong đó, **các thuật toán dóng hàng trình tự** là cơ sở của **nhóm phương pháp bảo thủ trình tự** sử dụng tối đa dữ liệu **giải trình tự bộ gen người**. Các trình tự bảo thủ là các trình tự giống nhau hoặc tương tự trong Protein, DNA, và RNA.

1.4.1 Khái niệm

Dóng hàng trình tự là cách sắp xếp các trình tự protein, DNA, hoặc RNA để xác định: các vùng giống nhau liên quan về chức năng, cấu trúc, hoặc tiến hóa giữa các trình tự; các biến thể của trình tự cá thể so với trình tự tham chiếu.

Các phần tử trong các trình tự được dóng hàng có thể là các nucleotide hoặc các amino acid, chúng được biểu diễn dưới dạng các hàng trong một ma trận. Ngoài ra, các khoảng trống được chèn vào giữa các phần tử với một tiêu chí tính điểm nào đó giúp tìm ra các chuỗi con chung dài nhất và phù hợp nhất về sinh học. Đối với protein, ta có thể tìm được các trình tự tương đồng của cùng một loài hoặc của các loài khác nhau. Đối với DNA và RNA, ta có thể tìm được các biến thể khi so sánh trình tự cá thể với trình tự tham chiếu.

1.4.2 Sự phát triển các thuật toán

Trước năm 2009 đã có nhiều nỗ lực để khớp các trình tự phân kỳ thấp với một bộ gen tham chiếu lớn. Các công cụ phổ biến thời đó là Bowtie, MAQ, và SOAP2. Về sau, Heng Li và Richard Durbin đã phát triển thuật toán của mình kế thừa những thuật toán trước đó để đưa ra gói phần mềm BWA được sử dụng rộng rãi đến tận bây giờ [LD09]. BWA bao gồm ba thuật toán: BWA-backtrack, BWA-SW và BWA-MEM. Thuật toán đầu tiên được thiết kế cho trình tự Illumina đọc tối đa 100bp, trong khi hai thuật toán còn lại sử dụng cho các chuỗi dài hơn dao động từ 70bp đến 1Mbp. BWA-MEM và BWA-SW chia sẻ các tính năng tương tự như hỗ trợ các trình tự dài và dóng hàng phân tách. Tuy nhiên phiên bản mới nhất BWA-MEM thường được khuyến nghị cho các truy vấn chất lượng cao vì nó nhanh hơn và chính xác hơn [Li13]. BWA-MEM cũng có hiệu suất tốt hơn BWA-backtrack với các trình tự Illumina dài 70-100bp [5].

Một hướng khác của thuật toán dóng hàng trình tự đã được phát triển từ lâu phục vụ cho việc tìm các trình tự tương đồng. Sankoff là người đầu tiên đưa ra bài toán dóng hàng trình tự và thuật toán lập trình động [San75]. Sau đó, thuật toán Smith-Waterman ra đời giúp so sánh các trình tự sinh học tương đương với việc tìm trình tự chung dài nhất giữa hai chuỗi phân tử [SW81]. Để phù hợp với đặc điểm sinh học, mô hình phạt khoảng trống Affine được sử dụng để tính điểm cho hai trình tự sau khi dóng hàng. Mô hình phạt khoảng trống Affine cùng với thuật toán lập trình động [Bel66] được áp dụng cho đồ thị Manhattan ba cấp để tìm trình tự chung dài nhất và phù hợp nhất với đặc điểm sinh học. Năm 1994, Thompson và các cộng sự đã tích hợp các kỹ thuật này vào công cụ CLUSTAL W để tìm các trình tự protein tương đồng. Cộng với việc sử dụng chiến thuật tham lam, CLUSTAL W đạt tốc độ rất nhanh vào thời bấy giờ [THG94]. Sau đó, các công cụ mạnh hơn như DIALIGN [Mor99], T-Coffee [NHH00] được phát triển.

[5] http://bio-bwa.sourceforge.net/

Đến năm 2004, thuật toán MUSCLE nhanh hơn và hiệu quả hơn hẳn CLUSTAL W ra đời [Edg04].

Mặc dù nhiều thuật toán được cải tiến nhưng chúng ta vẫn thường không thể dóng hàng các protein trên suốt chiều dài của chúng. Hơn nữa, tồn tại những protein có chức năng giống nhau nhưng không có bất cứ một trình tự chung nào với chiều dài đủ lớn. Vấn đề này có thể được giải quyết nhờ một công cụ so sánh trình tự dựa trên cơ sở dữ liệu gọi là *công cụ tìm kiếm dóng hàng địa phương cơ sở* (Basic Local Alignment Search Tool, viết tắt là BLAST) được công bố vào năm 1990 [Alt+90]. Công cụ này không chỉ giúp tìm kiếm sự giống nhau giữa các protein mà còn so sánh được một protein bị đột biến với các protein khác. BLAST dựa trên thuật toán tham lam, nó chạy rất nhanh và thường được sử dụng để gọi một protein dựa trên tất cả các protein đã biết trong cơ sở dữ liệu.

CHƯƠNG 2

PHÁT TRIỂN CÁC THUẬT TOÁN DÓNG HÀNG TRÌNH TỰ

Với sự phát triển của các công nghệ giải trình tự thế hệ tiếp theo, việc đưa ra một bộ dữ liệu trình tự gen người (sequence read archive, viết tắt là SRA) có thể được thực hiện chỉ trong vòng từ 2 đến 4 giờ. Cùng với đó là sự phát triển nhanh chóng của các thuật toán, giải trình tự chưa bao giờ trở nên nhanh chóng với chi phí thấp như hiện tại [Vas+19]. Ở đây, chúng ta sử dụng thuật toán dóng hàng dựa trên chuyển dạng **Burrows-Wheeler (BWA)** cho việc khớp các trình tự với bộ gen tham chiếu. Sau đó, thuật toán **Smith-Waterman (SWA)** thực hiện dóng hàng lại các Haplotype để kết quả gọi biến thể chính xác hơn. Quá trình so khớp kết thúc khi ta tìm được các vị trí mà nucleotide thay đổi. Trong giai đoạn đánh giá mức độ ảnh hưởng của các biến thể đến chức năng của protein, thuật toán Smith-Waterman lại được sử dụng để tìm các trình tự tương đồng dựa trên cơ sở dữ liệu lớn về protein.

2.1 Thuật toán dựa trên chuyển dạng Burrows-Wheeler

2.1.1 Một số cấu trúc dữ liệu

2.1.1.1 Mảng hậu tố (Suffix Arrays)

Mảng hậu tố được giới thiệu lần đầu tiên vào năm 1993 bởi Udi Manber và gen Myers như là một sự thay thế hiệu quả về bộ nhớ cho cây hậu tố trước đó [MM93]. Với một chuỗi T chứa các chữ cái ta cần đưa ra một mảng chứa các số nguyên không âm lưu lại vị trí các ký tự đầu tiên của các hậu tố thuộc T. Để thuận lợi cho việc dóng hàng sau này ký tự \$ được thêm vào cuối chuỗi T. Lý do ta chọn ký tự \$ là khi sắp xếp lại chuỗi T theo thứ tự bảng chữ cái thì \$ sẽ đứng ở vị trí đầu tiên. Ngoài ra, chúng ta có một số khái niệm sau:

- *Suffix String (SS)* là chuỗi con của T bắt đầu từ một vị trí nào đó và kết thúc tại \$.
- *Suffix Matrix (SM)* là một ma trận mà các hàng là các *SS* được sắp xếp theo thứ tự bảng chữ cái.
- *Suffix Arrays (SA)* là mảng số nguyên không âm chứa thứ tự của các ký tự đầu thiên thuộc *SS* trên T.

Để xây dựng *mảng hậu tố* của chuỗi T trước tiên ta cần sắp xếp tất cả các chuỗi con *SS* theo thứ tự trong bảng chữ cái. Hon et al. đã công bố một thuật toán hiệu quả cho việc sắp xếp này với thời gian $\mathcal{O}(|T|.log(|T|)$ sử dụng bộ nhớ $\mathcal{O}(|T|.log(|\Sigma|))$ bits, với $|\Sigma|$ là số ký tự duy nhất của chuỗi T không kể ký tự \$ [Hon+07]. Sau khi có được *mảng hậu tố SA* ta có thể sử dụng nó để nhanh chóng xác định vị trí của một trình tự mỗi lần nó xuất hiện trong T. Hơn nữa, các chuỗi con *SS* sau khi sắp xếp được nhóm lại ở các vị trí liên tiếp nếu chúng có chuỗi con tiền tố giống nhau. Ví dụ trong hình 2.1, các chuỗi con của chuỗi ATCATGATC\$ được nhóm lại trong ma trận *SM*. Hai chuỗi ATC\$ và ATCATGATC\$ có chuỗi con chung ATC nên chúng được nhóm lại ở các vị trí gần nhau 1 và 2 tương ứng với các giá trị 6 và 0 trong *mảng hậu tố SA*. Để giảm không gian lưu trữ, ta có thể chỉ cần lưu *mảng hậu tố một phần* (*Partial Suffix Arrays*).

BWTM	Index	SA
\$ATCATGAT**C**	0	9
ATC\$ATCAT**G**	1	6
ATCATGATC**\$**	2	0
ATGATC\$AT**C**	3	3
C\$ATCATGA**T**	4	8
CATGATC\$A**T**	5	2
GATC\$ATCA**T**	6	5
TC\$ATCATG**A**	7	7
TCATGATC\$**A**	8	1
TGATC\$ATC**A**	9	4

Hình 2.1: *Mảng hậu tố* của chuỗi ATCATGATC\$ được được tạo thành bởi sự sắp xếp lại các *chuỗi hậu tố* của nó theo thứ tự bảng chữ cái. Kết quả ta được *mảng hậu tố SA* sắp xếp lại thứ tự các ký tự đầu tiên của các chuỗi con *SS* trên chuỗi ban đầu. Mặt khác, khi thêm các tiền tố tương ứng với các chuỗi con *SS* vào ngay sau nó ta được ma trận BWT. Cột cuối cùng của *BWTM* được in đậm đại diện cho chuyển dạng Burrows-Wheeler.

2.1.1.2 Ma trận chuyển dạng Burrows-Wheeler

Năm 1994, Michael Burrows và David Wheeler đã công bố phương pháp chuyển đổi chuỗi bằng cách sắp xếp theo khối cho thuật toán nén dữ liệu không mất thông tin, chuỗi được chuyển đổi còn được gọi là *chuyển dạng Burrows-Wheeler (BWT)* [BW94]. Để nhận được BWT của một chuỗi T ban đầu ta cần tạo ra một ma trận BWT (BWTM) như sau:

- **Bước 1**: Tạo ra tập hợp các chuỗi con SS của chuỗi T, sắp xếp các chuỗi này theo thứ tự bảng chữ cái từ trên xuống dưới ta được ma trận SM.
- **Bước 2**: Xây dựng ma trận $BWTM$ bằng cách thêm vào sau các chuỗi SS các tiền tố tương ứng của chúng trên chuỗi T.

Để ý rằng, mỗi cột của ma trận $BWTM$ là một chuyển dạng của chuỗi T. Các chuyển dạng này có một số ký tự giống nhau được sắp xếp ở các vị trí liền nhau. Ví dụ ma trận BWT được sinh ra từ chuỗi ATCATGATC$ chứa cột thứ hai là ATTT$AACCG có ba ký tự T, hai ký tự A, và hai ký tự C liền nhau (xem hình 2.1). Như vậy, chuỗi chuyển dạng này có thể được nén lại bằng cách thêm chỉ số trước các kí tự giống nhau chẳng hạn: A3T$2A2CG. Nếu một chuỗi dài như bộ gen người (chỉ chứa bốn ký tự A, T, C, G) được chuyển dạng theo cách này thì sẽ có một số lượng lớn các ký tự giống nhau nằm cạnh nhau, đây là điều kiện tốt để nén dữ liệu. Tuy nhiên, không phải chuỗi chuyển dạng nào cũng có thể giải nén về chuỗi ban đầu. **Cột cuối cùng của ma trận *BWTM* được chọn để thực hiện nén dữ liệu và có thể giải nén thành chuỗi ban đầu**.

Để thực hiện quá trình giải nén, ta không chỉ dựa vào BWT mà còn cần đến cột đầu tiên của $BWTM$ (FC). Ban đầu, có thể thấy việc nén dữ liệu không hiệu quả vì ngoài BWT ta còn phải lưu FC. Tuy nhiên, vấn đề này được giải quyết trong thuật toán dóng hàng khi ta chỉ cần lưu các vị trí của các phần tử đầu tiên thuộc tập hợp {A, T, G, C} nằm trên FC (FO). Trước hết chúng ta xét hai tính chất của $BWTM$ là tính chất chu trình và tính chất đầu cuối. Tính chất chu trình Đối với cách xây dựng $BWTM$, các dòng của ma trận là các chuỗi được thu được từ T bởi cách xoay vòng các chuỗi hậu tố và tiền tố theo một chu trình. Quay lại ví dụ với chuỗi ATCATGATC$, ký tự $ nằm ở vị trí đầu tiên tại hàng đầu tiên của $BWTM$ do ta đã sắp xếp lại các SS tại bước 1, phần 2.1.1.2. Giả sử ban đầu chỉ có hai cột FC và BWT, ta cần tìm ký tự thứ hai của dòng 1. Rõ ràng, nếu chọn $ là hậu tố thì nó sẽ đứng trước một chuỗi tiền tố thuộc T, ký tự đầu tiên của chuỗi tiền tố này chính là ký tự thứ hai của dòng 1. Khi chưa quay vòng, $ đứng ở vị trí cuối cùng của chuỗi T, do đó ký tự cần tìm là A ở đầu dòng 3. Dễ thấy dòng 3 cũng chính là chuỗi T ban đầu (Xem hình 2.1). Giả sử ta cần tìm ký tự thứ hai của dòng 3. Với một ký tự A thuộc FC ta có thể tìm thấy 3 ký tự A thuộc BWT. Để lựa chọn ký tự tiếp theo là A ở dòng nào ta phải xét đến tính chất thứ hai. Tính chất đầu cuối Sự xuất hiện lần thứ k của một kí tự trong cột đầu và lần xuất hiện thứ k của ký tự này trong cột cuối tương

ứng với cùng vị trí của ký tự này trong chuỗi T. Nói cách khác, các ký tự giống nhau trên FC có thứ tự trước sau giống với thứ tự trước sau của chúng trên BWT. Đặt số thứ tự của các ký tự giống nhau bằng các chỉ số nguyên dương. Đối với chuỗi ATCATGATC$ ta có FC là $\$_1 A_1 A_2 A_3 C_1 C_2 G_1 T_1 T_2 T_3$, BWT là $C_1 G_1 \$_1 C_2 T_1 T_2 T_3 A_1 A_2 A_3$ (Xem hình 2.2). Như vậy, từ A_2 ở dòng 3 có thể suy ra vị trí tiếp theo của nó là T_2 ở đầu dòng 9. Quá trình này được lặp lại cho đến khi tìm ra chuỗi ban đầu.

Từ tính chất chu trình và *mảng hậu tố* SA, ta có thể xây dựng chuyển dạng Burrows-Wheeler của chuỗi T với thời gian tuyến tính [OS09]. Ký hiệu BWT_i, T_i tương ứng là ký tự thứ i của BWT và T, SA_i là giá trị tại vị trí thứ i của mạng hậu tố, ta có:

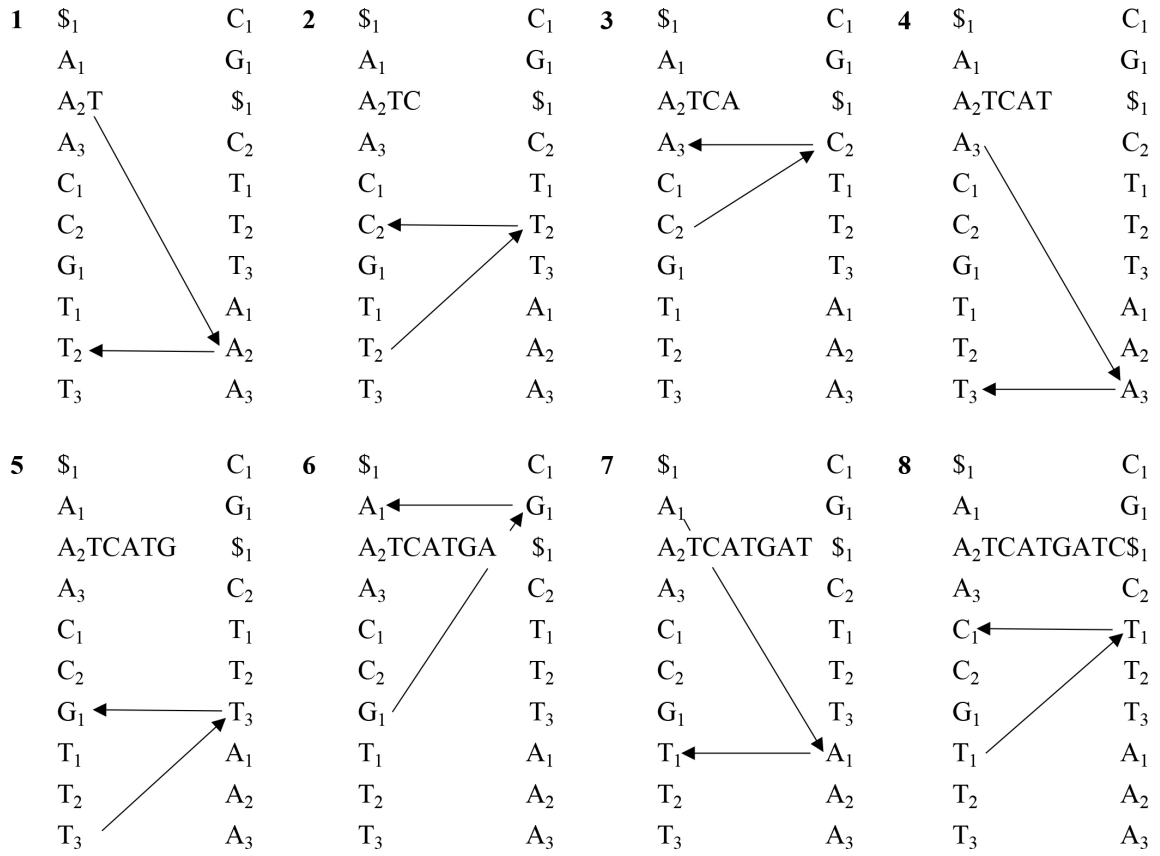
$$BWT_i = \begin{cases} T_{SA_i - 1} & \text{nếu } SA_i > 0 \\ T_{|T|-1} = \$ & \text{nếu } SA_i = 0 \end{cases}$$

với $i = 0, 1, \ldots, |T| - 1$.

Ngoài ra, khi sắp xếp lại BWT theo thứ tự bảng chữ cái ta nhận được FC. Các ký tự giống nhau trên FC được nhóm lại thành các cụm. Nhờ tính chất đầu cuối, ta chỉ cần lưu vị trí đầu tiên của các ký tự này (FO) cho các thuật toán về sau.

2.1.1.3 Ma trận điểm kiểm tra (Checkpoint Arrays)

Sau khi xây dựng được chuỗi BWT của trình tự T, vị trí của một ký tự thuộc chuỗi BWT có thể được tìm thấy trên chuỗi T. Để thực hiện điều này, ta cần đếm số lần xuất hiện của một ký tự trên BWT mà thứ tự của nó thuộc đoạn [0, n], với n = 0, ..., $|T| - 1$. Tuy nhiên thao tác này tiêu tốn khá nhiều thời gian. Do đó, một *ma trận điểm kiểm tra* (C) được sử dụng để lưu các giá trị đếm này. Giống như *mảng hậu tố*, ta cũng có thể chỉ cần lưu một phần của ma trận điểm kiểm tra để tiết kiệm không gian lưu trữ. Ví dụ trong hình 2.3 là *ma trận điểm kiểm tra* của chuỗi BWT CG$CTTTAAA. Đối với bộ gen người, nếu cả bốn cột A, T, C, G được lưu đầy đủ thì dung lượng cần thiết để lưu trữ ma trận này (15GB) gấp đến 5 lần dung lượng lưu trữ một bộ gen người (3GB). Do đó, ta có thể lưu một ma trận con của C mà chỉ chứa các dòng cách nhau một khoảng nào đó, chẳng hạn 100 đơn vị. Khi ấy dung lượng cần thiết dùng cho việc lưu trữ một phần *ma trận điểm kiểm tra* chỉ xấp xỉ 150MB mà tốc độ tính toán vẫn được nâng cao.



Hình 2.2: **Tính chất chu trình:** ở bước 1, hậu tố bắt đầu từ T ở dòng 3 được đặt lên trước tiền tố bắt đầu bằng A ở dòng 9. Tương tự, ở bước 2, hậu tố bắt đầu bằng C ở dòng 3 được đặt lên trước tiền tố AT ở dòng 6. **Tính chất đầu cuối:** ở bước 3, C_2 của FC tương ứng với C_2 của BWT, từ đó ta tìm ra ký tự tiếp theo ở dòng 3 là A.

Index	BWT	A	T	C	G
0	C	0	0	1	0
1	G	0	0	1	1
2	$	0	0	1	1
3	C	0	0	2	1
4	T	0	1	2	1
5	T	0	2	2	1
6	T	0	3	2	1
7	A	1	3	2	1
8	A	2	3	2	1
9	A	3	3	2	1

Hình 2.3: *Ma trận điểm kiểm tra* của chuỗi ATCATGATC$ lưu số lần xuất hiện của các ký tự trong tập hợp {A, T, G, C} mà thứ tự của nó trên BWT thuộc đoạn [0, n], với n = -1, 0, ..., 9.

2.1.2 Thuật toán

2.1.2.1 Thuật toán khớp chính xác

P. Ferragina và G. Manzini đã đưa ra thuật toán tìm kiếm lùi (backward search) đếm số lần xuất hiện của một mẫu P trên chuỗi T với thời gian $\mathcal{O}(|P| + occ)$ [FM05]. Trong đó, *occ* là số lần xuất hiện của mẫu P trong chuỗi T. Thuật toán này còn được gọi là FM-index, các ký tự được tìm kiếm ngược từ cuối lên đầu. Vị trí của mỗi kí tự này được xác định trên một đoạn thuộc FC. Gọi $FO(symbol)$ là thứ tự của vị trí đầu tiên mà kí tự *symbol* xuất hiện trong FC, $CO(symbol, i)$ là số lần kí tự *symbol* xuất hiện trong BWT từ vị trí 0 đến vị trí thứ i, với i = -1, 0, ..., |T| - 1. Mỗi lần xét một kí tự trong xâu mẫu P, thứ tự tương ứng với các vị trí trên và dưới của kí tự *symbol* đang tìm kiếm trên FC được cập nhật như sau:

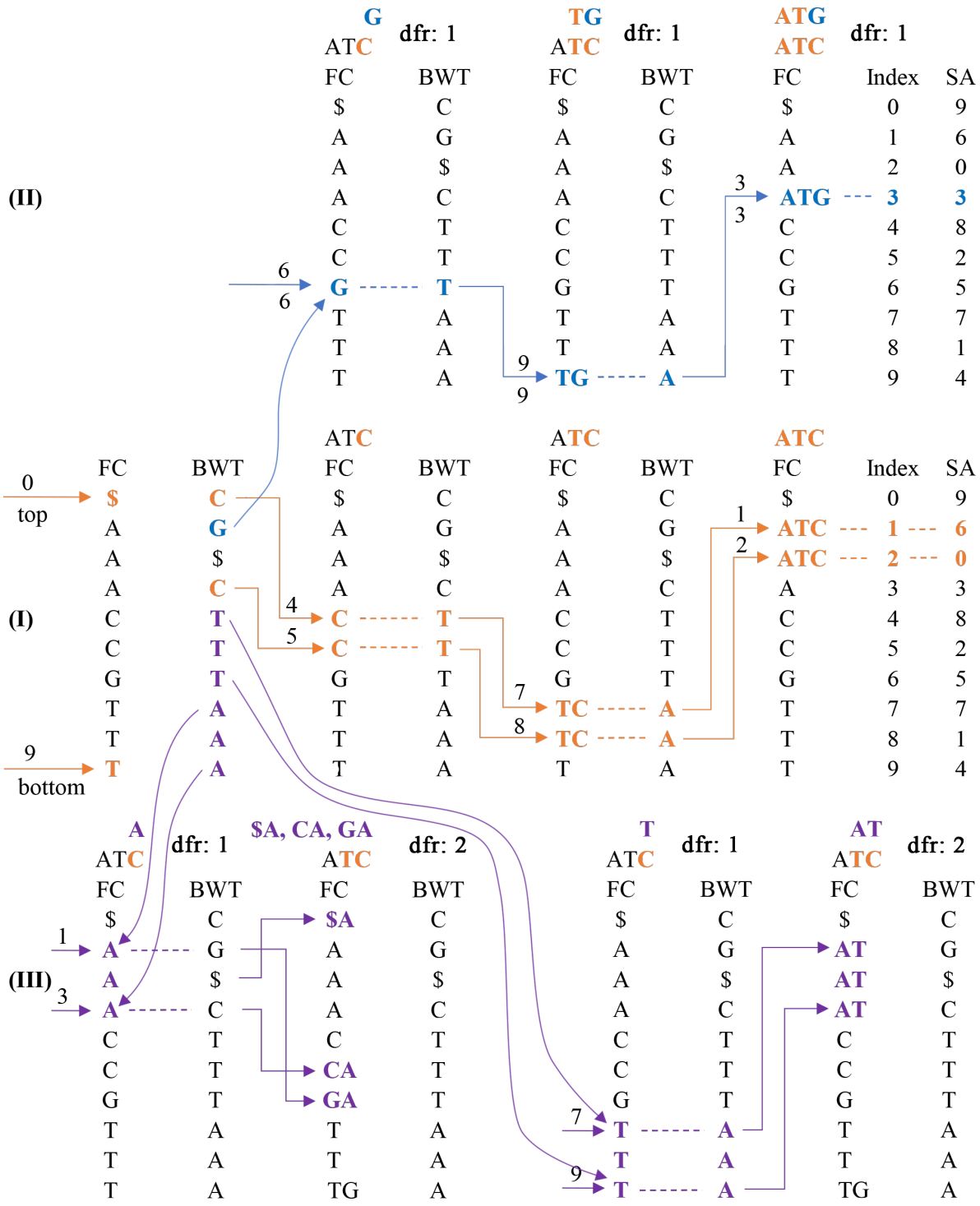
$$top \leftarrow FO(symbol) + CO(symbol, top - 1) \tag{2.1}$$

$$bottom \leftarrow FO(symbol) + CO(symbol, bottom) - 1 \tag{2.2}$$

Các vị trí khớp của mẫu P trên chuỗi T chính bằng giá trị của mảng hậu tố SA theo thứ tự tương ứng với các vị trí trên FC ở bước cuối cùng. Ở đây, hàm $CO(symbol, i)$ có thể được thay thế bằng cách gọi giá trị đếm kí tự *symbol* được lưu sẵn trong *ma trận điểm kiểm tra* để tăng tốc cho thuật toán. Trường hợp sử dụng *mảng hậu tố một phần*, ta tiếp tục quay lui các vị trí trong đoạn [*top*, *bottom*] cho đến khi các vị trí này tồn tại trong mảng hậu tố đó. Khi ấy, giá trị của vị trí khớp bằng giá trị tồn tại trong *mảng hậu tố* cộng thêm số bước quay lui. Ví dụ ta cần khớp chính xác mẫu ATC với chuỗi $ATCATGATC$ theo thứ tự từ cuối lên đầu C, T, A (Xem hình 2.4, (I)). Đoạn bắt đầu được chọn là [0, 9] gồm toàn bộ chiều dài của FC. Các ký tự C thuộc FC xuất hiện trong đoạn [0, 3] trên BWT. Để cập nhật đoạn [*top*, *bottom*] mới trên FC của bước sau theo công thức 2.1 và 2.2, ta cần tính $FO(C)$, $CO(C, -1)$ và $CO(C, 3)$. Các giá trị này lần lượt là 4, 0, và 2. Từ đó, đoạn mới của ký tự C được cập nhật trên FC là [4, 5]. Tương tự, các đoạn [*top*, *bottom*] được cập nhật tương ứng với các ký tự T và A là [7, 8] và [1, 2]. Đoạn cuối cùng [1, 2] nằm trên BWT cũng là các vị trí khớp chính xác của mẫu ATC với chuỗi đã cho. Sau khi đối chiếu với các giá trị trong *mảng hậu tố* ta được các vị trí khớp chính xác của hai chuỗi là 0 và 6.

2.1.2.2 Thuật toán khớp xấp xỉ

Đối với thuật toán tìm mẫu khớp chính xác, các kí tự thuộc mẫu P đều phải giống các kí tự trên chuỗi trình tự T theo thứ tự. Ngược lại, các kí tự thuộc mẫu khớp xấp xỉ có thể khác các kí tự trên chuỗi đã cho theo trình tự, miễn là số lượng ký tự không giống này nhỏ hơn một ngưỡng khác biệt cho phép (Khác biệt ở đây có thể là một không khớp (mismatch) hoặc một khoảng trống (gap). Khoảng trống với ý nghĩa là các Insert, hoặc các Delete liên tiếp, hay còn gọi



Hình 2.4: Quá trình tìm kiếm lùi mẫu *ATC* trong chuỗi *ATCATGATC*. (I) Mẫu khớp chính xác với chuỗi đã cho tại vị trí 0 và 6. (II) Mẫu khớp xấp xỉ tại vị trí 3 với ngưỡng khác biệt là 1. (III) Mẫu không khớp xấp xỉ vì vượt quá ngưỡng khác biệt cho phép là 1.

là Indels). Mặc dù vậy, thuật toán khớp chính xác vẫn được áp dụng cho việc tìm các mẫu khớp xấp xỉ. Vẫn với ví dụ hai chuỗi ATC và $ATCATGATC$, ta tiến hành khớp xấp xỉ với ngưỡng khác biệt là 1. Nếu bắt đầu bằng ký tự C trên BWT, ta có hai chuỗi khớp chính xác như trên, trường hợp này hiển nhiên thỏa mãn (Xem hình 2.4, (I)). Bây giờ ta xét ba ký tự còn lại là G, A, và T. Đối với ký tự G, vì G không giống C nên khác biệt được tính là 1. Tiếp tục cập nhật vị trí của G trên FC (Xem hình 2.4, (II)) ta tìm được đoạn mới [6, 6]. Các ký tự về sau tương ứng với đoạn mới này là T và A giống với các ký tự thứ hai và thứ nhất của mẫu ATC, do đó số khác biệt vẫn là 1. Cuối cùng, ta tìm được vị trí khớp xấp xỉ là 3, trong đó mẫu ATC có ký tự C khác với ký tự G của chuỗi $ATCATGATC$. Đối với các ký tự T và A, đến bước thứ 3 số khác biệt đã là 2, do đó ta không tìm được các vị trí khớp xấp xỉ với ngưỡng khác biệt là 1 (Xem hình 2.4, (III)).

Để không tốn thời gian tìm kiếm những trường hợp không thỏa mãn, ta có thể đặt một giới hạn dưới của các khác biệt. Các giới hạn khác biệt dưới của mẫu P và trình tự T có xu hướng giảm từ vị trí cuối của mẫu đến vị trí đầu của mẫu. Trong quá trình tìm kiếm lùi, nếu số các khác biệt nhỏ hơn giới hạn khác biệt dưới tại vị trí nào đó thì thuật toán không tiếp tục tìm kiếm với trường hợp này. Các giới hạn này được lưu trong một *mảng giới hạn khác biệt dưới* (DA). Nhận thấy, khả năng để một mẫu P khớp chính xác với chuỗi T nhỏ hơn khả năng một chuỗi con của P khớp chính xác với chuỗi T. Do đó, nếu ta tiến hành khớp chính xác các chuỗi con của P thì sẽ xác định được các giới hạn khác biệt dưới. Cụ thể, nếu bắt đầu thuật toán khớp chính xác từ một ký tự thuộc P giống với ký tự tương ứng của nó trên T thì bất cứ khi nào không kéo dài được chuỗi con khớp chính xác ta nhận được một chuỗi con của P khớp chính xác với T. Vị trí không kéo dài được, còn được gọi là vị trí khác biệt, có thể là một không khớp hoặc một khoảng trống. Quá trình xác định DA được bắt đầu từ ký tự đầu tiên của P, nếu vị trí tiếp theo không có khác biệt thì số khác biệt tại vị trí này bằng số khác biệt tại vị trí liền trước nó. Ngược lại, tại mỗi vị trí xuất hiện khác biệt thì số khác biệt tại đó bằng số khác biệt tại vị trí trước nó cộng thêm một đơn vị.

Thuật toán tìm DA được thiết kế gần giống thuật toán khớp chính xác, điểm khác nhau ở đây là ta tìm kiếm thuận trên mẫu P, và khớp chính xác với chuỗi đảo ngược của T (ký hiệu T') (Algorithm 1). Lý do chọn chuỗi T' là vì DA được thiết kế cho thuật toán tìm kiếm lùi. Xét ví dụ chuỗi T và P lần lượt là $TGCGTAGTA$ và $GCAGT$, các chuỗi đảo ngược của chúng T' và P' lần lượt là $ATGATGCGT$ và $TGACG$(Xem hình 2.5). Chú ý rằng mẫu P khi đem khớp với chuỗi T' theo thuật toán tìm kiếm lùi với chiều thuận của P tương đương với việc khớp mẫu đảo ngược P' với chuỗi T' theo chiều nghịch của P'. Mảng DA được tính toán nguyên tắc: bất cứ khi nào *top* lớn hơn *bottom* thì xuất hiện một khác biệt, khác biệt này có thể là một không khớp hoặc một khoảng trống (Algorithm 1). Như vậy, chuỗi con dài nhất của P' tính từ ký tự cuối cùng khớp chính xác với T' là CG, ta có $DA[0] = DA[1] = 0$. Chuỗi con này không thể kéo

dài đồng nghĩa với việc xuất hiện một không khớp hoặc một khoảng trống, do đó giá trị tiếp theo trong mảng DA được cộng thêm một đơn vị. Hơn nữa, chuỗi con tiếp theo là TGA khớp chính xác với T' nên ta có $DA[2] = DA[3] = DA[4] = 1$. Việc sử dụng mảng DA giúp tăng tốc độ tính toán cũng như giảm yêu cầu về bộ nhớ. Nhìn sang cột cuối cùng, rõ ràng với $D[4] = 1$ ta không thể khớp chính xác mẫu $GCAGT$ với chuỗi $TGCGTAGTA$, nói cách khác thuật toán không tính toán các trường hợp với khác biệt bằng 0 ngay tại ký tự cuối cùng T. Tương tự, vì $D[2] = 1$ nên có thể loại bỏ các trường hợp của A để khớp chính xác GCA.

T'/T	ATGATGCGT		TGCGTAGTA	
P'/P	TGA	CG	GC	AGT
DA	111	00	00	111

Hình 2.5: *Mảng giới hạn khác biệt dưới* sử dụng để khớp mẫu $GCAGT$ với chuỗi $TGCGTAGTA$. Để xác định mảng này ta áp dụng thuật toán tìm kiếm lùi cho mẫu đảo ngược $TGACG$ với chuỗi đảo ngược $ATGATGCGT$. *Mảng giới hạn khác biệt dưới* tìm được là $\{0, 0, 1, 1, 1\}$.

Algorithm 1 Difference Array Calculating

Input: BWT': Burrows - Wheeler Transform of Reverse String,
PAT: Pattern, FO: First Occurrence,
CO': Checkpoint Arrays of BWT'.
Output: DA: Difference Arrays.
Initialisation: $N, t, bt \leftarrow |PA|, 1, |BWT'|$; $z \leftarrow 0$

DAC (BWT', PAT, FO, CO')
1: **for** $i \leftarrow 1$ to N **do**
2: $symbol \leftarrow PAT[i]$
3: $t \leftarrow FO(symbol) + CO'(symbol, t-1)$
4: $bt \leftarrow FO(symbol) + CO'(symbol, bt) - 1$
5: **if** $t > bt$ **then**
6: $t, bt \leftarrow 1, |BWT'|$
7: $z \leftarrow z + 1$
8: **end if**
9: $DA(i) \leftarrow z$
10: **end for**
11: **return** DA

Giải thuật cho bài toán dóng hàng trình tự dựa trên chuyển dạng Burrows-Wheeler đã được trình bày dưới dạng đệ quy ([LD09]). Trong luận văn này, thuật

toán được khử đệ quy và trình bày lại dưới dạng tuần tự để tạo thuận lợi nếu ta cần lập trình song song hoặc đồng thời (Algorithm 2).

- Dữ liệu đầu vào bao gồm: BWT, *mảng giới hạn khác biệt dưới DA*, *mảng hậu tố một phần PSA*, ngưỡng khác biệt W, mẫu PAT, vị trí xuất hiện đầu tiên của các ký tự FO, và *ma trận điểm kiểm tra CO*.
- Khởi tạo mảng cập nhật thông tin với đoạn đầu tiên [1, $|BWT|$] chứa toàn bộ chiều dài T.
- Dòng 1: Khi mảng cập nhật thông tin khác rỗng thuật toán được chạy lặp lại.
- Dòng 2, 3: Lấy thông tin của đoạn [*top*, *bottom*] đầu tiên từ mảng cập nhật thông tin, đồng thời xóa đoạn này khỏi mảng cập nhật thông tin.
- Dòng 4: Tập hợp được đem khớp gồm bốn ký tự {A, T, G, C}.
- Dòng 5, 6: Nếu chiều dài chuỗi con của mẫu khác 0 ta tiến hành khớp chuỗi con này với chuỗi trình tự từ ký tự cuối cùng của chuỗi con.
- Dòng 7, 8: Duyệt qua các ký tự trong tập hợp được đem khớp, thuật toán sẽ tìm các đoạn [*top*, *bottom*] mới tương ứng với mỗi ký tự này.
- Dòng 9 đến 14: Nếu số khác biệt không nhỏ hơn giới hạn khác biệt dưới ta tiến hành cập nhật các đoạn [*top*, *bottom*] mới theo công thức 2.1, 2.2. Trường hợp không khớp, ngưỡng khác biệt được giảm đi 1 đơn vị.
- Dòng 15, 16: Nếu ngưỡng khác biệt lớn hơn 0 và $top \leq bottom$ (tức là các ký tự trong tập hợp ký tự đem khớp tồn tại trong đoạn [*top*, *bottom*] thuộc BWT) thì ta cập nhật thông tin của đoạn [*top*, *bottom*] mới.
- Dòng 20 đến 23: Nếu chiều dài chuỗi con của mẫu bằng 0 ta nhận được đoạn [*top*, *bottom*] cuối cùng. Từ đây, các vị trí khớp của mẫu với chuỗi trình tự được xác định thông qua *mảng hậu tố một phần*. Giữa hai chuỗi chỉ có một vị trí khớp nếu $top = bottom$, có nhiều vị trí khớp nếu $top < bottom$.
- Dòng 25, 26: Nếu không cập nhật được thông tin của các đoạn [*top*, *bottom*] mới mà chiều dài của chuỗi con thuộc mẫu vẫn khác không thì mẫu không khớp với chuỗi trình tự.

2.1.2.3 Cho điểm dóng hàng

Sau khi dóng hàng ta có thể nhận được nhiều kết quả giống nhau của cùng một mẫu. Vì vậy, ta cần xác định vị trí khớp nào của hai chuỗi liên quan nhiều nhất đến sinh học. Để ý rằng, sau khi dóng hàng, trên hai chuỗi sẽ xuất hiện các

khớp, các không khớp, hoặc các khoảng trống. Mẫu và chuỗi trình tự có nhiều ký tự khớp nhất không đảm bảo cho sự phù hợp nhất về mặt sinh học. Trên thực tế, ta có thể xây dựng một dóng hàng có nhiều ký tự khớp với nhiều Indels.

Algorithm 2 Sequential Approximate Pattern Matching

Input: BWT: Burrows-Wheeler Transform,
DA: Difference Arrays, PSA: Partial Suffix Arrays,
W: Difference Threshold; PAT, FO, CO.
Output: ML: Match Location, NM: Number of Mismatch.
Initialisation : $tbtUpdate \leftarrow (1, |BWT|, W, |PAT|)$.

while $|tbtUpdate| \neq 0$ **do**
  $tbt, t, bt, d, id \leftarrow tbtUpdate[1], tbt[1], tbt[2], tbt[3], tbt[4]$
  Remove tbt from $tbtUpdate$
  $apr \leftarrow (A, C, G, T)$
  if $id \neq 1$ **then**
    $PAT, symbol \leftarrow PAT[1 : id], PAT[id]$
    for $i \leftarrow 1$ to $|apr|$ **do**
      $aprS, nD, nT, nB \leftarrow apr[i], d, t, bt$
      if $nD \geq DA[id]$ **then**
        if $aprS \neq symbol$ **then**
          $nD \leftarrow nD - 1$
        end if
        $nT \leftarrow FO(aprS) + CO(aprS, t - 1)$
        $nB \leftarrow FO(aprS) + CO(aprS, bt) - 1$
        if $nT \leq nB \cap nD \geq 0$ **then**
          Append $(nT, nB, nD, id - 1)$ to $tbtUpdate$
        end if
      end if
    end for
  else
    Get all nT, nB, and nD from $tbtUpdate$
    $ML, NM \leftarrow$ values of $PSA[nT : nB]$, $W - nD$
    return ML, NM
  end if
end while
return Pattern doesn't match to sequence.

Tuy nhiên, càng nhiều Indels xuất hiện thì dóng hàng càng ít liên quan đến sinh học. Như vậy, cách chọn dóng hàng tốt nhất về sinh học là hạn chế tối đa các Indels. Để làm được điều này ta cần thưởng cho các khớp, phạt các không khớp và các Indels.

Nhận thấy, đột biến thường gây ra bởi lỗi sao chép DNA chèn (Insert) hoặc

xóa (Delete) toàn bộ một khoảng k nucleotides liên tiếp thay vì một nucleotide độc lập. Do đó nếu ta phạt như nhau các vị trí chèn hoặc xóa nucleotides thì hình phạt đó là quá mức. Để giải quyết vấn đề này, mô hình phạt khoảng trống Affine được sử dụng với các mức phạt khác nhau:

- Gap opening penalty (σ): phạt khoảng mở, vị trí đầu tiên trong gap.
- Gap extension penalty (ϵ): phạt khoảng kéo dài, các vị trí tiếp theo trong gap sau vị trí đầu tiên.

Với σ và ϵ là các số nguyên dương, giá trị của σ được chọn lớn hơn ϵ. Một gap có độ dài k > 1 bị phạt: $-\sigma - (k-1).\epsilon$. Cùng với việc cho điểm các khớp và phạt các không khớp ta tính được điểm cho dóng hàng. Vị trí khớp của mẫu được xác định dựa trên dóng hàng có điểm cao nhất. Ví dụ khớp mẫu $GCAGT$ với chuỗi $TGCGATAGTA$ (Xem hình 2.6), dóng hàng bên trái có hai khoảng mở, trong khi dóng hàng bên phải chỉ có một khoảng mở. Nếu các vị trí trong khoảng trống bị phạt cùng một giá trị thì hai dóng hàng có điểm giống nhau. Ngược lại, nếu khoảng mở bị phạt nặng hơn khoảng kéo dài thì dóng hàng bên phải có điểm cao hơn dóng hàng bên trái. Do đó, ký tự A ở bên trái được đẩy về cạnh GT ở bên phải, dóng hàng bên phải được lựa chọn có số Indels ít hơn số Indels của dóng hàng bên trái.

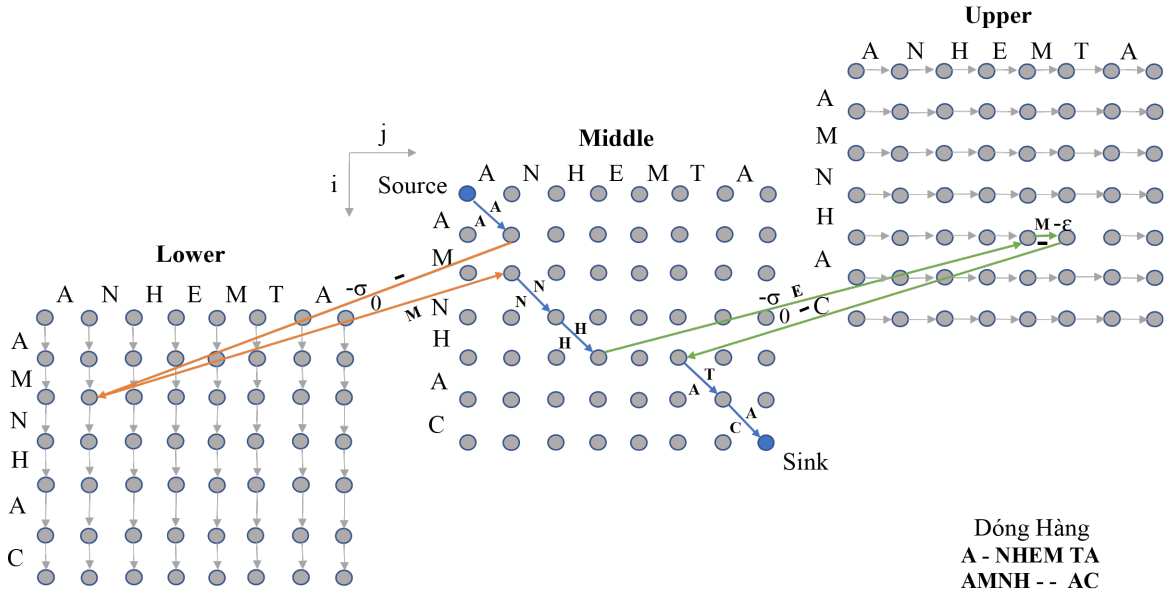
TGCGATAGTA TGCGATAGTA
GC - A - - GT GC - - - AGT

Hình 2.6: Cho điểm dóng hàng theo mô hình phạt khoảng trống Affine khi khớp mẫu $GCAGT$ với chuỗi $TGCGATAGTA$. Dóng hàng bên trái có 2 khoảng mở, trong khi dóng hàng bên phải chỉ có một khoảng mở. Do đó, dóng hàng bên phải có điểm cao hơn vì ta phạt khoảng mở nặng hơn phạt khoảng kéo dài. Dóng hàng bên phải được lựa chọn với chỉ một Indel là ba lần xóa nucleotide liên tiếp.

2.2 Thuật toán Smith-Waterman

Thuật toán dóng hàng đa trình tự dựa trên thuật toán Smith-Waterman có thể sử dụng để dóng hàng toàn bộ hoặc dóng hàng địa phương các trình tự. Tuy nhiên, trong trường hợp hai trình tự có độ dài chênh lệch lớn, dóng hàng địa phương thể hiện sự liên quan đến sinh học tốt hơn. Ta vẫn áp dụng cách cho điểm Indels theo mô hình phạt khoảng trống Affine như phần 2.1.2.3. Ngoài ra, các ma trận điểm còn được sử dụng để cho điểm các khớp và không khớp ví dụ như PAM250, BLOSUM62. Trước tiên, ta xây dựng thuật toán đệ quy dóng hàng hai trình tự dựa trên đồ thị Manhattan ba cấp, sau đó là các phương pháp cải tiến thuật toán. Cuối cùng, thuật toán tham lam được thêm vào để giải quyết bài toán dóng hàng đa trình tự.

2.2.1 Đồ thị Manhattan ba cấp



Hình 2.7: Sử dụng đồ thị Manhattan ba cấp để dóng hàng toàn bộ hai chuỗi v và w tương ứng là $AMNHAC$ và $ANHEMTA$. Bắt đầu từ soure ta có hai ký tự A giống nhau do đó đường đi đầu tiên là một đường chéo của đồ thị middle. Tiếp đó là một khoảng chèn M của v và khoảng xóa của w, đường đi chuyển từ đồ thị middle xuống đồ thị lower được đánh trọng số $-\sigma$. Đường đi từ đây quay trở lại đồ thị middle tương ứng với khoảng đóng được đánh trọng số bằng 0. Tương tự, ta có hai ký tự liên tiếp giống nhau là NE với hai đoạn đường chéo của đồ thị middle. Tiếp đến là hai ký tự xóa của v, trong đó đoạn đại diện cho khoảng mở từ đồ thị middle lên đồ thị upper được đánh trọng số $-\sigma$, đoạn đại diện cho khoảng kéo dài trên đồ thị upper được đánh trọng số $-\epsilon$. Quá trình diễn ra tương tự cho đến sink của đồ thị middle ta được hai chuỗi sau dóng hàng là $A--NHEMTA$ và $AMNH--AC$.

Đồ thị Manhattan là đồ thị dạng lưới chữ nhật hoặc vuông, trong đó các cạnh đại diện cho các ký tự được đánh trọng số, các nút được tính điểm cộng dồn từ các cạnh trước đó. Cạnh có hướng sang phải hoặc xuống dưới đại diện cho các Indels, cạnh có hướng chéo đại diện cho một khớp hoặc không khớp (Xem hình 2.8). Trình tự chung dài nhất (LCS) giữa hai chuỗi v và w được xác định bằng đường đi từ điểm đầu (source) tới điểm cuối (sink) của đồ thị. Để tìm chuỗi con chung dài nhất với mô hình phạt khoảng trống Affine ta cần xây dựng ba đồ thị Manhattan tương ứng với ba cấp:

- Đồ thị lower: chỉ gồm các cạnh đại diện cho các khoảng xóa kéo dài (Deletion extension), chúng có hướng từ trên xuống dưới. Các cạnh này được đánh trọng số $-\epsilon$.

- Đồ thị middle: chỉ gồm các cạnh đại diện cho các khớp hoặc không khớp, chúng có hướng theo đường chéo. Các cạnh này được đánh trọng số theo ma trận điểm.
- Đồ thị upper: chỉ gồm các cạnh đại diện cho các khoảng chèn kéo dài (Insertion extension), chúng có hướng từ trái sang phải. Các cạnh này được đánh trọng số $-\epsilon$.

Gọi (i, j) là tọa độ của một nút trên đồ thị, trong đó i là chỉ số hàng, j là chỉ số cột. Các khoảng mở tương ứng với các cạnh có hướng từ nút $middle_{i-1,j}$ xuống nút $lower_{i,j}$ hoặc từ nút $middle_{i,j-1}$ lên nút $upper_{i,j}$, các cạnh này được đánh trọng số $-\sigma$. Cuối cùng, các khoảng đóng (gap closing) tương ứng với các cạnh có hướng từ nút $lower_{i,j}$ lên nút $middle_{i,j}$ hoặc từ nút $upper_{i,j}$ xuống nút $middle_{i,j}$, các cạnh này được đánh trọng số bằng 0 (Xem hình 2.7).

Đồ thị không chu trình mà ta vừa xây dựng có thể phức tạp, nhưng nó chỉ sử dụng $\mathcal{O}(n.m)$ cạnh với các chuỗi có độ dài n và m mà vẫn xây dựng được một dóng hàng tối ưu nhờ áp dụng mô hình phạt khoảng trống Affine. Gọi $low_{i,j}$, $middle_{i,j}$ và $upper_{i,j}$ là độ dài của các đường dài nhất từ source đến $(i, j)_{lower}$, $(i, j)_{middle}$ và $(i, j)_{upper}$. Sử dụng thuật toán quy hoạch động, đồ thị dóng hàng ba cấp được xác định bởi công thức truy hồi sau:

$$lower_{i,j} = max \begin{cases} \text{lower}_{i-1,j} - \epsilon \\ \text{middle}_{i-1,j} - \sigma \end{cases}$$

$$middle_{i,j} = max \begin{cases} \text{lower}_{i,j} \\ \text{middle}_{i-1,j} + score(v_i, w_j) \\ \text{upper}_{i,j} \end{cases}$$

$$upper_{i,j} = max \begin{cases} \text{upper}_{i,j_1} - \epsilon \\ \text{middle}_{i,j-1} - \sigma \end{cases}$$

Để dóng hàng địa phương, ta đưa thêm điều kiện 0 vào trong các quan hệ truy hồi của *lower*, *middle*, và *upper*. Điều này tương ứng với việc thêm các cạnh với trọng số bằng 0 vào 3 ma trận *lower*, *middle*, và *upper*. Điểm cuối của dóng hàng địa phương được xác định khi điểm cộng dồn đạt giá trị lớn nhất trên bất kì ma trận nào trong ba ma trận kể trên. Từ điểm cuối này kết hợp với ma trận lưu các con trỏ quay lui ta xác định được đường đi dài nhất của đồ thị (có điểm dóng hàng cao nhất) hay chuỗi con chung tối ưu giữa hai trình tự.

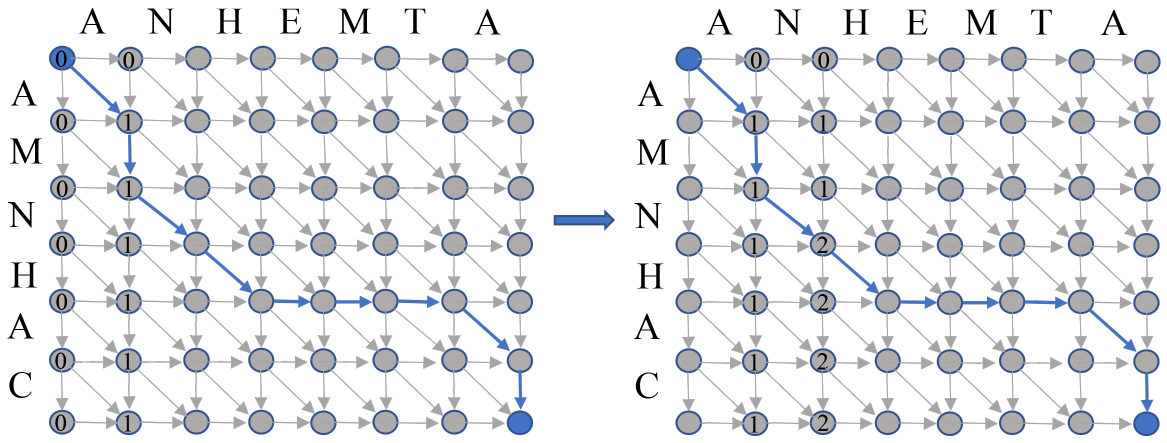
Thời gian chạy của thuật toán quy hoạch động để dóng hàng hai chuỗi có độ dài n và m tỷ lệ thuận với số cạnh trong đồ thị dóng hàng của chúng, là $\mathcal{O}(n.m)$. Vì chúng ta cần lưu trữ các tham chiếu quay lui nên bộ nhớ được yêu cầu bởi thuật toán này cũng là $\mathcal{O}(n.m)$.

2.2.2 Thuật toán tiết kiệm bộ nhớ

Trong phần này, chúng tôi sẽ trình bày một thuật toán dóng hàng chỉ với không gian lưu trữ $\mathcal{O}(n)$ với chi phí thời gian vẫn là $\mathcal{O}(n.m)$. Để đơn giản mà không mất tính tổng quát, chúng tôi sẽ xét trường hợp dóng hàng toàn bộ với mô hình phạt khoảng trống đều, tức là các ký tự "-" trong một gap bị phạt giống nhau là $-\sigma$.

Thuật toán tiết kiệm bộ nhớ sử dụng chiến lược *chia để trị*. Đây là chiến lược thường được áp dụng khi lời giải của một bài toán lớn được xây dựng từ các lời giải của các bài toán nhỏ hơn. Chiến lược chia để trị với hai giai đoạn: Giai đoạn chia là việc phân rã bài toán ban đầu thành các bài toán con nhỏ hơn có thể giải được; Giai đoạn trị là việc kết hợp các lời giải của các bài toán con để nhận được lời giải cho bài toán lớn ban đầu.

Ở đây chúng ta nhận thấy, để tìm một chuỗi trình tự chung dài nhất ta cần lưu điểm dóng hàng tại các nút của hai cột liên tiếp nhau j và $j+1$, các điểm của các cột ở trước j sẽ không được lưu. Như vậy, không gian lưu trữ chỉ là hai lần số nút của một cột, tức là $\mathcal{O}(n)$. Các con trỏ quay lui lúc này cũng chỉ được lưu từ cột j đến cột $j+1$ và không gian lưu trữ cũng chỉ là $\mathcal{O}(n)$ (Xem hình 2.8). Thuật toán tiết kiệm bộ nhớ với hai giai đoạn chia và trị được thiết kế như sau:



Hình 2.8: Thuật toán tiết kiệm không gian lưu trữ cho việc dóng hàng toàn bộ hai chuỗi v và w tương ứng là $AMNHAC$ và $ANHEMTA$. Ta chỉ lưu điểm dóng hàng các nút của hai cột liên tiếp, không gian lưu trữ chỉ là hai lần số nút của một cột, tức là $\mathcal{O}(n)$. Các con trỏ quay lui cũng chỉ được lưu trong phạm vi hai cột này, không gian lưu trữ của chúng cũng là $\mathcal{O}(n)$.

2.2.2.1 Giai đoạn chia (bài toán tìm cạnh giữa)

Cho các chuỗi $v = v_1...v_n$ và $w = w_1...w_m$, đặt $middle = [\frac{m}{2}]$. Cột giữa của đồ thị dóng hàng v, w là cột chứa tất cả các nút $(i, middle)$ với $0 \leq i \leq n$. Đường dài nhất từ source đến sink trong đồ thị dóng hàng phải đi qua nút giao của cột giữa này và một hàng nào đó và nhiệm vụ đầu tiên của chúng ta là tìm ra nút này mà chỉ sử dụng $\mathcal{O}(n)$ bộ nhớ. Trong hình 2.9, với phạm vi toàn bộ đồ thị ta tìm được $middle = 5$ và đường dóng hàng tối ưu đi qua cột giữa tại nút giữa $M_1(4, 5)$.

Nút giữa mà LCS đi qua có thể được tìm thấy mà không phải xây dựng LCS này trong đồ thị dóng hàng. Gọi một đường từ source tới sink là đường i nếu nó đi qua cột giữa ở hàng i. Đối với các i trong khoảng từ 0 đến n, ta cần tìm độ dài của đường i dài nhất. Nhận thấy rằng các đường đi từ source đến sink trong đồ thị dóng hàng sẽ chỉ đi qua phần trên bên trái của nút giữa và phần dưới bên phải của nút giữa. Do đó, ta sẽ tiếp tục tìm các nút giữa trong hai phần này của đồ thị, quá trình tiếp tục đến khi không thể phân chia thành các phần nhỏ hơn. Các phần còn lại của đồ thị không chứa LCS thì không tham gia vào quá trình tính toán. Do đó, thời gian tính toán của thuật toán chính bằng tổng diện tích của các phần chia:

$$n.m + \frac{n.m}{2} + \frac{n.m}{4} + ... < 2.n.m = \mathcal{O}(n.m)$$

Kết hợp với việc tìm nút giữa (middle node) ta có thể tìm luôn cạnh giữa (middle edge) thuộc LCS xuất phát từ nút giữa này. Khi tìm thấy cạnh giữa, thậm chí ta chỉ cần xét hai hình chữ nhật mà LCS phải đi ở hai bên của cạnh giữa. Bây giờ hai hình chữ nhật này chiếm ít hơn một nửa diện tích của đồ thị dóng hàng, đó là một lợi thế của việc chọn cạnh giữa thay vì nút giữa. Trong giai đoạn tiếp theo, ta cần thiết kế một thuật toán sao cho có thể kết hợp các cạnh từ các phần chia với các cạnh giữa để tạo thành một dóng hàng toàn bộ hai chuỗi.

2.2.2.2 Giai đoạn trị

Một đường dẫn dài nhất trong đồ thị dóng hàng được xây dựng cho một chuỗi con v_{top} ... v_{bottom} của v và một chuỗi con w_{left} ... w_{right} của w bằng cách đệ quy. Hàm **ME** (top, $bottom$, $left$, $right$) được xây dựng trước để trả về tọa độ i của nút giữa (i, j) và hướng cạnh giữa. Hướng của các cạnh giữa được lưu bởi các con trỏ quay lui (control), chúng có thể đi sang phải, xuống dưới hoặc theo đường chéo tùy thuộc vào việc cạnh giữa là ngang, dọc hoặc chéo. Dóng hàng không gian tuyến tính của chuỗi v và w được xây dựng bằng cách gọi đệ quy hàm **LSA** (0, n, 0, m). Trường hợp $left = right$ là dóng hàng của một chuỗi trống so với chuỗi $v_{top}...v_{bottom}$, trường hợp $top = bottom$ là dóng hàng của một chuỗi trống so với chuỗi w_{left} ... w_{right} (Thuật toán 3). Ví dụ xét thuật toán đệ quy xây dựng dóng hàng toàn bộ hai chuỗi v = $PLEASANTLY$ và w = $MEANLY$ (Xem

hình 2.9). Quá trình đệ quy bắt đầu diễn ra bên nhánh trái, sau khi suy biến về các trường hợp cơ bản quá trình này quay ngược trở lại và tiếp tục diễn ra bên nhánh phải. Ta tìm được các nút giữa M_1, M_2, M_3, M_4, M_5, M_6, M_7, M_8 và các cạnh giữa bắt đầu từ các nút này. Dóng hàng toàn bộ hai chuỗi thu được là $PLEASANTLY$ và $-MEA--N-LY$.

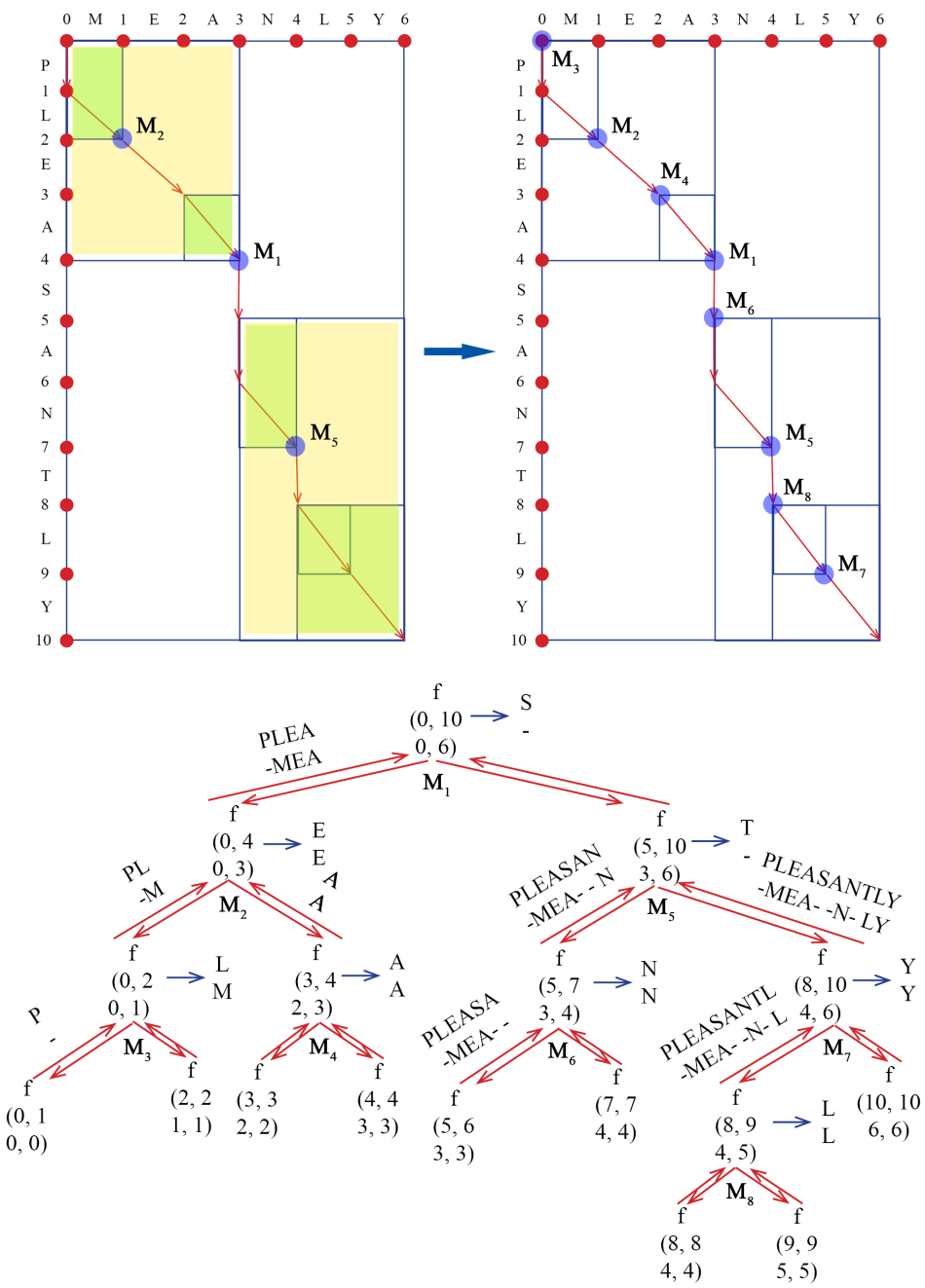
Algorithm 3 Linear Space Alignment

Input: v, w: Strings, SM: Scoring matrix, σ.
Output: AL: Alignment.
Initialisation: $top, bottom \leftarrow 0, n$; $left, right \leftarrow 0, m$.

LSA (top, $bottom$, $left$, $right$)
1: **if** $left = right$ **then**
2: **return** $v[top : bottom]$, "-".$(bottom - top)$
3: **end if**
4: **if** $top = bottom$ **then**
5: **return** "-".$(right - left)$, $w[left : right]$
6: **end if**
7: $middle = [(left + right)/2]$
8: $midNode, control =$ **ME** (top, $bottom$, $left$, $right$)
9: $A =$ **LSA** (top, $midNode$, $left$, $middle$)
10: **if** $control =$ "$down$" **then**
11: $midEdge = (v[midNode : midNode + 1]$, "-")
12: $midNode = midNode + 1$
13: **end if**
14: **if** $control =$ "$diagonal$" **then**
15: $midEdge = (v[midNode : midNode + 1], w[middle : (middle + 1)])$
16: $midNode = midNode + 1$
17: $middle = middle + 1$
18: **end if**
19: **if** $control =$ "$right$" **then**
20: $midEdge =$ ("-", $w[middle : middle + 1]$)
21: $middle = middle + 1$
22: **end if**
23: $B =$ **LSA** ($midNode$, $bottom$, $middle$, $right$)
24: $AL =$ **Combine** (A, $middleEdge$, B)
25: **return** $AL[0]$, $AL[1]$

2.2.3 Thuật toán tham lam cho dóng hàng đa trình tự

Cho trước một ma trận gồm các chuỗi DNA (chứa 4 kí tự A, C, T, G), hoặc các chuỗi protein (chứa 20 loại amino acid). Phương pháp dóng hàng tăng dần



Hình 2.9: Thuật toán chia để trị cho việc dóng hàng toàn bộ hai chuỗi v và w tương ứng là $PLEASANTLY$ và $MEANLY$. Ở bước đầu tiên, đồ thị dóng hàng được chia thành bốn phần, trong đó hai phần đồ thị có LCS đi qua ở hai bên của cạnh giữa bắt đầu từ điểm giữa M_1 (được tô màu) tham gia vào việc phân chia tiếp theo. Quá trình phân chia tiếp tục ở nhánh bên trái với các cạnh giữa bắt đầu từ các nút giữa M_2, M_3, và M_4. Khi suy biến về các trường hợp cơ bản thì thuật toán đệ quy quay ngược trở lại thực hiện quá trình trị và bắt đầu phân chia bên nhánh phải. Các cạnh giữa tiếp theo bắt đầu từ các nút giữa M_5, M_6, M_7, và M_8 phân chia phía phải thành các diện tích nhỏ hơn. Cuối cùng, giai đoạn trị bên nhánh phải đưa ra được kết quả dóng hàng là $PLEASANTLY$ và $-MEA--N-LY$.

(progressive alignment) cho dóng hàng đa trình tự được xây dựng với thuật toán tham lam. Ở đây, chiến thuật chọn thứ tự dóng hàng dựa trên cây hướng dẫn được thiết lập từ ma trận điểm của các cặp được dóng hàng:

Bước 1: Xây dựng ma trận trọng số dựa trên dóng hàng theo cặp.

- Các cặp được dóng hàng và tính điểm dựa trên tỷ số giữa tổng các khớp và độ dài chuỗi.
- Xây dựng ma trận tương tự (ma trận trọng số) với các trọng số tương ứng là điểm của các trình tự. Như vậy, đây là ma trận vuông với các cạnh là các trình tự.

Bước 2: Xây dựng cây dẫn hướng (guide tree) dựa trên thuật toán phân cụm theo thứ bậc.

- Áp dụng thuật toán phân cụm theo thứ bậc với dữ liệu là ma trận tương tự.
- Áp dụng ClustalW (phân cụm theo trọng số) với phương pháp kết hợp hàng xóm (neighbor-joining method).

Bước 3: dóng hàng tăng dần được dẫn hướng bởi cây dẫn hướng.

- Chọn hai dóng hàng trình tự có điểm cao nhất.
- Dựa trên cây dẫn hướng tiếp tục thêm các trình tự mới vào và dóng hàng dựa trên các trình tự cũ.
- Thêm các kí tự trống cần thiết.

2.2.4 Tính điểm cho dóng hàng đa trình tự

Việc lựa chọn hàm cho điểm có thể ảnh hưởng mạnh đến chất lượng của nhiều dóng hàng. Ta có thể tính điểm bằng tổng các khớp của một cột, trong đó một cột là khớp nếu tất cả các ký tự trong cột giống nhau. Hoặc có thể tính điểm theo Entropy, với p_x là tần số của một kí tự x trong một cột của dóng hàng đa trình tự, Entropy của mỗi cột được tính như sau:

$$Entropy\,of\,Column = -\sum_{X=A,T,G,C} P_X log P_X$$

Điểm Entropy bằng tổng các entropy của tất cả các cột. Nếu các cột càng giống nhau thì điểm Entropy càng nhỏ, do đó để tìm kiếm một dóng hàng đa trình tự tối ưu ta sẽ đi tìm một dóng hàng đa trình tự có điểm Entropy nhỏ nhất. Ngoài ra, ta còn có thể tính điểm dóng hàng đa trình tự theo tổng điểm các cặp (SP-Score). Dóng hàng theo cặp của các trình tự a_i, a_j có thể được suy ra từ

dóng hàng k trình tự. Ký hiệu điểm của dóng hàng theo cặp này là $s^*(a_i, a_j)$, tổng điểm của các cặp được tính như sau:

$$s(a_1, ..., a_k) = \sum_{i,j} s^*(a_i, a_j)$$

2.3 Thực nghiệm thuật toán

2.3.1 Thuật toán song song với Golang

Với lý thuyết về dóng hàng dựa trên chuyển dạng Burrows-Wheeler ở phần 2.1, chúng tôi đã triển khai thuật toán bằng ngôn ngữ Go (Golang), một ngôn ngữ hỗ trợ mạnh cho việc tính toán song song và đồng thời. Mục đích của thực nghiệm thuật toán là để hiểu rõ hơn về thuật toán cũng như cách thiết lập các tham số cần thiết lập ban đầu.

Golang là ngôn ngữ lập trình hỗ trợ tự động khai thác sự hoạt động của các nhân (core) máy tính mà không phụ thuộc vào việc cấp phát của hệ điều hành. Các threads trên các nhân được triển khai bởi các goroutine. Chúng có thể làm việc cùng nhau tại chính xác một thời điểm hoặc đợi nhau trong một hàng đợi. Mặt khác, các goroutine hoạt động trên nguyên tắc không chia sẻ biến nên giao tiếp của chúng được đồng bộ hóa bằng các kênh đệm (buffered channel) hoặc các kênh không đệm (unbuffered channel). Ngoài ra, ta có thể khởi tạo số lượng goroutine lớn hơn hoặc bằng số các logical processor của máy tính. Khi chúng bằng nhau thời gian thực thi bằng thời gian chạy của goroutine chậm nhất. Ngược lại, khi số lượng các goroutine lớn hơn, chúng được đồng bộ hóa để tối ưu thời gian và đảm bảo không có nhân nào nhàn rỗi.

Thuật toán tuần tự (Algorithm 2) được thiết kế lại cho việc áp dụng kỹ thuật đồng thời của Golang. Ở đây, nếu một dãy được sử dụng để lưu các đoạn cập nhật [*top*, *bottom*] sau mỗi vòng lặp trong thuật toán tuần tự thì một số lượng phù hợp các goroutine được sử dụng để thực hiện việc cập nhật này trong thuật toán đồng thời (Algorithm 4). Các kênh đệm cũng được khởi tạo và hoạt động như một trung gian để nhận và gửi dữ liệu giữa các goroutine. Chú ý rằng, các kênh đệm có thể gửi thông tin tới các goroutine khác nhau tại cùng một thời điểm, nhưng chúng không lưu dữ liệu đã gửi. Bởi vậy, ta không cần xóa các đoạn [*top*, *bottom*] cũ sau khi tính toán. Đây cũng là cơ sở cho việc đồng bộ hóa quá trình làm việc của các goroutine thông qua các kênh đệm.

- Dòng 1: Vòng lặp có ý nhĩa thiết lập các goroutine một cách đồng thời với số lượng cho trước.
- Dòng 3: Các goroutine nhận dữ liệu của các đoạn [*top*, *bottom*] tại cùng một thời điểm từ kênh đệm, dữ liệu này mất đi ngay sau đó trên kênh chứa nó.

- Dòng 23: Dữ liệu của đoạn [*top*, *bottom*] mới được gửi trở lại kênh đệm trước đó.

- Dòng 30: Dữ liệu về đoạn [*top*, *bottom*] cuối cùng được gửi tới kênh đệm, nơi mà ta có thể tìm thấy các vị trí khớp của mẫu trên chuỗi tham chiếu.

- Dòng 33: Nếu kênh đệm rỗng, ta cần thoát vòng lặp để tránh tình trạng "deadlock" khi dữ liệu được gọi bởi một goroutine từ một kênh rỗng. Điều đó cũng có nghĩa là không có trình tự khớp xấp xỉ nào được tìm thấy.

Algorithm 4 Concurrency Approximate Pattern Matching

Input: BWT, PAT, PSA, FO, C, D, W, GS: Number of Goroutines.
Output: ML, NM.

Initialisation:
$posChan, tbtUpchan \leftarrow$ buffered channel,
$tbtUpChan, gs \leftarrow (1, |BWT|, W, |PAT|), 4$

CAPM (BWT, PAT, PSA, FO, C, D, W, GS)

1: **for** $g = 1$ to GS **do**
2: **go func()**
3: **for** Receive tbt from range of $tbtUpChan$ **do**
4: $t, bt, d, id \leftarrow tbt[1], tbt[2], tbt[3], tbt[4]$
5: $apr \leftarrow (A, C, G, T)$
6: **if** $id \neq 1$ **then**
7: $PAT, symbol \leftarrow PAT[1 : id], PAT[id]$
8: **for** $g \leftarrow 1$ to gs **do**
9: **if** $tbt = (0, 0, 0, 0)$ **then**
10: Send $(0, 0, 0, 0)$ to $tbtUpChan$
11: **break**
12: **end if**
13: **go func(g)**
14: $aprS, nD \leftarrow apr[g], d$
15: $nT, nB \leftarrow t, bt$
16: **if** $nD \geq D[id]$ **then**
17: **if** $aprS \neq symbol$ **then**
18: $nD \leftarrow nD - 1$
19: **end if**
20: $nT \leftarrow FO(aprS) + C(aprS, t - 1)$
21: $nB \leftarrow FO(aprS) + C(aprS, bt) - 1$
22: **if** $nT \leq nB \cap nD \geq 0$ **then**
23: Send $(nT, nB, nD, id - 1)$ to $tbtUpChan$
24: **end if**
25: **end if**
26: **end go func(g)**
27: **end for**
28: **else**
29: **if** $tbt \neq (0, 0, 0, 0)$ **then**
30: Send tbt to $posChan$
31: **end if**

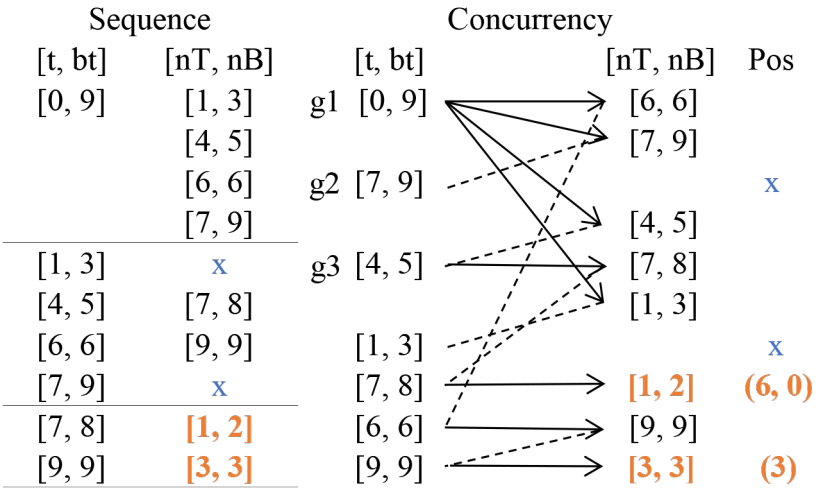
```
32:       end if
33:       if |tbtUpChan| = 0 then
34:         for i = 1 to GS do
35:           Send (0, 0, 0, 0) to tbtUpChan
36:         end for
37:         break
38:       end if
39:     end for
40:     end go func()
41: end for
42: if |posChan| ≠ 0 then
43:     close(posChan)
44:     Receive nT, nB, and nD from range of posChan
45:     ML, NM ← values of PSA[nT: nB], W − nD
46: end if
47: return ML, NM
```

Ở dòng 8, các goroutine mới (khác các goroutine ban đầu được khởi tạo ở dòng 1) nhận nhiệm vụ cập nhật các đoạn [*top*, *bottom*] mới tương ứng với các ký tự thuộc tập $\{A, C, G, T\}$ xuất phát từ đoạn [*top*, *bottom*] trước đó. Vì chỉ có 4 phần tử thuộc tập $\{A, C, G, T\}$ nên không cần quá nhiều goroutine, tránh trường hợp chúng phải đợi nhau, số lượng các goroutine được chọn bằng số các phần tử. Sau khi các goroutine này hoàn thành nhiệm vụ của mình, các goroutine ban đầu tiếp tục phần việc còn lại. Trong thuật toán này, các goroutine con được tạo trong một goroutine mẹ, mặc dù chúng làm các công việc khác nhau nhưng các goroutine mẹ vẫn phải đợi các goroutine con của nó hoàn thành công việc. Nếu ta so sánh các goroutine con với các goroutine mẹ khác mà không chứa nó và làm việc đồng thời với nó thì rõ ràng chúng làm các công việc khác nhau một cách đồng thời.

Quá trình làm việc của các goroutine khá phức tạp, do đó dễ phát sinh lỗi "all goroutines are asleep - deadlock!" nếu một goroutine nào đó gọi dữ liệu từ một kênh rỗng. Để tránh điều này, ta cần gửi vào kênh đệm số phần tử (0, 0, 0, 0) bằng với số goroutine mẹ như dòng 35. Các phần tử này không tham gia vào quá trình cập nhật các đoạn [*top*, *bottom*] và bị loại bỏ khỏi kết quả nếu chúng xuất hiện.

Quay trở lại với ví dụ khớp xấp xỉ mẫu ATC với chuỗi$ATCATGATC$ (Xem hình 2.4), ta sẽ quan sát quá trình cập nhật các đoạn [*top*, *bottom*] với thuật toán tuần tự và thuật toán đồng thời (Xem hình 2.10). Với thuật toán tuần tự, chúng ta bắt đầu tìm kiếm trên cả chiều dài chuỗi tương ứng với đoạn [0, 9]. Vì quá trình tìm kiếm dựa trên việc so sánh các ký tự của mẫu với các ký tự của chuỗi BWT, các đoạn mới được tìm thấy lần lượt theo thứ tự không đổi [1, 3],

[4, 5], [6, 6], và [7, 9]. Đối với thuật toán đồng thời, chúng ta không biết trước được thứ tự này. Khi một goroutine mẹ nào đó nhận nhiệm vụ tính toán với đoạn [0, 9] thì các goroutine con của nó tìm kiếm các đoạn mới tương ứng với các ký tự trong BWT. Chú ý rằng sự đồng thời ngăn chúng ta biết ký tự nào được chọn trước trong thuật toán tìm kiếm ngược, đồng nghĩa với việc ta không biết đoạn mới nào được tìm ra đầu tiên. Vì vậy, mỗi lần thực hiện thuật toán sẽ cho ra một thứ tự gần như ngẫu nhiên, trong ví dụ này, thứ tự của các đoạn mới là [6, 6], [7, 9], [4, 5], và [1, 3].



Hình 2.10: Xem xét quá trình khớp xấp xỉ mẫu ATC với chuỗi $ATCATGATC$ với thuật toán tuần tự và thuật toán đồng thời. Với thuật toán đồng thời, các goroutine g1, g2, và g3 làm việc cùng nhau nhưng bắt đầu tại các thời điểm khác nhau, tương tự cho các goroutine sau đó.

Ta hãy xem xét sự làm việc của chỉ các goroutine mẹ. Khi goroutine 1 tìm được một đoạn mới [7, 9], đó cũng là thời điểm cho goroutine 2 bắt đầu các tính toán tiếp theo với đoạn này giống như công việc của goroutine 1. Trong khi goroutine 1 tiếp tục tìm kiếm, goroutine 2 kết thúc công việc của nó vì số các không khớp vượt quá 1. Ngay sau khi goroutine 1 tìm ra đoạn mới [4, 5] thì goroutine 3 bắt đầu nhận nhiệm vụ với đoạn mới này. Và trong khi goroutine 1 vẫn tiếp tục làm việc, goroutine 3 tìm được đoạn [7, 8]. Phiên làm việc của goroutine 1 kết thúc sau khi nó cập nhật dữ liệu của đoạn [1, 3], bây giờ nó có thể làm việc với một đoạn khác hoặc đợi đến lượt mình. Như vậy, các goroutine mẹ làm việc đồng thời, nhưng không bắt đầu và kết thúc tại cùng một thời điểm. Các goroutine được đồng bộ để đảm bảo các công việc diễn ra một cách nhanh nhất có thể. Điều này cũng giúp tránh trạng thái nhàn rỗi của bất kỳ một nhân nào trong suốt quá trình thực thi.

Trên thực tế, số các trình tự được xuất ra từ máy giải trình tự rất lớn, nhưng số lượng đó không thay đổi trong suốt quá trình chạy thuật toán. Khối lượng

công việc có thể khác nhau ở từng trình tự nhưng khá tương đồng nếu ta tính trên các phần chia bằng nhau của tổng các trình tự. Dạng khối lượng công việc này còn được gọi là CPU-bound, nếu mỗi goroutine tính toán một phần chia thì chúng có thể làm việc song song, độc lập, cùng thời điểm, và không phát sinh trạng thái chờ. Ở đây số lượng các goroutine được thiết lập bằng đúng số logical processor (thường bằng hai lần số lượng các nhân máy tính) (Xem thuật toán 5). Toàn bộ mã nguồn của các thuật toán đồng thời có thể xem tại https://github.com/nhanta/BI_Golang_BWT.

Algorithm 5 Concurrency Multiple Approximate Pattern Matching

Input: BWT, $PATS$, PSA, FO, C, D, W, GS.
Output: ML, NM.
Initialisation : $GS \leftarrow$ number of logical processors

CMAPM $(BWT, PATS, PSA, FO, C, D, W, GS)$
1: $st \leftarrow |PATS|/GS$
2: **for** $g \leftarrow 1$ to GS **do**
3: **go func(g)**
4: $start \leftarrow g * st$
5: $end \leftarrow start + st$
6: **if** $g = GS$ **then**
7: $end \leftarrow |PATS|$
8: **end if**
9: **for** $i \leftarrow start$ to end **do**
10: $PAT \leftarrow PATS[i]$
11: $ML, NM \leftarrow$ MP$(BWT, PAT, PSA, FO, C, D, W, GS)$
12: Send ML, NM to ch
13: **end for**
14: **end go func()**
15: **end for**
16: **close** ch
17: Receive ML, NM from ch
18: **return** ML, NM

2.3.2 Thực nghiệm

2.3.2.1 Dữ liệu

Đại dịch COVID-19 bắt đầu từ cuối năm 2019 đã trở thành cuộc khủng hoảng sức khỏe cộng đồng lớn nhất từ đó đến nay. Nguyên nhân là do sự bùng phát của virus SARS-CoV-2 thuộc họ Coronavirrus, virus này gây ra "Hội chứng hô hấp

cấp tính nghiêm trọng" (SARS) [Com15]. SARS-CoV-2 thâm nhập vào tế bào người thông qua enzyme chuyển Angiotensin 2 (ACE-2) gắn vào màng tế bào của các tế bào ở phổi, động mạch, tim, thận và ruột. Sau đó, nó tấn công các tế bào trình diện kháng nguyên (APC) của hệ thống miễn dịch, đồng thời làm giảm các tế bào T (một loại Lymphocyte - một phân lớp của Bạch cầu) [YFK20]. Nguyên nhân chính dẫn đến tốc độ đột biến nhanh của virus là do nó sở hữu vật chất di truyền là RNA thay vì DNA, quá trình sao chép của RNA dễ bị lỗi hơn so với quá trình sao chép DNA. Do đó SARS-CoV-2 nói riêng và coronavirus nói chung thay đổi nhanh chóng và khó lường.

Thực nghiệm sử dụng bộ lưu trữ trình tự SARS-CoV-2 từ máy giải trình tự Illumina được công bố vào 28 tháng 7 năm 2020 bởi KwaZulu-Natal Research Innovation and Sequencing Platform [1]. Tệp định dạng FASTQ bao gồm 436,610 paired-end reads được chuyển sang định dạng sang fasta. Bộ trình tự này được đem khớp với bộ gen tham chiếu SARS-CoV-2 dài 24748 bp được công bố bởi Fan Wu et al. (2020)[2] [Wu+20].

2.3.2.2 Tham số và đầu vào

Chuyển dạng Burrows-Wheeler, mảng hậu tố, và *ma trận điểm kiểm tra* được tính trước khi thực hiện thuật toán. Ta thay đổi khoảng cách c giữa cấc giá trị trong *mảng hậu tố* và khoảng cách k giữa thứ tự trong *ma trận điểm kiểm tra* để đánh giá mức độ thay đổi giữa thời gian chạy và bộ nhớ được sử dụng. Tiếp theo, ta chạy thuật toán với các giá trị khác biệt và so sánh kết quả với kết quả thu được từ công cụ BWA-MEM. Các kết quả giống nhau khi các tham số của công cụ được thiết lập như sau:

- T (Không xuất các dóng hàng với điểm thấp hơn một số nguyên cho trước) = 0.
- k (các khớp ngắn hơn một số nguyên cho trước được loại bỏ) = 0.

2.3.2.3 Kết quả

Thuật toán được triển khai trên máy ảo của nền tảng Google Cloud với cấu hình 8 vCPUs và 52 GB bộ nhớ. Với *mảng hậu tố một phần*, không có sự khác biệt đáng kể về thời gian chạy và không gian làm việc khi các giá trị c và k khác nhau. Với *ma trận điểm kiểm tra*, thời gian chạy đo được chậm hơn 104.5 lần và bộ nhớ yêu cầu lớn hơn 6.5 lần khi c và k bằng 1 so với khi chúng cùng bằng 100

[1] https://sra-pub-sars-cov2.s3.amazonaws.com/sra-src/SRR12338312/KPCOVID-345_S81_L001_R1_001.fastq.gz.1

[2] https://www.ncbi.nlm.nih.gov/nuccore/NC_045512.2

(Xem bảng 2.1). Tuy nhiên, ta chỉ cần chạy thuật toán một lần để tìm *ma trận điểm kiểm tra*, chúng được lưu lại để tái sử dụng nhiều lần.

Với ngưỡng khác biệt là 3, ta tìm được 367,946 trình tự khớp với bộ gen tham chiếu (Xem bảng 2.2). Các trình tự khớp có thể gồm cả chuỗi thuận và chuỗi nghịch được gán nhãn 0 và 16 tương ứng. Sau khi dóng hàng, ta có thể gọi ra các nucleotide khớp và không khớp với bộ gen tham chiếu tại các vị trí khác nhau (Xem bảng 2.3).

Bảng 2.1: Mảng hậu tố một phần và ma trận điểm kiểm tra

	Mảng hậu tố một phần		**Ma trận điểm kiểm tra**	
c, k	***Time (ms)***	***Memory (MiB)***	***Time (ms)***	***Memory (MiB)***
1	18.64	2	302.09	13
30	15.26	2	10.59	4
60	15.18	2	4.79	3
100	14.00	2	2.89	2

Bảng 2.2: Tổng các trình tự khớp với các ngưỡng khác biệt khác nhau

D	**Time** (s)	**Memory (MiB)**	**Total**
0	178.56	96	242 943
1	234.31	270	354 930
2	436.03	268	365 724
3	1229.37	357	367 946

Bảng 2.3: Kết quả dóng hàng trình tự với ngưỡng khác biệt là 3

Tên	**Hướng**	**Vị trí**	**Chuỗi không khớp**
100062/2	16	13 030	0C0C0A248
100104/1	0	14 275	10G68T52T50
100160/1	0	9679	1C0A0C88
100160/2	16	9679	1C0A0C88
100223/2	16	14 314	0C67T24T157
100269/1	0	14 358	49T51G55G67
...			...

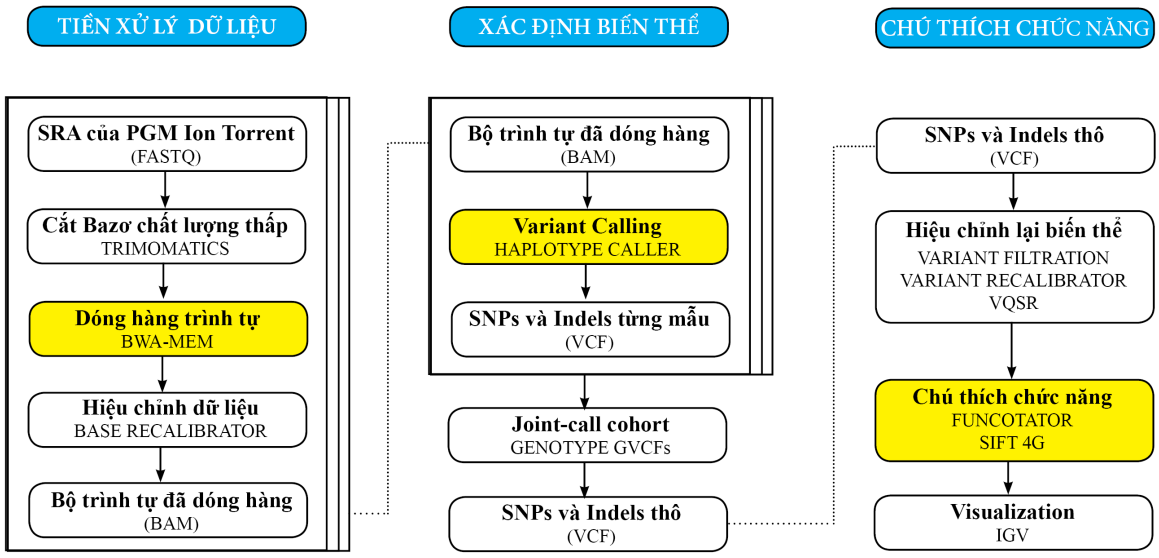
CHƯƠNG 3

ỨNG DỤNG THUẬT TOÁN TRONG DỰ ĐOÁN BIẾN THỂ GEN

Ở chương này, ta áp dụng thuật toán dóng hàng trình tự dựa trên chuyển dạng Burrows-Wheeler và Smith-Waterman để xác định các biến thể gen, đồng thời kết hợp với các phương pháp bổ xung để xử lý dữ liệu và tìm ra sự ảnh hưởng của các biến thể đến chức năng của protein. Phạm vi tìm kiếm được trình bày chi tiết hơn ở phần 3.4. Quy trình làm việc được chia ra thành 3 giai đoạn: tiền xử lý dữ liệu, xác định biến thể, và chú thích chức năng (Xem hình 3.1). Chú ý rằng, thuật toán dựa trên chuyển dạng Burrows-Wheeler được sử dụng trong phần mềm BWA-MEM dùng để dóng hàng bộ dữ liệu được xuất ra từ máy giải trình tự với bộ gen tham chiếu. Từ đó ta có được bộ trình tự với các tọa độ khớp và các vị trí biến thể ban đầu. Mặt khác, thuật toán Smith-Waterman được sử dụng trong phần mềm Haplotype Caller ở bước dóng hàng lại mỗi Haplotype với Haplotype tham chiếu. Ngoài ra, thuật toán này còn được tích hợp trong phần mềm SIFT 4G ở bước tìm trình tự tương đồng dựa trên cơ sở dữ liệu lớn về protein [Vas+16]. Các biến thể cuối cùng được hiển thị thông qua phần mềm Integrative Genomics Viewer (IGV) [TRM13].

3.1 Dữ liệu

Trong luận văn này chúng ta sẽ tìm những biến thể gen liên quan đến bệnh tâm thần phân liệt (Schizophrenia), một chứng rối loạn tâm thần nghiêm trọng. Về biểu hiện lâm sàng, người bệnh thường phải chịu các ảo giác, ảo tưởng, họ có suy nghĩ và hành vi cực kỳ rối loạn làm suy giảm hoạt động hàng ngày và có thể dẫn đến tàn phế suốt đời. Về phương diện di truyền, tâm thần phân liệt là một bệnh đa gen với yếu tố di truyền cao [Sek+16]. Nghiên cứu của Ripke và các cộng sự năm 2014 chỉ ra 108 vùng trên các nhiễm sắc thể liên quan đến bệnh này với dữ liệu GWAS gồm 9.5 triệu biến thể sau quá trình kiểm soát chất lượng [Rip+14]. Vào năm 2019, nhóm nghiên cứu thuộc trường Y khoa Icahn ở Mount



Hình 3.1: Quy trình làm việc xác định ảnh hưởng của biến thể gen gồm ba giai đoạn. **Giai đoạn 1: tiền xử lý dữ liệu**. Thuật toán dóng hàng trình tự dựa trên chuyển dạng Burrows-Wheeler được sử dụng để khớp bộ trình tự lưu trữ với bộ gen tham chiếu. **Giai đoạn 2: xác định biến thể**. Thuật toán Smith-Waterman được sử dụng để dóng hàng lại các haplotype ở bước gọi biến thể. **Giai đoạn 3: chú thích chức năng**. Lọc ra các biến thể và đánh giá mức độ ảnh hưởng của chúng đến protein, từ đó đưa ra các gen có khả năng bị đột biến.

Sinai đã công bố 413 gen thuộc 13 vùng của não có liên quan đến bệnh tâm thần phân liệt, phương pháp được sử dụng là máy học dựa trên dữ liệu GWAS của 40,299 mẫu bệnh và 65,264 mẫu thường [Huc+19]. Đến tháng 3 năm 2020, Joshua Gordon thông báo trên The National Institute of Mental Health (NIMH) rằng các nhà nghiên cứu từ quỹ NIMH đã xác định được hơn 250 vị trí trên bộ gen góp phần vào nguy cơ chung của bệnh tâm thần phân liệt [1].

Dữ liệu ở đây được lấy từ nghiên cứu về giải trình tự exome ở bệnh nhân tâm thần phân liệt có mức độ đồng hợp tử cao xác định các đột biến mới và cực kỳ hiếm trong đường chuyển hóa GABA/Glutamatergic [Gia+17]. 7 mẫu bệnh được lựa chọn thuộc về những người Italia có các vùng đồng hợp tử dài (runs of homozygosity, viết tắt là ROH) mà hai allele giống hệt nhau về nguồn gốc. Đây là lý do ta áp dụng phương pháp tìm đột biến mầm (germline mutations) cho các mẫu bệnh. Dữ liệu bao gồm 7 bộ lưu trữ trình tự đọc (Sequence Read Archive, viết tắt là SRA) được xuất ra từ máy giải trình tự Ion Torrent. Các trình tự thuộc loại trình tự đơn (single-end reads) tổng hợp từ thư viện NGS được chuẩn bị bằng phương pháp khuếch đại trình tự (Amplicon) (Xem bảng

[1] https://www.nimh.nih.gov/about/director/messages/2020/piecing-together-the-gentic-puzzle-of-schizophrenia.shtml

3.1). Dữ liệu SRA được dóng hàng với bộ gen tham chiếu GRCh13.p13 được lấy từ National Center for Biotechnology Information (NCBI). Đây là bộ gen tham chiếu của người có tổng chiều dài 3,099,706,404 bp được cập nhật vào ngày 28 tháng 2 năm 2019 [2].

Bảng 3.1: Thống kê dữ liệu của 7 bộ lưu trữ trình tự các đồng hợp tử bệnh tâm thần phân liệt. Dữ liệu các mẫu dạng PGM Ion Torrent được công bố trên NCBI.

Tên	Kích thước(GB)	Tổng trình tự	Chiều dài	%GC
SRR5344691	5.2	53915205	8-378	49
SRR5344690	2.0	22362189	12-378	51
SRR5344689	2.4	26819319	12-378	51
SRR5344688	2.9	33194464	12-377	51
SRR5344687	3.9	45684724	8-378	50
SRR5344686	3.8	43831694	12-378	50
SRR5344685	3.4	39248833	8-378	50

3.2 Tiền xử lý dữ liệu

3.2.1 Kiểm tra chất lượng

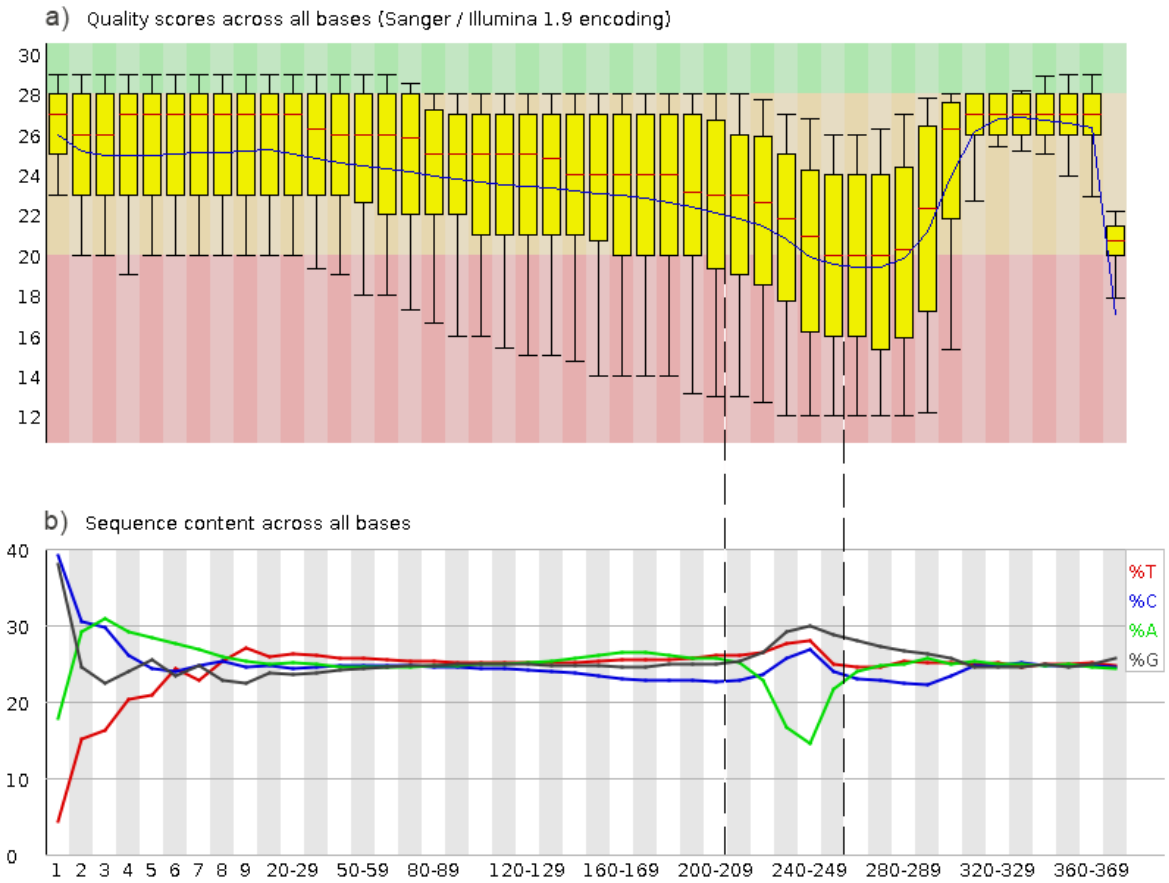
3.2.1.1 Điểm chất lượng trên mỗi vị trí nucleotide

Đối với các máy giải trình tự nói chung, các bazơ có điểm chất lượng giảm dần về phía cuối mỗi trình tự vì cường độ tín hiệu huỳnh quang thoái hóa ở cuối mỗi chu kỳ (signal decay). Ngoài ra, tín hiệu huỳnh quang cũng có thể mờ do bị phân kỳ (signal phasing) khi mà chất huỳnh quang ở các nucleotide chỉnh sửa không bị loại bỏ hoàn toàn. Đối với mẫu SRR5344691, điểm chất lượng bị giảm ngay từ bazơ thứ 20 và thấp hơn 20 từ bazơ 240 đến 289 (xem hình 3.2, a) có thể do một số lỗi khác như:

- Quá tải về số cụm (overclustering): số cụm (xem lại khái niệm về cụm ở phần 1.2.2) trên flow cell quá nhiều dẫn đến khoảng cách giữa các cụm nhỏ và sự chồng chéo tín hiệu. Hai cụm có thể bị nhầm lẫn thành một cụm với các tín hiệu huỳnh quang hỗn hợp. Hiện tượng này gây ra điểm chất lượng thấp trên toàn bộ một read.
- Sự cố thiết bị: có thể gây ra sự sụt giảm đột ngột về chất lượng hoặc một tỷ lệ lớn các read có chất lượng thấp.

[2]https://www.ncbi.nlm.nih.gov/assembly/GCF$_0$00001405.39

Các mẫu còn lại trong bộ dữ liệu này đều có hiện tượng giảm chất lượng tại vùng tương tự như mẫu SRR5344691, do đó một phương pháp chung sẽ được đưa ra để xử lý vấn đề này trong phần 3.2.2.



Hình 3.2: a) Biểu đồ hộp cho điểm chất lượng trên mỗi nucleotide của các trình tự thuộc mẫu SRR5344691. Trục hoành là các vị trí nucleotide trên các trình tự, trục tung là điểm chất lượng. Đối với mỗi hộp, giá trị nhỏ nhất ở 10% tổng điểm dữ liệu, giá trị lớn nhất ở 90% tổng điểm dữ liệu. Đường xanh là trung bình điểm chất lượng của các nucleotide với các vị trí khác nhau trên các trình tự. Biểu đồ còn được chia thành 3 vùng. Vùng xanh: điểm chất lượng tốt (từ 28 đến 38), vùng cam: điểm chất lượng chấp nhận được (từ 20 đến 27), vùng hồng đậm: điểm chất lượng kém (từ 0 đến 19). Điểm chất lượng trung bình bị giảm dần trên các vị trí từ 20 đến 289, sau đó nó lại tăng dần lên. Tuy nhiên ở các vị trí cuối của mỗi trình tự điểm chất lượng trung bình lại giảm. b) Biểu đồ dạng đường thành phần GC của các trình tự thuộc mẫu SRR5344691. Trong khoảng 12 bazơ đầu tiên, thành phần GC có sự thiên lệch, đây là điều bình thường. Tuy nhiên, số lượng A và T chênh lệch khá lớn ở đoạn giữa của biểu đồ có liên quan đến sự sụt giảm chất lượng của các bazơ từ 209 đến 259, đây là lỗi cần được xử lý.

3.2.1.2 Thành phần GC trên các bazơ

Như ta đã biết trong liên kết giữa hai sợi xoắn đơn DNA, A của sợi này liên kết với T của sợi kia, G của sợi này liên kết với C của sợi kia (xem lại phần 1.1.1). Do đó, số lượng A bằng số lượng T, số lượng G bằng số lượng C. Tuy nhiên, ở vị trí từ 10 đến 12 bazơ đầu tiên của read thường có sự thiên lệch (bias) của thành phần GC bởi cách chuẩn bị thư viện (xem lại khái niệm thư viện ở phần 1.2.2) trước khi giải trình tự. Lỗi kỹ thuật này không thể giải quyết được bằng cách cắt đi các bazơ trong quá trình tiền xử lý dữ liệu, nhưng nó cũng thường không ảnh hưởng xấu đến phân tích hạ nguồn (downstream analysis). Ở các bazơ cuối của các read, nếu xuất hiện bias thì nguyên nhân có thể do nhiễm bẩn adapter (xem lại phần 1.2.2). Nếu sự thay đổi thành phần GC ở đoạn giữa tương quan với việc giảm chất lượng trình tự thì đây là lỗi cần được xử lý. Tiếp tục với mẫu SRR5344691, đoạn giữa của biểu đồ thành phần GC cho thấy sự chênh lệch khá lớn giữa số lượng A so với T (xem hình 3.2, b), sự thay đổi thành phần GC này rõ ràng có liên quan đến sự sụt giảm chất lượng của các bazơ từ 209 đến 259. Hiện tượng này cũng sảy ra với tất cả các mẫu còn lại.

3.2.1.3 Phần trăm trình tự trùng lặp

Một trình tự trên một read bị trùng lặp (duplicate) khi ta tìm thấy một trình tự giống nó trên một read khác. Việc khử trùng lặp giúp ta giúp ta xem xét hiệu quả về mặt kinh tế khi phần trăm trình tự sau khi khử trùng lặp càng lớn thì khả năng đem lại cho chúng ta thêm thông tin càng cao. Với mẫu SRR5344691, ta giữ lại được 85.68% trình tự sau khi khử trùng lặp. Hơn nữa, có đến 79.79% trình tự duy nhất trong dữ liệu ban đầu , và 93.12% trình tự duy nhất trong dữ liệu sau khi khử trùng lặp (xem hình 3.3). Việc có ít trình tự trùng lặp trong dữ liệu này có thể do loại trình tự được xem xét là single-end read. Các mẫu còn lại có mức độ trùng lặp trình tự thấp tương tự như SR5344691, điều này cho thấy hiệu quả cao về mặt kinh tế. Việc loại bỏ trùng lặp khỏi dữ liệu còn phụ thuộc vào loại trùng lặp. Trùng lặp có thể không ảnh hưởng hoặc ảnh hưởng đến các phân tích về sau:

- Trùng lặp không ảnh hưởng: có nguyên nhân từ việc giải trình tự một vùng nhỏ của bộ gen (ví dụ giải trình tự nhắm mục tiêu) nơi mà không có nhiều những read khác nhau, đồng thời với việc tăng độ sâu bao phủ (depth of coverage) để tạo ra nhiều read hơn thì sự trùng lặp chỉ làm tăng độ bao phủ.

- Trùng lặp có ảnh hưởng: trong trường hợp thư viện chuẩn bị giải trình tự chứa các DNA kém chất lượng, các fragment có thể được sao chép để làm tăng chất lượng của DNA nhờ kỹ thuật PCR, quá trình này gọi là khuếch đại (amplification). Nếu sự trùng lặp bắt nguồn từ quá nhiều khuếch đại thì nó có thể gây ra sự sai lệch. Ví dụ, có một lỗi của máy giải trình tự trên một đoạn DNA, chúng sẽ được nhân lên trong quá trình sao chép. Như vậy,

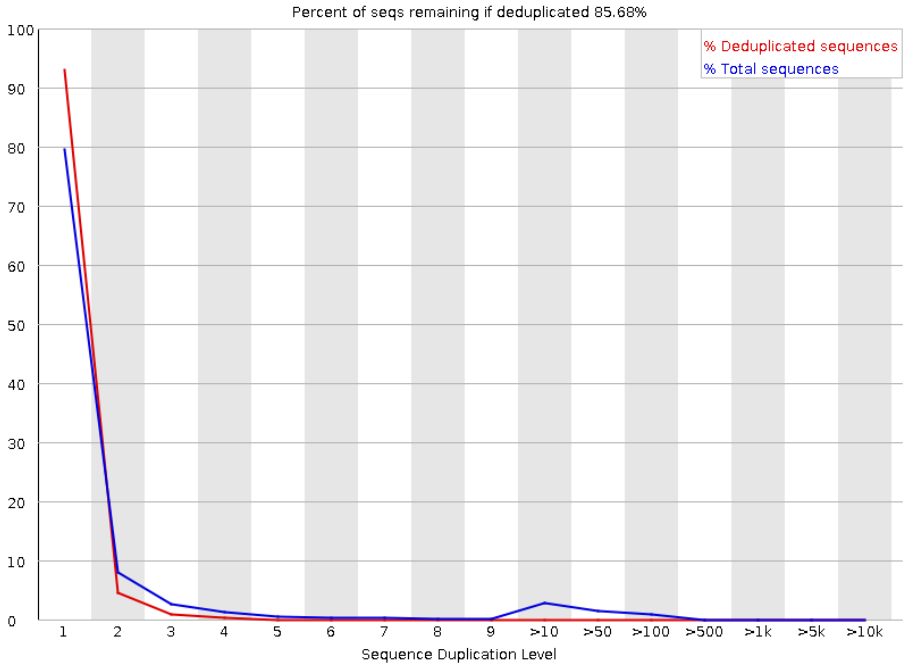
tất cả các trùng lặp sẽ bao gồm lỗi kỹ thuật này. Khi ta tiến hành dóng hàng, các lỗi của máy giải trình tự có thể bị hiểu nhầm là các biến thể. Các trùng lặp kiểu này cần được loại bỏ, khi đó chỉ còn một lỗi kỹ thuật trên read, nó chỉ có thể là biến thể nếu như một read khác cũng có một bazơ tương tự tại cùng vị trí. Ngoài ra, các trình tự trùng lặp cũng có thể là kết quả của sự xác định không chính xác một cụm khuếch đại thành nhiều cụm bởi cảm biến quang học của thiết bị giải trình tự. Những trùng lặp này còn này được gọi là trùng lặp quang học.

Các trùng lặp cũng ảnh hưởng đến thành phần GC mà chúng ta đã xem xét ở phần trước. Sự thay đổi của thành phần GC có thể được gây ra bởi một trình tự nào đó bị trùng lặp nhiều hơn những trình tự khác. Trong thực hành, ta chưa cần loại bỏ các trùng lặp trong quá trình tiền xử lý dữ liệu. Các trùng lặp được phát hiện sau khi dóng hàng sẽ được quyết định loại bỏ dựa vào các tọa độ (mapping coordinate) của chúng. Việc khử trùng lặp thông thường sẽ được thực hiện bởi công cụ Picard, trong đó các lỗi trùng lặp do quá trình chuẩn bị thư viện NGS bằng PCR và các lỗi trùng lặp quang học sẽ được loại bỏ. Tuy nhiên, đối với công nghệ giải trình tự amplicon như Ion Torrent PGM, các đoạn DNA được khuếch đại rất lớn, ta không thể dùng công cụ Picard mà phải dùng phần mềm của chính nhà phát triển Ion Torrent là Torrent Suit Software với plugin FilterDuplicates [3]. Tuy nhiên, Torrent Suit Software chỉ có bản thương mại, do đó ta sẽ không thực hiện khử trùng lặp và chấp nhận các kết quả dương tính giả. Tuy nhiên, sau khi gọi ra các biến thể ta cần hiệu chỉnh lại chúng để hạn chế đến mức tối đa dương tính giả.

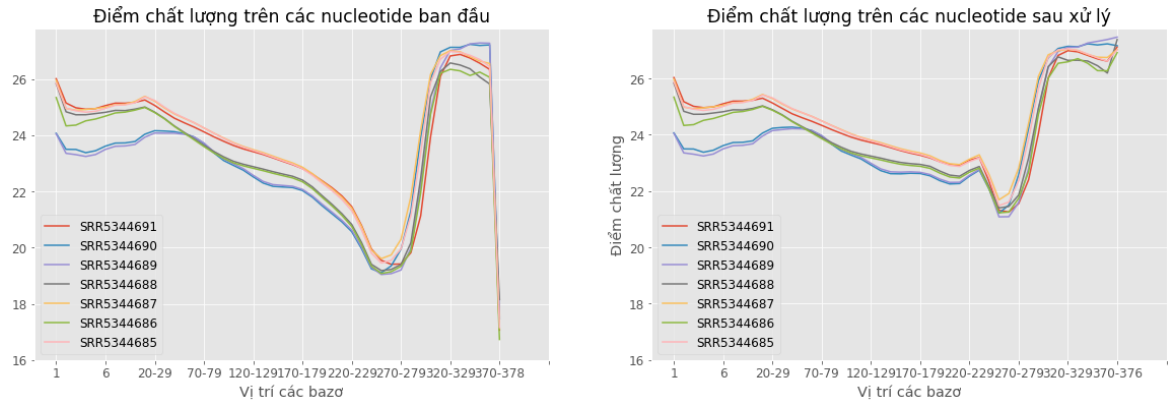
3.2.2 Loại bỏ các bazơ có điểm chất lượng kém

Các bazơ có chất lượng kém tại đuôi của các trình tự được cắt bỏ bằng công cụ trimomatics. Công cụ này có khả năng xác định nhiễm bẩn adapter hiệu quả cùng với các chức năng lọc, cắt và loại bỏ dữ liệu chất lượng kém. Hơn nữa, công cụ còn hỗ trợ phân tách dữ liệu loại paired-end reads, không can thiệp vào phân tích dữ liệu hạ nguồn [BLU14]. Ở đây, ta sử dụng trimomatics với ngưỡng điểm chất lượng là 24 để giữ lại các bazơ ở phần đuôi của các trình tự (Xem hình 3.4). Dữ liệu sau xử lý đã tốt hơn nhiều so với dữ liệu ban đầu, tuy nhiên nó có thể vẫn chưa đủ tốt để hạn chế đến mức tối đa tình trạng dương tính giả.

[3]https://www.thermofisher.com/vn/en/home/life-science/sequencing/next-genration-sequencing/ion-torrent-next-genration-sequencing-workflow/ion-torrent-next-genration-sequencing-data-analysis-workflow/ion-torrent-suite-software-plugins.html



Hình 3.3: Biểu đồ phần trăm trình tự còn lại sau khi khử trùng lặp mẫu SRR5344691. Trục hoành là số lượng trùng lặp, trục tung là phần trăm các trình tự. Đường xanh biểu diễn tổng tất cả trình tự, đường đỏ biểu diễn các trình tự sau khi được khử trùng lặp. Trên hình, phần trăm các trình tự còn lại sau khi khử trùng lặp là 85.68%. Đối với dữ liệu ban đầu, có 79.79% trình tự duy nhất, 17.11% trình tự có số trùng lặp lớn hơn 10. Đối với dữ liệu sau khi khử trùng lặp, có 93.12% trình tự duy nhất, trong khi có rất ít các trình tự trùng lặp từ 5 trở lên.



Hình 3.4: Điểm chất lượng trên các bazơ trước và sau khi cắt bỏ các bazơ ở đuôi các reads. Với ngưỡng cắt bỏ là 24, điểm chất lượng của các bazơ tại các vị trí từ 209 đến 259 đã được nâng cao. Ngoài ra, phần đuôi của đồ thị không còn đoạn giảm chất lượng với độ dốc lớn.

3.2.3 Dóng hàng trình tự

Dữ liệu hiện nay có thể nhận được từ nhiều nền tảng của các nhà cung cấp khác nhau như Illumina, Roche 454 GS, FLX Titanium, Ion Torrent PGM, PacBio [Mis+14]. Do đó công cụ BWA được chia ra làm 3 loại để áp dụng cho các dữ liệu khác nhau [4]:

- BWA-MEM: sử dụng các trình tự Illumina 70bp hoặc dài hơn, Roche 454, Ion Torrent và Sanger reads.
- BWA-backtrack: sử dụng cho các trình tự ngắn.
- BWA-SW: sử dụng cho dữ liệu có tần suất các khoảng trống (gaps) dóng hàng cao.

Dữ liệu đang được sử dụng là loại single-banded amplicon của Ion Torrent PGM, do đó chúng ta sử dụng công cụ BWA-MEM để thực hiện dóng hàng trình tự [LD09; LD10; Li13]. Chú ý rằng bước dóng hàng trình tự vẫn nằm trong giai đoạn tiền xử lý dữ liệu. Sau bước này thường là bước khử trùng lặp và dóng hàng lại các Indels. Tuy nhiên, ta bỏ qua bước khử trùng lặp vì lý do đã nêu ở phần 3.3, còn bước dóng hàng lại các Indels hiện nay đã không còn khả dụng với GATK 4.x.

Để phục vụ cho công việc phân tích hạ nguồn, thông tin nhóm cho các reads được thêm vào, thông tin này cũng là yêu cầu bắt buộc khi thực hiện dóng hàng lại Indels. Với mate pairs, dữ liệu sau khi dóng hàng thường được xắp xếp theo tọa độ để công cụ xử lý không đánh dấu các mate không khớp với các trình tự bổ xung là trùng lặp. Tuy nhiên, chúng ta vẫn xắp xếp như vậy với dữ liệu single-banded amplicon bằng công cụ SortSam của Picard để đảm bảo sự chính xác cho các phân tích về sau. Sau khi dóng hàng trình tự, dữ liệu được hiệu chỉnh lại các bazơ bằng công cụ BaseRecalibrator của GATK [5] dựa trên cơ sở dữ liệu về SNPs 00-All.vcf.gz (dung lượng 15GB) [6]. Cuối cùng, các trình tự có chất lượng dóng hàng mapQ < 60 được đánh dấu bằng công cụ Samtools [7]. Các tham số được lựa chọn cùng với các phần mềm trong giai đoạn tiền xử lý dữ liệu được liệt kê trong bảng 3.2.

3.3 Xác định biến thể

Bước đầu tiên của giai đoạn này là gọi ra các biến thể từ dữ liệu đã được dóng hàng. Để xác định biến thể mầm ta sử dụng công cụ Haplotype Caller phiên bản

[4] http://bio-bwa.sourceforge.net/
[5] https://gatk.broadinstitute.org/hc/en-us/articles/360036898312-BaseRecalibrator
[6] https://ftp.ncbi.nih.gov/snp/organisms/human_9606/VCF/GATK/
[7] http://www.htslib.org/download/

Bảng 3.2: Lựa chọn các tham số trong giai đoạn tiền xử lý dữ liệu. Các tham số trong BWA-MEM: A - thưởng điểm khớp, B - phạt không khớp, O - phạt khoảng mở, E - phạt khoảng mở kéo dài. Các công cụ Base Recalibrator, Apply BQSR, và Samtools thuộc về GATK.

Công Cụ	**Tham Số**	**Giá Trị**
Trimomatic	TRAILING	24
BWA-MEM	reference	NCBI GRCh38.p13
	A	1
	B	4
	O	6
	E	1
GATK		
BaseRecalibrator	known_sites	00-All.vcf.gz
ApplyBQSR		
Samtools	MAPQ	60

4.4.1 [8]. Để có thể kết hợp kết quả của 7 mẫu vào một tệp, chế độ GVCF được sử dụng để gọi biến thể cho từng mẫu. Haplotype Caller hoạt động theo trình tự gồm 4 bước:

- Xác định vùng hoạt động: chương trình xác định vùng nào của bộ gen mà nó cần hoạt động dựa trên các bằng chứng về biến thể.
- Xác định các Haplotype bằng cách ráp các vùng hoạt động: đối với mỗi vùng hoạt động, chương trình xây dựng một đồ thị De Bruijn để ráp lại vùng hoạt động và xác định đâu là các dạng Haplotype có thể có trong dữ liệu. Sau đó, chương trình sẽ dóng hàng lại từng Haplotype với Haplotype tham chiếu bằng cách sử dụng thuật toán Smith-Waterman.
- Xác định khả năng (likelihood) xảy ra các dạng Haplotype với dữ liệu trình tự: đối với mỗi vùng hoạt động, chương trình thực hiện căn chỉnh theo từng cặp của mỗi trình tự với mỗi loại Haplotype bằng cách sử dụng thuật toán PairHMM [RBA18].
- Chỉ định kiểu gen: đối với mỗi vùng có khả năng chứa biến thể, chương trình áp dụng quy tắc Bayes, sử dụng khả năng của các allele để tính toán khả năng xảy ra của mỗi kiểu gen trên mỗi mẫu dựa trên dữ liệu trình tự được quan sát cho mẫu đó. Sau đó, kiểu gen có khả năng xảy ra cao nhất sẽ được đưa vào mẫu.

[8]https://gatk.broadinstitute.org/hc/en-us/

Đây là quá trình chiếm nhiều thời gian nhất so với các bước còn lại. Với mẫu SRR5344691 có kích thước 5.2GB, thời gian gọi biến thể là gần 53 giờ khi thực thi song song trên 15 CPUs. Ta sử dụng chế độ GVCF để gọi ra từng tập biến thể của từng mẫu để sau đó chúng được tổng hợp lại. Phương pháp tổng hợp được lựa chọn là lấy giao (INTERSECTION) của các tập biến thể con. Như vậy, xuất phát từ việc dóng hàng trình tự ta đã có một bộ dữ liệu thô các biến thể gồm SNPs và Indels. Dữ liệu này được đưa vào giai đoạn cuối cùng để tìm kiếm các đột biến trong gen.

3.4 Chú thích chức năng

Trước khi xác định mức độ ảnh hưởng của các biến thể đến chức năng của protein ta cần lọc các biến thể có chất lượng thấp. Phương pháp hiệu chỉnh dữ liệu dựa trên 6 tiêu chí: QualByDepth (QD), FisherStrand (FS), StrandOddsRatio (SOR), RMSMappingQuality (MQ), MappingQualityRankSumTest (MQRankSum), ReadPosRankSumTest (ReadPosRankSum). Với tập biến thể được tạo ra từ bước 3.3, tham số MQ và MQRankSum gần như không đổi qua tất cả các lần gọi (MQ = 60.0, MQRankSum = 0.0), do đó ta loại bỏ hai tham số này ra khỏi quá trình điều chỉnh tham số. Dữ liệu thô được hiệu chỉnh bởi công cụ hiệu chỉnh điểm chất lượng biến thể (Variant Quality Score Recalibration, viết tắt là VQSR) áp dụng máy học với các cơ sở dữ liệu lớn [9]. Công cụ này đánh dấu các điểm dữ liệu không đạt dựa trên 4 loại chỉ số chất lượng: QD, FS, SOR, ReadPosRankSum. Ngoài ra, dữ liệu được sử dụng cho việc huấn luyện mô hình là các cơ sở dữ liệu lớn cho Indels (như mills, axiomPoly, dbsnp) và SNPs (như hapmap, omni, 1000G, dbsnp) (Xem bảng 3.3). Sau khi các biến thể có điểm không đạt được đánh dấu, ta tiến hành chú thích chức năng bằng công cụ Funcotator sử dụng dữ liệu cho germline [10].

Trong nghiên cứu gốc bao gồm 7 bộ dữ liệu đang xem xét, Giacopuzzi và các cộng sự xác định các đột biến mới và cực kỳ hiếm trong đường chuyển hóa GABA/Glutamatergic. Do đó họ lựa chọn các biến thể với tần số Allele (AF) rất nhỏ chẳng hạn 0.00000828 với biến thể gen FMN1, 0.0000664 với biến thể ANO2. Hai biến thể gen khác được tìm thấy được cho là mới là MEGF8 và GAD1. Trong luận văn này, **phạm vi tìm kiếm biến thể** được mở rộng. **Mục đích của chúng ta là xác định vùng dữ liệu có khả năng chứa nhiều nhất các biến thể đồng thời với việc hạn chế dương tính giả**. Kết quả cuối cùng được đối chiếu với các nghiên cứu trước đó (Xem phần 3.1). Ngoài ra, chúng còn được so sánh với dữ liệu về các protein đã được thí nghiệm có liên quan đến bệnh tâm thần phân liệt được tải về từ UniProtKB [11] với cú pháp

[9] https://gatk.broadinstitute.org/hc/en-us/articles/360035531612-Variant-Quality-Score-Recalibration-VQSR-

[10] https://gatk.broadinstitute.org/hc/en-us/articles/360037224432-Funcotator

[11] https://www.uniprot.org/

tìm kiếm: annotation:(type:disease schizophrenia) AND organism:"Homo sapiens (Human) [9606]".

Bảng 3.3: Các tham số và dữ liệu sử dụng trong quá trình hiệu chỉnh biến thể. Công cụ Variant Recalibrator thực thi với hai chế độ INDEL và SNP, các tập dữ liệu huấn luyện trong hai chế độ này khác nhau được liệt kê ở cột dữ liệu.

Công Cụ	**Dữ liệu**
VariantRecalibrator	ExcessHet > 54.69
Mode: INDEL	max-gaussians: 4 Mills_and_1000G_gold_standard.indels.hg38.vcf.gz Axiom_Exome_Plus.genotypes.all_populations.poly.hg38.vcf.gz 00-All.vcf.gz
Mode: SNP	max-gaussians: 6 hapmap_3.3.hg38.vcf.gz 1000G_omni2.5.hg38.vcf.gz 1000G_phase1.snps.high_confidence.hg38.vcf.gz 00-All.vcf.gz
ApplyVQSR	truth-sensitivity: 99.7

3.5 Kết quả

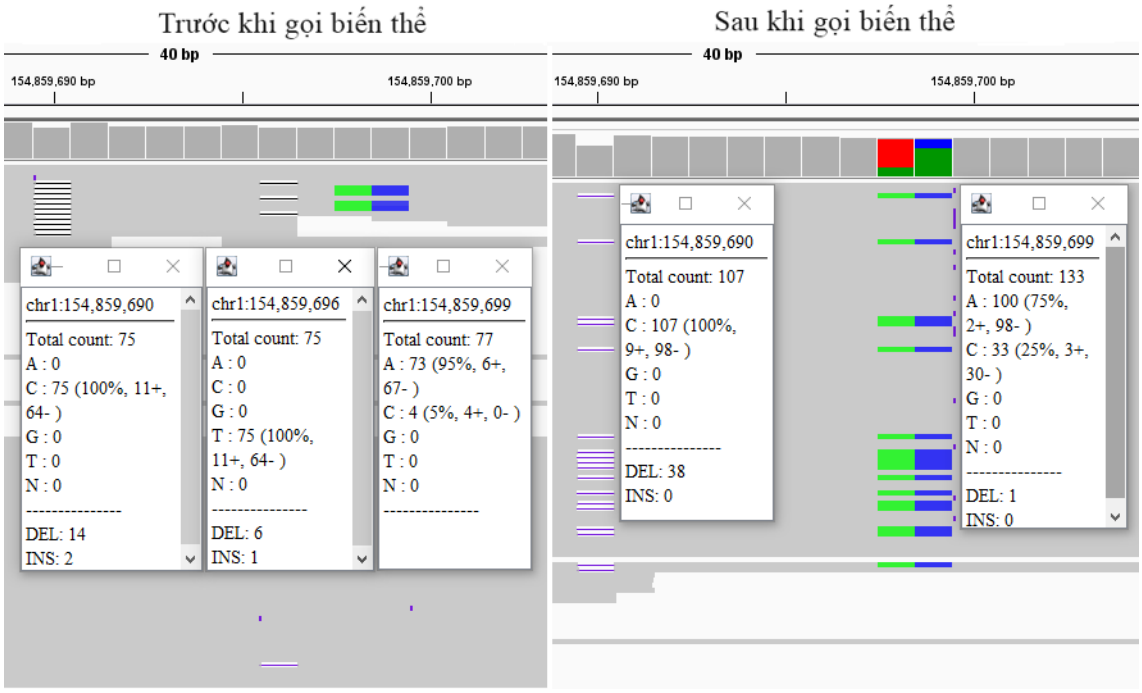
Dóng hàng trình tự và các bước xử lý sau đó trong giai đoạn tiền xử lý dữ liệu cho ta các bộ dữ liệu đã được khớp với bộ gen tham chiếu tại các tọa độ xác định (Xem bảng 3.4). Tổng trình tự của các bộ dữ liệu sau khi loại bỏ các trình tự có chất lượng khớp thấp được thể hiện ở cột Tổng trình tự. Cột Số bazơ khớp chứa số lượng các bazơ khớp với các bazơ của bộ gen tham chiếu. Cột MM (mismatches) là tổng các không khớp của mỗi mẫu, các không khớp này có thể là SNPs hoặc Indels.

Các trình tự sau khi dóng hàng được đưa vào công cụ Haplotype Caller cho việc gọi biến thể, các Haplotype trong các vùng hoạt động sẽ được dóng hàng với các Haplotype tham chiếu bằng thuật toán Smith-Waterman (Xem phần 3.3). Nhờ việc dóng hàng lại các Haplotype mà việc gọi ra các Indels cũng như các SNPs trở nên chính xác hơn. Một trường hợp sau khi dóng hàng lại các Haplotype trên nhiễm sắc thể số 1 của mẫu SRR5344691 được thể hiện trong hình 3.5. Xuất hiện các vị trí mà Indels được dồn lại (ở tọa độ 15485969, từ 14 xóa và 2 chèn trở thành 38 xóa), cũng như các vị trí mà chúng biến mất (ở tọa độ 154859696). Tại các SNPs, tần số của các nucleotide thay đổi cũng cao hơn (ở tọa độ 154859699, từ 5% C tới 25% C).

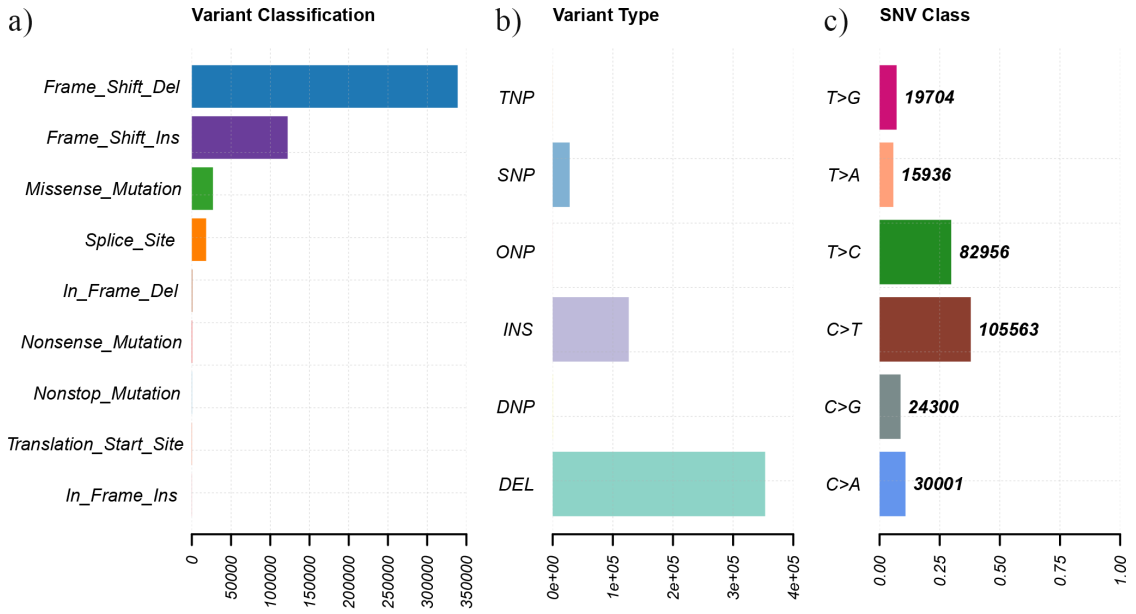
Bảng 3.4: Thống kê các bộ trình tự sau giai đoạn tiền xử lý. Ký hiệu viết tắt của các cột trong bảng: MM - tổng các không khớp, AL - chiều dài trung bình của trình tự, ML - chiều dài lớn nhất trong các trình tự, AQ - chất lượng trung bình.

Mẫu	Tổng trình tự	Số bazơ khớp	MM	AL	ML	AQ
SRR5344685	32138109	5089292737	51035662	159	376	31.5
SRR5344686	37578694	5678894937	62207174	151	375	31.4
SRR5344687	37154673	5826116380	57476347	157	375	31.6
SRR5344688	28554740	4378531950	48571451	154	375	31.4
SRR5344689	23216250	3679398807	50099158	159	375	31.9
SRR5344690	19270365	2960931705	38928150	154	375	31.5
SRR5344691	45833172	7946530926	80582216	174	376	31.9

Sau quá trình lọc biến thể ta thu được tập các biến thể sử dụng cho quá trình đánh giá ảnh mức độ ảnh hưởng đến kiểu hình (Xem hình 3.6). Nhận thấy số lượng SNPs khá nhỏ so với Indels, số lượng biến thể C thành T là lớn nhất với 105563 vị trí. Mặt khác, tỷ lệ Ts/Tv (transition-to-transversion) = 67.7/32.3 = 2.10 phù hợp với tiêu chí đánh giá cho dữ liệu WGS của người. Điều này cũng chứng tỏ tỷ lệ dương tính giả thấp và dữ liệu không bị lệch (bias).



Hình 3.5: Dóng hàng trước và sau khi gọi biến thể của mẫu SRR5344691 tại các vị trí trên nhiễm sắc thể số 1. Ở tọa độ 15485969, từ 14 xóa và 2 chèn trở thành 38 xóa. Tại tọa độ 154859696 các xóa và chèn được loại bỏ. Tại tọa độ 154859699, tần số nucleotide thay đổi cao hơn từ 5% C tới 25% C.

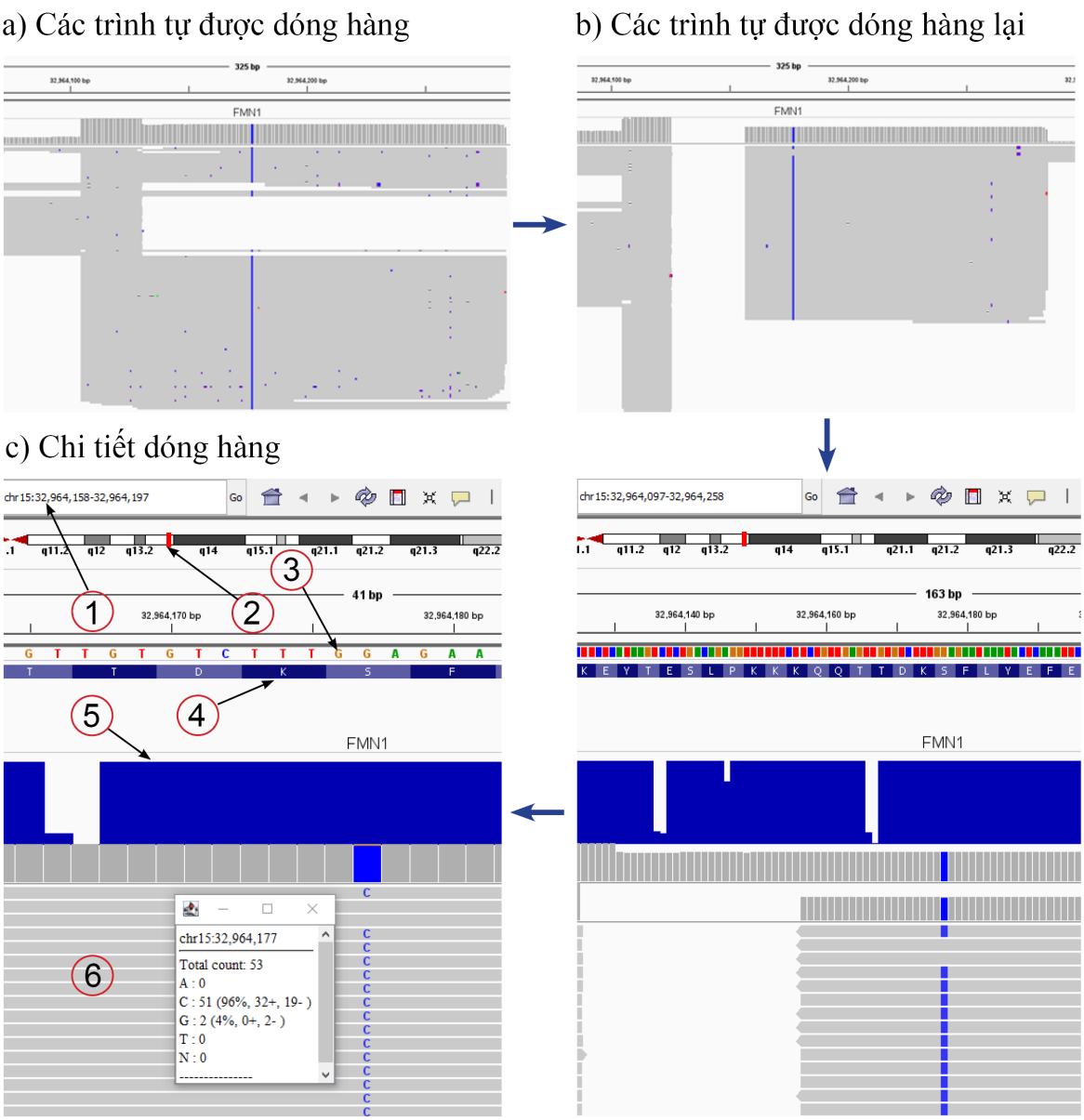


Hình 3.6: Các biểu đồ thống kê các biến thể sau quá trình lọc biến thể. a) Phân loại biến thể b) Loại biến thể: số lượng SNPs nhỏ nhất so với thêm và xóa c) Các biến thể nucleotides đơn: số lượng biến thể C thành T lớn nhất tại 105536 vị trí.

Tập biến thể cuối cùng được chú thích chức năng, ta xác định được 7362 biến thể trên tổng số 5059 gen có thể làm thay đổi chức năng của protein. Một số gen đột biến được tìm thấy trùng hợp với kết quả của các nghiên cứu trước đó.

Trong số các biến thể gen đã được tìm thấy, ta sẽ hiển thị đột biến FMN1 trên mẫu SRR5344691 để xem xét một cách trực quan kết quả dóng hàng tìm biến thể (Xem hình 3.7).

- 1: Biến thể nằm trên nhiễm sắc thể thứ 15.
- 2: Đột biến được đánh dấu gạch đỏ thuộc nhánh dài q13.2 của nhiễm sắc thể.
- 3: Tại vị trí đột biến, mã hóa của bazơ trên bộ gen tham chiếu là G.
- 4: Trình tự amino acid, bộ ba GGA mã hóa cho amino acid Serine(S).
- 5: Vùng bảo tồn của 99 loài động vật có xương sống và con người. Trong đó chiều cao của khối thể hiện mức độ bảo tồn. Với đột biến FMN1, vị trí của biến thể nằm trong vùng bảo tồn cao.
- 6: Các trình tự sau khi được dóng hàng đều có vị trí 32964177 là nucleotide C, bazơ này khác với G của bộ gen tham chiếu. Tại đây bộ ba GGA mã hóa amino acid Serine (S) chuyển thành GCA mã hóa cho Cysteine (C) làm thay đổi chức năng của protein Formin-1.



Hình 3.7: a) Các trình tự ngay sau khi được dóng hàng bằng thuật toán BWA-MEM b) Các trình tự được dóng hàng lại nhờ công cụ Haplotype Caller c) Chi tiết dóng hàng trình tự: Dóng hàng phát hiện biến thể G thành C (G-C) tại vị trí 32964177 nằm trên nhiễm sắc thể thứ 15 (chr15) thuộc gen FMN1. Biến thể G-C làm amino acid Serine (S) chuyển thành Cysteine (C) gây ra sự thay đổi chức năng của Protein Formin-1. Biến thể gen này giống kết quả nghiên cứu của Edoardo Giacopuzzi et al. (2017) thực hiện bằng phương pháp WES.

Trong các bảng thống kê đột biến gen sau đây, cột Chr là tên nhiễm sắc thể, Pos là vị trí của biến thể, cột Allele chứa sự thay đổi các Nucleotide, cột gen hiển thị tên gen, cột Amino thống kê các amino acid thay đổi, cột Score chứa điểm được tính bởi công cụ SIFT4G (từ 0 đến 0.05 với các biến thể có ảnh hưởng

làm thay đổi protein), cột dbSNP cho biết các đột biến gen đã có trong cơ sở dữ liệu hay là những đột biến mới. Kết quả các gen trùng hợp được thể hiện bằng tỷ lệ tổng số gen đột biến giống nhau trên số lượng gen đột biến của các nghiên cứu có sẵn. Đối với các nghiên cứu cùng phương pháp tỷ lệ gen đột biến được tìm thấy là 3/4 trong công bố của Giacopuzzi et al. (2017) [Gia+17] và 4/7 trong công bố của Nishioka et al. (2018) [Nis+18] (Xem bảng 3.5). Đối với nghiên cứu biểu hiện gen của Tom Walsh et al. (2008) [Wal+08] ta có 14 trên 34 gen (Xem bảng 3.6). Mặt khác, ở nghiên cứu về GWAS năm 2019, Laura m. Huckins và các cộng sự [Huc+19] đã công bố 70 gen đột biến trong số 413 gen đột biến được tìm thấy, trong đó có 17 gen có trong kết quả của luận văn (Xem bảng 3.7). Toàn bộ các gen tìm được còn được đối chiếu với dữ liệu trên UniProtKB [12] bao gồm các gen đột biến đã được thí nghiệm, ta có 11/57 gen giống nhau (Xem bảng 3.8). Ngoài ra, ta còn có 9/16 gen xuất hiện trong nghiên cứu bằng phương pháp DNMs của Daniel P.Howrigan et al. (2020) [How+20] (Xem bảng 3.9). Số lượng đột biến cũng có thể được thu gọn nếu chúng ta giới hạn các kết quả bằng các tiêu chí khác như tần số allele, đường chuyển hóa, hay các biến thể mới chưa có trong cơ sở dữ liệu.

Bảng 3.5: Những gen giống với các nghiên cứu cùng phương pháp giải trình tự exome. Tìm được 3 trong số 4 gen được Giacopuzzi công bố năm 2017, 4 trong số 7 gen thuộc nghiên cứu của Nishioka năm 2018.

Chr	**Pos**	**Allele**	**Gen**	**Amino**	**Score**	**dbSNP**
Giacopuzzi et al. (2017) - WES (3/4) [Gia+17]						
12	5854114	G-A	ANO2	R-W	0	rs767675843
15	32964177	G-C	FMN1	S-C	0.002	rs762291357
15	33067169	C-T	FMN1	G-E	0.039	rs11072170
19	42326316	G-C	MEGF8	A-P	0.049	novel (l)
19	42336114	G-A	MEGF8	A-T	0	novel (l)
19	42356089	G-A	MEGF8	G-S	0	novel (l)
19	42368605	G-T	MEGF8	G-W	0.032	novel (l)
19	42376182	T-C	MEGF8	F-L	0	novel (l)
Nishioka et al. (2018) - WES (4/7) [Nis+18]						
7	1.06E+08	G-A	CDHR3	V-M	0	rs35008315 (l)
7	1.06E+08	G-C	CDHR3	Q-H	0.025	rs34426483 (l)
11	73235159	C-T	P2RY2	R-C	0.004	rs1626154
12	21887888	C-T	ABCC9	D-N	0.008	rs757681761
12	78175348	G-A	NAV3	S-N	0.001	novel
12	78200517	A-G	NAV3	R-G	0.002	novel

[12]https://www.uniprot.org/

Bảng 3.6: Những gen giống với nghiên cứu biểu hiện gen. Tìm được 4 trong 12 gen thuộc nghiên cứu của Harrison năm 2005, 14 trong 34 gen được Tomas Walsh công bố năm 2008.

Chr	**Pos**	**Allele**	**Gen**	**Amino**	**Score**	**dbSNP**
P J Harrison & D R Weinberger (2005) -gen Expression (4/12) [HW05]						
8	32595840	G-A	NRG1	R-Q	0.019	rs3924999
8	32754452	G-T	NRG1	V-L	0	rs74942016
19	35800075	C-T	PRODH2	R-Q	0.006	rs3761097
22	19963748	G-A	COMT	V-M	0.029	rs4680
1	2.32E+08	C-T	DISC1	S-L	0.015	rs2492367 (l)
1	2.32E+08	A-T	DISC1	R-S	0	rs821616 (l)
8	31640670	C-T	NRG1	P-L	0	novel (l)
8	32756465	T-C	NRG1	M-T	0.037	rs10503929 (l)
Tomas Walsh et al. (2008) - gen Expression (14/34) [Wal+08]						
1	2.34E+08	C-T	TARBP1	V-M	0.028	rs111283864
1	2.34E+08	A-G	TARBP1	C-R	0.004	novel
1	2.34E+08	C-T	TARBP1	C-Y	0.004	novel
2	48581013	G-C	STON1-GTF2A1L	R-T	0.01	rs940389
3	53187345	A-T	PRKCD	Y-F	0.035	novel
3	78635874	C-T	ROBO1	S-N	0.012	rs35456279
7	78135152	G-T	MAGI2	A-E	0.003	novel
7	1.01E+08	G-A	SLC12A9	V-M	0.009	rs200508321
9	2186102	G-A	SMARCA2	E-K	0.004	novel
9	13188954	A-T	MPDZ	S-T	0.003	rs200475640
12	25204101	T-C	LYRM5	L-P	0.001	novel
18	6965302	C-T	LAMA1	R-Q	0	rs140792199
18	6986228	G-A	LAMA1	A-V	0.045	rs12607841
18	7008592	T-C	LAMA1	M-V	0.005	rs662471
18	7038946	T-C	LAMA1	N-S	0.019	rs200035723
18	7049122	T-C	LAMA1	S-G	0.007	novel
18	7888307	C-T	PTPRM	S-F	0	novel
1	99709114	C-T	FRRS1	M-I	0	rs12145706 (l)
5	36184002	A-T	SKP2	L-F	0	rs776163190 (l)
7	77922810	G-A	PHTF2	R-H	0	rs848486 (l)

Bảng 3.7: Những gen giống với các nghiên cứu GWAS. Tìm được 3 trong 10 gen được công bố bởi Peilin Jia năm 2010, 17 trong 70 gen được Laura M. Huckins công bố năm 2019.

Chr	**Pos**	**Allele**	**Gen**	**Amino**	**Score**	**dbSNP**
Peilin Jia et al. (2010) - GWAS (3/10) [Jia+10]						
2	2.11E+08	A-G	CPS1	T-A	0.036	rs1047883
6	24503369	C-T	ALDH5A1	P-L	0.037	rs3765310
6	53505349	C-T	GCLC	V-I	0.049	rs41271287 (l)
Laura M. Huckins et al. (2019) - GWAS (17/70) [Huc+19]						
1	8356186	C-T	RERE	R-H	0	novel
1	8358264	G-A	RERE	P-L	0	novel
1	8361126	G-A	RERE	P-L	0.007	rs201922249
1	8361356	G-C	RERE	S-R	0.024	novel
1	29260616	C-A	PTPRU	P-Q	0.001	novel
1	29260618	A-C	PTPRU	T-P	0.035	novel
1	29304816	G-T	PTPRU	V-F	0.047	novel
3	1.36E+08	C-A	PCCB	S-R	0.016	novel
6	26501339	T-C	BTN1A1	L-P	0.041	novel
9	34726674	G-A	FAM205A	T-M	0.026	rs1854574
11	17129366	G-A	PIK3C2A	S-F	0.006	novel
11	17131997	C-T	PIK3C2A	G-E	0.02	novel
11	1.31E+08	G-A	SNX19	P-L	0.01	rs62621284
12	1.23E+08	G-A	PITPNM2	R-C	0.028	rs146027647
14	61720459	A-T	HIF1A	Y-F	0.042	novel
16	29996374	C-T	INO80E	L-F	0.006	novel
19	48728969	G-A	RASIP1	R-C	0.001	rs2287922
20	38750078	G-A	ACTR5	M-I	0.017	rs34862565
20	63542208	C-G	SRMS	V-L	0.018	rs310657
1	8656146	G-A	RERE	A-V	0.006	novel (l)
3	1.36E+08	G-T	PCCB	A-S	0.017	novel (l)
7	1.06E+08	A-T	RINT1	S-C	0	rs11556986 (l)
15	78098072	T-C	SH2D7	M-T	0	rs2289524 (l)

Bảng 3.8: Tìm được 10 trong số 57 gen đã được thí nghiệm trên UniProtKB tính đến tháng 1 năm 2021.

Chr	Pos	Allele	Gen	Amino	Score	dbSNP
UniProtKB (10/57) [13]						
1	11796321	G-A	MTHFR	A-V	0.048	rs1801133
1	2.03E+08	T-C	CHI3L1	R-G	0.001	rs880633
3	41911360	T-C	ULK4	S-G	0.014	rs35263917
9	2110391	T-C	SMARCA2	F-L	0.003	novel
9	2186102	G-A	SMARCA2	E-K	0.004	novel
16	29911857	T-C	KCTD13	K-R	0.01	novel
19	35800075	C-T	PRODH2	R-Q	0.006	rs3761097
19	35812762	G-C	PRODH2	P-R	0.032	rs3848666
20	35654425	T-G	RBM12	N-H	0.011	novel
20	35655162	C-A	RBM12	R-M	0	novel
22	19963748	G-A	COMT	V-M	0.029	rs4680
1	2.32E+08	A-T	DISC1	R-S	0	rs821616
13	1.05E+08	G-A	DAOA	R-K	0	rs2391191

Bảng 3.9: Tìm được 1 gen giống với nghiên cứu liên kết của Brzustovic năm 2000, 9 trong 16 gen thuộc nghiên cứu DNMs của Daniel Howrigan năm 2020.

Chr	Pos	Allele	Gen	Amino	Score	dbSNP
Brzustovic et al. (2000) - Linkage Analysis (1/1) [Brz+00]						
1	1.55E+08	A-C	KCNN3	I-R	0.008	novel (l)
Daniel P. Howrigan et al. (2020) - DNMs (9/16) [How+20]						
2	1.79E+08	G-A	TTN	P-L	0.01	rs151253841
5	14488234	T-C	TRIO	S-P	0.012	novel
10	1.28E+08	G-A	MKI67	R-W	0.002	rs34916904
10	1.28E+08	C-G	MKI67	E-D	0	rs11016076
14	21385863	G-A	CHD8	P-L	0.038	novel
14	78709265	C-T	NRXN3	R-W	0.003	rs748055286
16	30959145	T-C	SETD1A	S-P	0.007	novel
16	30959146	C-A	SETD1A	S-Y	0.002	novel
1	41510459	T-G	HIVEP3	N-H	0.035	novel (l)
4	1.22E+08	C-T	KIAA1109	T-I	0	novel (l)
4	1.22E+08	G-A	KIAA1109	D-N	0	novel (l)
14	21429308	G-A	CHD8	L-F	0	rs192989929 (I)

KẾT LUẬN

Trong luận văn này, sau phần tổng hợp và trình bày một số kiến thức cơ sở cần thiết liên quan tới tin sinh học, thuật toán dóng hàng dựa trên chuyển dạng Burrows-Wheeler được tìm hiểu kĩ và trình bày một cách tường minh hơn so với các bài báo gốc. Ngôn ngữ Go với kỹ thuật song song và đồng thời được chọn để thử nghiệm thuật toán, phát huy tối đa những điểm mạnh khi làm việc với dữ liệu lớn. Kết quả nhận được trong phần thử nghiệm thuật toán giống với kết quả nhận được từ công cụ BWA-MEM với cùng dữ liệu đầu vào.

Thuật toán dóng hàng trình tự dựa trên chuyển dạng Burrows-Wheeler và thuật toán Smith-Waterman được áp dụng trong bài toán dự đoán biến thể gen đã ráp lại các trình tự lưu trữ một cách chính xác. Từ dữ liệu dóng hàng ta thu được tập các SNPs và Indels với tỉ lệ Ts/Tv phù hợp cho thấy độ tin cậy của chương trình. Đồng thời các dữ liệu này cũng có thể sử dụng để thực hiện phân tích tiếp theo cho dự đoán các biến thể. Theo phạm vi tìm kiếm rộng được xác định từ đầu, một số lượng lớn các biến thể trên các gen đã được tìm thấy, trong đó có nhiều đột biến gen giống với các công bố trước đây. Dựa trên các tiêu chí khác nhau mà chúng ta có thể đưa ra các tập biến thể nhỏ hơn cho các công việc phân tích và kiểm chứng bằng thực nghiệm.

Về mặt phương pháp, có thể giảm hơn nữa các dương tính giả nếu thực hiện thêm bước khử trùng lặp cho dữ liệu Ion Torrent. Mặt khác, phạm vi biến thể có thể được thu hẹp dựa vào các tiêu chí xác định trước. Ngoài ra, giải trình tự gen để xác định biến thể có thể được kết hợp với các phương khác như: phân tích liên kết, biểu hiện gen, GWAS... để cho ra kết quả chính xác hơn. Tuy vậy, các công cụ, thuật toán chỉ giúp phân tích, khoanh vùng một tập hợp có khả năng cao các biến thể gen. Để đánh giá kết quả một cách chắc chắn ta vẫn cần thực hiện các thí nghiệm sinh hóa.

Chúng tôi cho rằng các phương pháp được nghiên cứu trong luận văn có tính áp dụng thực tiễn cao. Qua những kĩ thuật này, kết quả thu được có thể làm cơ sở cho việc dự đoán các biến thể gen di truyền và không di truyền ở người, động vật, thực vật cũng như tìm kiếm các gen tương đồng trên người và động vật. Những dự đoán này rất có ý nghĩa trong hỗ trợ cho các nghiên cứu thực

nghiệm, giúp tăng tính khả thi cũng như hiệu quả của các thực nghiệm. Xa hơn nữa, việc phân tích dữ liệu giải trình tự cũng giúp tìm ra nguyên nhân gây bệnh, dự đoán khả năng mắc bệnh do di truyền trong một phả hệ, hoặc áp dụng trong hỗ trợ điều trị bệnh sử dụng trình tự nhắm mục tiêu. Từ đây, những dịch vụ xác định gen tiềm năng, tầm soát ung thư và các bệnh di truyền có thể được triển khai trong các hệ thống bệnh viện và y tế dự phòng.

Tài liệu tham khảo

[ADL08] Altshuler, D., Daly, M. J., and Lander, E. S. "Genetic Mapping in Human Disease". In: *Science* vol. 322, no. 5903 (Nov. 7, 2008), pp. 881–888. pmid: 18988837.

[Alt+90] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. "Basic Local Alignment Search Tool". In: *Journal of Molecular Biology* vol. 215, no. 3 (Oct. 5, 1990), pp. 403–410.

[And90] Anderson, W. F. "September 14, 1990: The Beginning". In: *Human Gene Therapy* vol. 1, no. 4 (Dec. 1, 1990), pp. 371–372.

[Bel66] Bellman, R. "Dynamic Programming". In: *Science* vol. 153, no. 3731 (July 1, 1966), pp. 34–37. pmid: 17730601.

[BLU14] Bolger, A. M., Lohse, M., and Usadel, B. "Trimmomatic: A Flexible Trimmer for Illumina Sequence Data". In: *Bioinformatics* vol. 30, no. 15 (Aug. 1, 2014), pp. 2114–2120.

[Brz+00] Brzustowicz, L. M., Hodgkinson, K. A., Chow, E. W. C., Honer, W. G., and Bassett, A. S. "Location of a Major Susceptibility Locus for Familial Schizophrenia on Chromosome 1q21-Q22". In: *Science* vol. 288, no. 5466 (Apr. 28, 2000), pp. 678–682. pmid: 10784452.

[BW94] Burrows, M. and Wheeler, D. J. *A Block-Sorting Lossless Data Compression Algorithm*. 1994.

[CF] Ciochon, R. L. and Fleagle, J. G. *The Human Evolution Source Book*.

[Com15] Compeau, P. *BIOINFORMATICS ALGORITHMS,VOL.I*. 2nd Edition. La Jolla, CA: Active Learning Publishers, Jan. 1, 2015. 384 pp.

[Coo] Cooper, G. *The Cell: A Molecular Approach*.

[Cri70] Crick, F. "Central Dogma of Molecular Biology". In: *Nature* vol. 227, no. 5258 (5258 Aug. 1970), pp. 561–563.

[Edg04] Edgar, R. C. "MUSCLE: Multiple Sequence Alignment with High Accuracy and High Throughput". In: *Nucleic Acids Research* vol. 32, no. 5 (Mar. 1, 2004), pp. 1792–1797.

[FM05] Ferragina, P. and Manzini, G. "Indexing Compressed Text". In: *Journal of the ACM* vol. 52, no. 4 (July 1, 2005), pp. 552–581.

[Gia+17] Giacopuzzi, E. et al. "Exome Sequencing in Schizophrenic Patients with High Levels of Homozygosity Identifies Novel and Extremely Rare Mutations in the GABA/Glutamatergic Pathways". In: *PLOS ONE* vol. 12, no. 8 (Aug. 7, 2017), e0182778.

[Gus89] Gusella, J. F. "Location Cloning Strategy for Characterizing Genetic Defects in Huntington's Disease and Alzheimer's Disease". In: *The FASEB Journal* vol. 3, no. 9 (1989), pp. 2036–2041.

[Haz] Hazen, R. M. *The Story of Earth: The First 4.5 Billion Years, from Stardust to Living Planet*.

[Hon+07] Hon, W.-K., Lam, T.-W., Sadakane, K., Sung, W.-K., and Yiu, S.-M. "A Space and Time Efficient Algorithm for Constructing Compressed Suffix Arrays". In: *Algorithmica* vol. 48, no. 1 (May 1, 2007), pp. 23–36.

[How+20] Howrigan, D. P. et al. "Exome Sequencing in Schizophrenia-Affected Parent–Offspring Trios Reveals Risk Conferred by Protein-Coding de Novo Mutations". In: *Nature Neuroscience* vol. 23, no. 2 (2 Feb. 2020), pp. 185–193.

[Huc+19] Huckins, L. M. et al. "Gene Expression Imputation across Multiple Brain Regions Provides Insights into Schizophrenia Risk". In: *Nature Genetics* vol. 51, no. 4 (4 Apr. 2019), pp. 659–674.

[HW05] Harrison, P. J. and Weinberger, D. R. "Schizophrenia Genes, Gene Expression, and Neuropathology: On the Matter of Their Convergence". In: *Molecular Psychiatry* vol. 10, no. 1 (1 Jan. 2005), pp. 40–68.

[Jia+10] Jia, P., Wang, L., Meltzer, H. Y., and Zhao, Z. "Common Variants Conferring Risk of Schizophrenia: A Pathway Analysis of GWAS Data". In: *Schizophrenia Research* vol. 122, no. 1 (Sept. 1, 2010), pp. 38–42.

[Jin+12] Jinek, M., Chylinski, K., Fonfara, I., Hauer, M., Doudna, J. A., and Charpentier, E. "A Programmable Dual-RNA–Guided DNA Endonuclease in Adaptive Bacterial Immunity". In: *Science* vol. 337, no. 6096 (Aug. 17, 2012), pp. 816–821. pmid: 22745249.

[KB13] Kilpinen, H. and Barrett, J. C. "How Next-Generation Sequencing Is Transforming Complex Disease Genetics". In: *Trends in Genetics* vol. 29, no. 1 (Jan. 1, 2013), pp. 23–30.

[Kha+15] Khafizov, K., Ivanov, M. V., Glazova, O. V., and Kovalenko, S. P. "Computational Approaches to Study the Effects of Small Genomic Variations". In: *Journal of Molecular Modeling* vol. 21, no. 10 (Sept. 8, 2015), p. 251.

[LD09] Li, H. and Durbin, R. "Fast and Accurate Short Read Alignment with Burrows–Wheeler Transform". In: *Bioinformatics* vol. 25, no. 14 (July 15, 2009), pp. 1754–1760.

[LD10] Li, H. and Durbin, R. "Fast and Accurate Long-Read Alignment with Burrows–Wheeler Transform". In: *Bioinformatics* vol. 26, no. 5 (Mar. 1, 2010), pp. 589–595.

[Li13] Li, H. *Aligning Sequence Reads, Clone Sequences and Assembly Contigs with BWA-MEM*. May 26, 2013. arXiv: 1303.3997 [q-bio]. URL: http://arxiv.org/abs/1303.3997 (visited on 09/27/2020).

[Lod+07] Lodish, H., Berk, A., Kaiser, C. A., Krieger, M., Scott, M. P., Bretscher, A., Ploegh, H., and Matsudaira, P. *Molecular Cell Biology*. 6th edition. New York: W. H. Freeman, June 15, 2007. 973 pp.

[Ma+17] Ma, H. et al. "Correction of a Pathogenic Gene Mutation in Human Embryos". In: *Nature* vol. 548, no. 7668 (7668 Aug. 2017), pp. 413–419.

[Mis+14] Misale, C., Ferrero, G., Torquati, M., and Aldinucci, M. *Sequence Alignment Tools: One Parallel Pattern to Rule Them All?* BioMed Research International. July 24, 2014.

[MM93] Manber, U. and Myers, G. "Suffix Arrays: A New Method for On-Line String Searches". In: *SIAM Journal on Computing* vol. 22, no. 5 (Oct. 1, 1993), pp. 935–948.

[Mor99] Morgenstern, B. "DIALIGN 2: Improvement of the Segment-to-Segment Approach to Multiple Sequence Alignment.". In: *Bioinformatics* vol. 15, no. 3 (Mar. 1, 1999), pp. 211–218.

[NHH00] Notredame, C., Higgins, D. G., and Heringa, J. "T-Coffee: A Novel Method for Fast and Accurate Multiple Sequence Alignment11Edited by J. Thornton". In: *Journal of Molecular Biology* vol. 302, no. 1 (Sept. 8, 2000), pp. 205–217.

[Nis+18] Nishioka, M. et al. "Identification of Somatic Mutations in Monozygotic Twins Discordant for Psychiatric Disorders". In: *npj Schizophrenia* vol. 4, no. 1 (1 Apr. 13, 2018), pp. 1–7.

[OS09] Okanohara, D. and Sadakane, K. *A Linear-Time Burrows-Wheeler Transform Using Induced Sorting*. Vol. 5721. Aug. 25, 2009, p. 101. 90 pp.

[Ott01] Ott, J. “10 Major Strengths and Weaknesses of the Lod Score Method”. In: *Advances in Genetics*. Vol. 42. Academic Press, Jan. 1, 2001, pp. 125–132.

[Per+18] Pertea, M., Shumate, A., Pertea, G., Varabyou, A., Chang, Y.-C., Madugundu, A. K., Pandey, A., and Salzberg, S. L. “Thousands of Large-Scale RNA Sequencing Experiments Yield a Comprehensive New Human Gene List and Reveal Extensive Transcriptional Noise”. In: *bioRxiv* (May 29, 2018), p. 332825.

[Pev15] Pevsner, J. *Bioinformatics and Functional Genomics*. 3rd Edition. Chichester, West Sussex, UK ; Hoboken, New Jersey: Wiley-Blackwell, Oct. 26, 2015. 1160 pp.

[RBA18] Ren, S., Bertels, K., and Al-Ars, Z. “Efficient Acceleration of the Pair-HMMs Forward Algorithm for GATK HaplotypeCaller on Graphics Processing Units”. In: *Evolutionary Bioinformatics* vol. 14 (Jan. 1, 2018), p. 1176934318760543.

[Rip+14] Ripke, S. et al. “Biological Insights from 108 Schizophrenia-Associated Genetic Loci”. In: *Nature* vol. 511, no. 7510 (7510 July 2014), pp. 421–427.

[San75] Sankoff, D. “Minimal Mutation Trees of Sequences”. In: *SIAM Journal on Applied Mathematics* vol. 28, no. 1 (Jan. 1, 1975), pp. 35–42.

[Sch08] Schuster, S. C. “Next-Generation Sequencing Transforms Today’s Biology”. In: *Nature Methods* vol. 5, no. 1 (1 Jan. 2008), pp. 16–18.

[Sek+16] Sekar, A. et al. “Schizophrenia Risk from Complex Variation of Complement Component 4”. In: *Nature* vol. 530, no. 7589 (7589 Feb. 2016), pp. 177–183.

[SW81] Smith, T. and Waterman, M. “Identification of Common Molecular Subsequences”. In: *Journal of Molecular Biology* vol. 147, no. 1 (Mar. 1981), pp. 195–197.

[THG94] Thompson, J. D., Higgins, D. G., and Gibson, T. J. “CLUSTAL W: Improving the Sensitivity of Progressive Multiple Sequence Alignment through Sequence Weighting, Position-Specific Gap Penalties and Weight Matrix Choice”. In: *Nucleic Acids Research* vol. 22, no. 22 (Nov. 11, 1994), pp. 4673–4680.

[TRM13] Thorvaldsdóttir, H., Robinson, J. T., and Mesirov, J. P. “Integrative Genomics Viewer (IGV): High-Performance Genomics Data Visualization and Exploration”. In: *Briefings in Bioinformatics* vol. 14, no. 2 (Mar. 1, 2013), pp. 178–192.

[Vas+16] Vaser, R., Adusumalli, S., Leng, S. N., Sikic, M., and Ng, P. C. "SIFT Missense Predictions for Genomes". In: *Nature Protocols* vol. 11, no. 1 (1 Jan. 2016), pp. 1–9.

[Vas+19] Vasimuddin, M., Misra, S., Li, H., and Aluru, S. "Efficient Architecture-Aware Acceleration of BWA-MEM for Multicore Systems". In: *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS). May 2019, pp. 314–324.

[Wal+08] Walsh, T. et al. "Rare Structural Variants Disrupt Multiple Genes in Neurodevelopmental Pathways in Schizophrenia". In: *Science* vol. 320, no. 5875 (Apr. 25, 2008), pp. 539–543. pmid: 18369103.

[WC53] Watson, J. D. and Crick, F. H. C. "Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid". In: *Nature* vol. 171, no. 4356 (4356 Apr. 1953), pp. 737–738.

[Wu+20] Wu, F. et al. "A New Coronavirus Associated with Human Respiratory Disease in China". In: *Nature* vol. 579, no. 7798 (7798 Mar. 2020), pp. 265–269.

[YFK20] Yuki, K., Fujiogi, M., and Koutsogiannaki, S. "COVID-19 Pathophysiology: A Review". In: *Clinical Immunology* vol. 215 (June 1, 2020), p. 108427.

Phụ lục

Log file

Xem xét quá trình tiền xử lý mẫu SRR5344685 qua một số phần được ghi lại trong log file sau khi sử dụng các công cụ.

Tiền xử lý dữ liệu

Sử dụng công cụ FastQC Read Quality reports trên nền tảng Galaxy để kiểm tra chất lượng mẫu. Chỉ hiển thị chất lượng trình tự trên mỗi bazơ.

##FastQC 0.11.8
>>Basic Statistics pass
#Measure Value
Filename SRR5344685 _fastq-dump_.gz
File type Conventional base calls
Encoding Sanger / Illumina 1.9
Total Sequences 39248833
Sequences flagged as poor quality 0
Sequence length 8-378
%GC 50
>>END_MODULE
>>Per base sequence quality warn

#Base	Mean	Median	Lower	Upper	10th	90th
1	25.8496038850378	27.0	25.0	28.0	22.0	28.0
2	24.9572153903276	26.0	23.0	28.0	20.0	29.0
3	24.8664657876579	26.0	23.0	28.0	20.0	29.0
4	24.8165187229898	27.0	23.0	28.0	19.0	29.0
5	24.8486185309000	27.0	23.0	28.0	20.0	29.0
6	24.9495256330296	27.0	23.0	28.0	20.0	29.0
7	25.0646503553366	27.0	23.0	28.0	20.0	29.0
8	25.0764975101297	27.0	23.0	28.0	20.0	29.0
9	25.1540818923811	27.0	23.0	28.0	20.0	29.0
10-19	25.3615256782883	27.0	23.0	28.0	20.0	29.0
20-29	25.2348815046644	27.0	23.0	28.0	20.0	29.0
30-39	24.9920492588028	27.0	23.0	28.0	19.8	29.0
40-49	24.7752272184731	26.3	23.0	28.0	19.0	29.0
50-59	24.6085790658247	26.0	23.0	28.0	18.8	29.0

60-69	24.4427113141152	26.0	22.6	28.0	18.0	29.0
70-79	24.2570140203879	26.0	22.0	28.0	17.8	29.0
80-89	24.0669791700910	25.6	22.0	28.0	17.0	28.6
90-99	23.8855627942413	25.0	22.0	27.1	16.1	28.0
100-109	23.7129692874507	25.0	21.6	27.0	16.0	28.0
110-119	23.5734338060454	25.0	21.0	27.0	15.7	28.0
120-129	23.4550904699750	25.0	21.0	27.0	15.0	28.0
130-139	23.3427482309084	25.0	21.0	27.0	15.0	28.0
140-149	23.2162013271700	24.0	21.0	27.0	14.8	28.0
150-159	23.0875326545610	24.0	20.9	27.0	14.0	28.0
160-169	22.9645043165903	24.0	20.0	27.0	14.0	28.0
170-179	22.8047324697658	24.0	20.0	27.0	14.0	28.0
180-189	22.5716498928104	23.9	20.0	27.0	13.9	28.0
190-199	22.2914323098561	23.0	19.9	27.0	13.0	28.0
200-209	22.0258855128091	23.0	19.0	26.3	13.0	28.0
210-219	21.7141983233292	23.0	19.0	26.0	13.0	28.0
220-229	21.3119374479132	22.2	18.2	25.7	12.5	27.3
230-239	20.6295231704006	21.6	17.6	25.0	12.0	27.0
240-249	19.8026313932172	20.6	16.1	24.0	12.0	26.7
250-259	19.4501657485137	20.0	16.0	24.0	12.0	26.0
260-269	19.5459985118248	20.0	16.0	24.0	12.0	26.7
270-279	19.9537820451112	20.6	16.0	24.7	12.0	27.0
280-289	21.1357194677598	22.1	17.2	26.5	12.2	27.9
290-299	23.3676493332425	25.9	20.4	27.4	14.2	28.0
300-309	25.8621782023115	27.0	25.8	28.0	21.6	28.0
310-319	26.7026237515121	27.0	26.0	28.0	25.2	28.0
320-329	26.9728573386942	27.0	26.0	28.0	25.9	28.2
330-339	26.9449208847956	27.0	26.0	28.0	25.7	28.4
340-349	26.8419355812905	27.0	26.0	28.0	25.0	28.9
350-359	26.6897038460963	27.0	26.0	28.0	24.5	29.0
360-369	26.4412936728482	27.0	26.0	28.0	22.9	29.0
370-378	17.1573855902215	20.7	20.0	21.4	17.9	22.1

>>END_MODULE

Sử dụng Trimomatic trên nền tảng Galaxy phiên bản 0.38.0 để loại bỏ các bazơ có điểm chất lượng thấp hơn 24 ở phần đuôi của các trình tự.

```
## Trimomatic  Galaxy Version 0.38.0
Filename SRR5344685 _fastq-dump_.gz
Picked up _JAVA_OPTIONS: -Djava.io.tmpdir=/galaxy-repl/main/jobdir/030/844
/30844645/_job_tmp -Xmx28g  -Xms256m
TrimmomaticSE: Started with arguments:
 -threads 6 fastq_in.fastqsanger.gz fastq_out.fastqsanger.gz TRAILING:24
Quality encoding detected as phred33
Input Reads: 39248833 Surviving: 39099921 (99.62%) Dropped: 148912 (0.38%)
TrimmomaticSE: Completed successfully
```

Dóng hàng trình tự với công cụ BWA-MEM trên nền tảng Galaxy phiên bản 0.7.17.1.

```
[bwa_index] Pack FASTA... 34.47 sec
[bwa_index] Construct BWT for the packed sequence...
[BWTIncCreate] textLength=6544178410, availableWord=472472396
[BWTIncConstructFromPacked] 10 iterations done. 99999994 characters processed.
[BWTIncConstructFromPack
```

Xác định biến thể

Gọi các biến thể bằng công cụ Haplotype Caller của GATK phiên bản 4.1.4.1.

```
0 read(s) filtered by: MappingQualityAvailableReadFilter
0 read(s) filtered by: MappedReadFilter
0 read(s) filtered by: NotSecondaryAlignmentReadFilter
0 read(s) filtered by: NotDuplicateReadFilter
0 read(s) filtered by: PassesVendorQualityCheckReadFilter
0 read(s) filtered by: NonZeroReferenceLengthAlignmentReadFilter
0 read(s) filtered by: GoodCigarReadFilter
0 read(s) filtered by: WellformedReadFilter
0 total reads filtered
02:34:05.604 INFO  ProgressMeter - NC_012920.1:15664    1015.9     11723406    11539.4
02:34:05.604 INFO  ProgressMeter - Traversal complete.
Processed 11723406 total regions in 1015.9 minutes.
02:34:05.742 INFO  VectorLoglessPairHMM - Time spent in setup for JNI call : 30.405236508
02:34:05.742 INFO  PairHMM - Total compute time in PairHMM
computeLogLikelihoods() : 14020.819401494
02:34:05.742 INFO  SmithWatermanAligner - Total compute time in java Smith-Waterman :
33112.29 sec
02:34:05.742 INFO  HaplotypeCaller - Shutting down engine
[November 20, 2020 2:34:05 AM ICT] org.broadinstitute.hellbender.tools.walkers.
haplotypecaller.HaplotypeCaller done. Elapsed time: 1,015.96 minutes.
Runtime.totalMemory()=7897350144
```

Kếp hợp các tập biến thể của 7 mẫu bằng công cụ GenomicsDBImport phiên bản 4.0.10.0.

12:24:45.754 INFO GenomicsDBImport - Importing batch 1 with 7 samples
12:25:54.657 INFO GenomicsDBImport - Importing batch 1 with 7 samples
12:27:23.784 INFO GenomicsDBImport - Importing batch 1 with 7 samples
12:27:53.609 INFO GenomicsDBImport - Importing batch 1 with 7 samples
12:29:32.465 INFO GenomicsDBImport - Importing batch 1 with 7 samples
12:30:23.044 INFO GenomicsDBImport - Importing batch 1 with 7 samples
12:30:42.353 INFO GenomicsDBImport - Importing batch 1 with 7 samples
12:31:24.048 INFO GenomicsDBImport - Importing batch 1 with 7 samples
12:32:20.918 INFO GenomicsDBImport - Importing batch 1 with 7 samples
12:32:22.174 INFO ProgressMeter - NC_000001.11:1 27.7 1 0.0
12:32:22.174 INFO GenomicsDBImport - Done importing batch 1/1
12:32:22.175 INFO ProgressMeter - NC_000001.11:1 27.7 1 0.0
12:32:22.175 INFO ProgressMeter - Traversal complete. Processed 1 total batches in 27.7 minutes.
12:32:22.175 INFO GenomicsDBImport - Import completed!
12:32:22.175 INFO GenomicsDBImport - Shutting down engine
[November 24, 2020 12:32:22 PM ICT] org.broadinstitute.hellbender.tools.genomicsdb.GenomicsDBImport done. Elapsed time: 27.69 minutes.
Runtime.totalMemory()=3080192000
Tool returned:
true

Chú thích chức năng

Hiệu chỉnh biến thể bằng công cụ VariantRecalibrator phiên bản 4.0.10.0

Indels
07:19:17.654 INFO ProgressMeter - Traversal complete.
Processed 1441213 total variants in 166.4 minutes.
07:19:17.710 INFO VariantDataManager - QD: mean = 24.03 standard deviation = 9.18
07:19:17.785 INFO VariantDataManager - MQRankSum: mean = 0.00 standard deviation = 0.00
07:19:17.823 INFO VariantDataManager - ReadPosRankSum: mean = 0.14
standard deviation = 1.36
07:19:17.861 INFO VariantDataManager - FS: mean = 4.61 standard deviation = 20.24
07:19:17.887 INFO VariantDataManager - MQ: mean = 60.00 standard deviation = 0.00
07:19:17.913 INFO VariantDataManager - SOR: mean = 1.75 standard deviation = 1.18
07:19:17.938 INFO VariantDataManager - DP: mean = 156.93 standard deviation = 204.57
07:19:17.955 INFO VariantRecalibrator - Shutting down engine
[November 26, 2020 7:19:17 AM ICT] org.broadinstitute.hellbender.tools.walkers.vqsr.VariantRecalibrator done. Elapsed time: 166.38 minutes.
Runtime.totalMemory()=3043491840
SNPs
06:25:38.572 INFO ProgressMeter - Traversal complete.
Processed 1441213 total variants in 113.7 minutes.
06:25:38.742 INFO VariantDataManager - FS: mean = 36.65 standard deviation = 75.99

06:25:38.946 INFO VariantDataManager - ReadPosRankSum: mean = -0.47
standard deviation = 2.10
06:25:39.142 INFO VariantDataManager - MQRankSum: mean = 0.00
standard deviation = 0.00
06:25:39.357 INFO VariantDataManager - QD: mean = 7.26 standard deviation = 8.88
06:25:39.535 INFO VariantDataManager - SOR: mean = 2.23 standard deviation = 1.93
06:25:39.706 INFO VariantDataManager - DP: mean = 284.20 standard deviation = 189.48
06:25:39.783 INFO VariantRecalibrator - Shutting down engine
[November 26, 2020 6:25:39 AM ICT] org.broadinstitute.hellbender.tools.walkers.vqsr.
VariantRecalibrator done. Elapsed time: 113.70 minutes.
Runtime.totalMemory()=25607274496

Mã nguồn

Mã nguồn thực thi trên máy chủ Linux của các bước chính được liệt kê bên dưới không bao gồm: FastQC, Trimomatic, BWA-MEM trên nền tảng Galaxy, tải dữ liệu, đổi tên nhiễm sắc thể, và đánh chỉ số một số tệp định dạng bam.

```
1 Pre-calculating input of reference genome
# Index reference genome
samtools faidx GRCh38.fasta
# Create dictionary
gatk CreateSequenceDictionary -R GRCh38.fasta

2 Pre-calculating input of SRA
samtools index sr-85.bam

3 Base Recalibration
# Change chromosome name of vcf file
bcftools annotate --threads 64 --rename-chrs hg38-to-b38.txt
dbSNPs.vcf > GRCh38_SNPs.vcf
# Index vcf file
bgzip -c GRCh38_SNPs.vcf > GRCh38_SNPs.vcf.gz
tabix -fp vcf GRCh38_SNPs.vcf.gz

# Create table
gatk BaseRecalibrator -I sr-85.bam -R GRCh38.fasta
--known-sites GRCh38_SNPs.vcf.gz
-O recal-85.table

# Apply BQSR
gatk ApplyBQSR -R GRCh38.fasta -I sr-85.bam
--bqsr-recal-file recal-85.table
-O recal-85.bam

4 Filtering reads having mapQ < 60
samtools view -@ 64 -bq recal-85.bam > fil-85.bam
```

5 Calling variants
gatk --java-options "-Xmx16g -XX:ParallelGCThreads=64" HaplotypeCaller
--native-pair-hmm-threads 64 --min-base-quality-score 20
-R GRCh38.fasta -I fil-85.bam
-O var_85.g.vcf.gz -ERC GVCF
-G StandardAnnotation -G AS_StandardAnnotation -G StandardHCAnnotation

6 Consolidating GVCFs
gatk --java-options "-Xmx4g" GenomicsDBImport
-V var-85.g.vcf.gz -V var-86.g.vcf.gz -V var-87.g.vcf.gz
-V var-88.g.vcf.gz -V var-89.g.vcf.gz -V var-90.g.vcf.gz
-V var-91.g.vcf.gz
--genomicsdb-workspace-path my_database
--intervals intervals.list

gatk --java-options "-Xmx4g" GenotypeGVCFs -R GRCh38.fasta
-V gendb://my_database --interval-set-rule INTERSECTION
-O all-var.vcf.gz

7 Variant Recalibration
Filtering Heterozygous
gatk --java-options "-Xmx3g -Xms3g" VariantFiltration -V var.vcf
--filter-expression "ExcessHet > 54.69"
--filter-name ExcessHet --missing-values-evaluate-as-failing true
-O cohort_excesshet.vcf.gz

Exclude information samples
gatk MakeSitesOnlyVcf -I cohort_excesshet.vcf.gz -O cohort_siteonly.vcf.gz

Calculate VQSLOD tranches for indels
gatk --java-options "-Xmx24g -Xms24g" VariantRecalibrator -V cohort_siteonly.vcf.gz
--trust-all-polymorphic -tranche 100.0 -tranche 99.95 -tranche 99.9 -tranche 99.5
-tranche 99.0 -tranche 97.0 -tranche 96.0 -tranche 95.0 -tranche 94.0
-tranche 93.5 -tranche 93.0 -tranche 92.0 -tranche 91.0 -tranche 90.0
-an FS -an ReadPosRankSum -an QD -an SOR
-mode INDEL --max-gaussians 4
-resource:mills,known=false,training=true,truth=true,prior=12
Mills_and_1000G_gold_standard.indels.b38.primary_assembly_change.vcf.gz
-resource:axiomPoly,known=false,training=true,truth=false,prior=10
hg38_v0_Axiom_Exome_Plus.genotypes.all_populations.poly.hg38_change.vcf.gz
-resource:dbsnp,known=true,training=false,truth=false,prior=2 GRCh38_SNPs.vcf.gz
-O cohort_indels.recal
--tranches-file cohort_indels.tranches

Calculate VQSLOD tranches for snps
gatk --java-options "-Xmx3g -Xms3g" VariantRecalibrator -V cohort_siteonly.vcf.gz
--trust-all-polymorphic -tranche 100.0 -tranche 99.95 -tranche 99.9 -tranche 99.8
-tranche 99.6 -tranche 99.5 -tranche 99.4 -tranche 99.3 -tranche 99.0 -tranche 98.0
-tranche 97.0 -tranche 90.0
-an QD -an ReadPosRankSum -an FS -an SOR
-mode SNP --max-gaussians 6

-resource:hapmap,known=false,training=true,truth=true,prior=15
hg38_v0_hapmap_3.3.hg38_change.vcf.gz
-resource:omni,known=false,training=true,truth=true,prior=12
hg38_v0_1000G_omni2.5.hg38_change.vcf.gz
-resource:1000G,known=false,training=true,truth=false,prior=10
hg38_v0_1000G_phase1.snps.high_confidence.hg38_change.vcf.gz
-resource:dbsnp,known=true,training=false,truth=false,prior=7 GRCh38_SNPs.vcf.gz
-O cohort_snps.recal
--tranches-file cohort_snps.tranches

Apply VQSR
gatk --java-options "-Xmx5g -Xms5g" ApplyVQSR -V cohort_excesshet.vcf.gz
--recal-file cohort_indels.recal --tranches-file cohort_indels.tranches
--truth-sensitivity-filter-level 99.7 --create-output-variant-index true
-mode INDEL -O indel.recalibrated.vcf.gz

gatk --java-options "-Xmx5g -Xms5g" ApplyVQSR -V indel.recalibrated.vcf.gz
--recal-file cohort_snps.recal --tranches-file cohort_snps.tranches
--truth-sensitivity-filter-level 99.7 --create-output-variant-index true
-mode SNP -O snps-indels.recalibrated.vcf.gz

8 Variant Annotation using Funcotator
Note: Change chromosome name of reference genome and snps-indels.recalibrated.vcf.gz to fit hg38 before implementing Funcotator.

gatk Funcotator -R GRCh38-funco.fasta -V snps-indels-funco.recalibrated.vcf.gz
-O funco-var --output-file-format MAF
--data-sources-path funcotator_dataSources.v1.7.20200521g
--ref-version hg38

9 Predicting the impact level of variation using SIF4G
Note: Change chromosome name of snps-indels.recalibrated.vcf.gz to fit data of SIFT4G before implementing
java -jar SIFT4G_Annotator.jar -c -i snps-indels-sift.recalibrated.vcf.gz
-d -d <Path to SIFT4G database directory> -r <Path to results folder>

Mục từ tra cứu

R

S

T

U

V

W

X