Penerapan Algoritma Brute Force pada Proses Pengurutan Genom Manusia

Martino Christanto Khuangga - 13514084
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514084@std.stei.itb.ac.id

***Abstract*—Dewasa ini, penggunaan komputer dalam dunia medis makin berkembang. Penggunaan komputer ini tidak hanya sebatas dalam proses operasi pasien seperti masyarakat melihat pada umumnya, tetapi juga dalam lingkup penelitian sel, gen, DNA, dan berbagai komponen biologis kecil lainnya. Penggunaan komputer dalam lingkup terkecil biologi tersebut biasa ditemui dalam bidang biomedik, atau lebih spesifik lagi bioinformatika. Dalam bioinformatika, tentunya digunakan berbagai macam algoritma untuk melakukan komputasi, salah satu algoritma yang digunakan adalah *brute force*. Makalah ini akan membahas penerapan algoritma *brute force* dalam proses pengurutan genom manusa (*human genome sequencing*)**

Keywords—pengurutan genom, algoritma brute force, bioinformatika

I. PENDAHULUAN

Salah satu cabang dalam informatika adalah bioinformatika. Bioinformatika adalah penerapan teknologi komputer pada manajemen informasi biologis. Tujuan dari penggunaan komputer ini adalah untuk menyimpan, menganalisis, dan menerapkan biologi dan informasi genetik untuk penemuan dan pengembangan teknologi kesehatan baik dalam bentuk obat atau terapi [1]. Penggunaan komputer dalam bidang biologi ini sangat membantu para peneliti, mengingat kecepatan komputer zaman sekarang dalam melakukan komputasi. Para peneliti tidak perlu lagi mengamati satu persatu informasi genetik yang terdapat dalam suatu gen. Dengan demikian adanya cabang bioinformatika ini sangat meningkatkan efisiensi dari penelitian berbasis biologi.

Kumpulan informasi genetik dalam tubuh manusia tersimpan di dalam genom. Kumpulan informasi ini disusun oleh basa nukleotida yang menyusun DNA, yaitu Adenin (A), Thymin (T), Guanin (G), Sitosin (C) [4]. Hampir setiap sel dalam tubuh suatu spesies mengandung genom, sehingga hal ini masih menjadi salah satu rintangan dalam pengurutan genom, karena jumlah sel dalam tubuh manusia sangat banyak. Hal lain yang menjadi rintangan adalah para peneliti tidak tahu urutan informasi dalam suatu genom, sehingga para peneliti harus menyusun satu persatu "potongan" informasi yang didapat untuk mendapatkan informasi genetik asli dari suatu organisme.

Rintangan ini dialami para peneliti dalam salah satu projek terbesar yang ada dalam bidang bioinformatika, yaitu *Human Genome Project (HGP)*, yang merupakan sebuah projek untuk mengurutkan (*sequencing*) dan memetakan (*mapping*) seluruh genom manusia (*Homo sapiens*) [3]. Untuk mengatasi rintangan tersebut, para peneliti di bidang bioinformatika mencari algoritma yang tepat. Salah satu algoritma awal yang digunakan para peneliti adalah *brute force*.

Namun, *brute force* adalah algoritma yang sangat memakan waktu karena *brute force* mencoba setiap kemungkinan yang ada. Padahal dalam satu genom, ada sekitar 3 milyar informasi yang tersimpan di dalamnya [2]. Para peneliti akhirnya menciptakan dan menggunakan berbagai macam algoritma lainnya untuk menggantikan algoritma *brute force* dalam HGP. Algoritma-algoritma tersebut tidak akan dibahas dalam makalah ini, karena algoritma-algoritma tersebut sudah di luar bahasan makalah ini.

Dengan dibuatnya makalah ini, pembaca diharapkan mengetahui penggunaan algoritma *brute force* dalam proses pengurutan dan pemetaan genom manusia. Selain itu, pembaca juga diharapkan mengetahui kelebihan dan kelemahan algoritma *brute force* dalam HGP.

II. DASAR TEORI

A. Algoritma Brute Force

Brute force adalah algoritma yang paling sederhana dalam proses pemecahan masalah [5]. Algoritma *brute force* adalah algoritma yang mencoba segala kemungkinan yang ada untuk mendapatkan solusi. Dengan demikian lama waktu eksekusi algoritma *brute force* bergantung pada jumlah datanya. Dalam teori kompleksitas algoritma, umumnya algoritma *brute force* memiliki kompleksitas yang buruk dalam proses pemecahan masalah.

Walaupun algoritma *brute force* tidak cepat, tetapi algoritma *brute force* dapat memecahkan sebagian besar masalah dalam pemrograman, bahkan ada beberapa persoalan yang hanya dapat diselesaikan dengan algoritma *brute force* [5]. Hal ini menyebebabkan algoritma *brute force* banyak digunakan untuk persoalan-persoalan yang kecil.

Salah satu istilah lain yang berkaitan dengan *brute force* adalah *exhaustive search*. *Exhaustive search* adalah proses pencarian solusi secara *brute force* untuk masalah khusus, seperti mencari himpunan bagian, kombinasi, permutasi, dan lain-lain [5]. Dengan demikian, dapat disimpulkan bahwa *exhaustive search* adalah turunan/himpunan bagian dari *brute force*, karena tidak semua *brute force* mencari objek-objek kombinatorik.

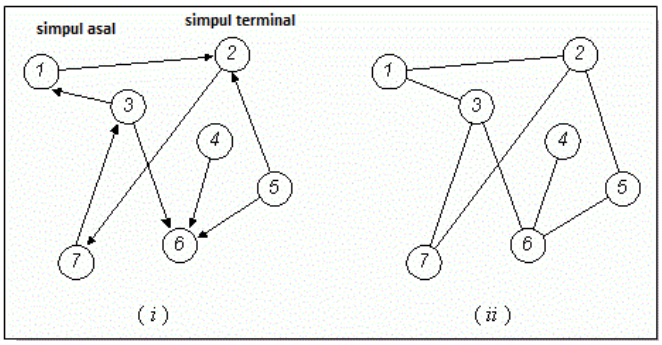
Ada beberapa cara untuk mempercepat algoritma *exhaustive search*. Salah satunya dengan menggunakan teknik heuristik [5]. Teknik heuristik adalah cara yang sedikit berlawanan dengan algoritma yang digunakan. Dalam ilmu komputer, heuristic dirancang untuk memecahkan masalah dengan cara mengabaikan apakah solusi yang dihasilkan dapat dibuktikan benar secara matematis[5].

Pada algoritma *brute force*, misalnya, dalam mencari anagram, program tidak perlu memeriksa pasangan huruf-huruf yang jarang digunakan dalam suatu bahasa. Misalkan dalam kata *charm* dan *march*. Untuk mendapatkannya dengan cepat, heuristik yang digunakan adalah menyatukan huruf c dan h karena dalam bahasa Inggris, huruf c dan h selalu berdampingan [5].

Pada penerapannya, teknik heuristik ini tidak selalu menghasilkan solusi [5]. Karena itu, teknik heuristik ini harus digunakan pada kondisi yang tepat. Selain itu, pemilihan metode heuristik dalam algoritma *brute force* juga harus diperhatikan agar tidak merusak ketepatan dari algoritma *brute force*.

B. *Graf*

Graf adalah suatu struktur yang menggambarkan objek-objek yang diskrit, dilambangkan dengan titik yang disebut dengan simpul, serta hubungan antara objek-objek tersebut, yang dihubungkan dengan garis dengan disebut dengan sisi [7]. Definisi dari graf adalah sebuah hiimpunan yang tidak kosong dari simpul (V) dan sebuah himpunan yang mungkin kosong dari sisi (E) [7].



Gambar 1 (i) Graf berarah (ii) Graf tidak berarah
Sumber : http://aimyaya.com/id/komputer/sekilas-mengenai-path-cycle-dan-hamiltonian-cycle/

Berdasarkan simpul dan sisinya, graf bisa dibagi menjadi banyak jenis. Beberapa yang akan dibahas adalah graf berarah dan tidak berarah. Graf berarah adalah graf yang sisi-sisinya memiliki arah atau biasa disebut busur. Dalam graf berarah, ada sebuah istilah yang disebut dengan simpul asal dan simpul terminal. Simpul asal adalah simpul yang bersisian dengan pangkal dari busur, sedangkan simpul terminal adalah simpul yang bersisian dengan ujuang panah dari busur. Sedangkan graf tidak berarah adalah graf yang sisi-sisinya tidak memiliki arah [6]. Dengan demikian suatu simpul pada graf berarah tidak dapat dicapai jika tidak ada sisi yang mengarah ke simpul tersebut. Berbeda dengan graf tak-berarah yang simpulnya dapat dicapai selama ada sisi yang terhubung dengan dirinya.

Ada dua terminologi graf yang akan dibahas dalam makalah ini, yaitu lintasan Euler dan lintasan Hamilton. Lintasan Euler adalah lintasan dalam graf yang melalui setiap sisinya tepat satu kali. Sedangkan lintasan Hamilton adalah lintasan dalam graf yang melalui setiap simpulnya tepat sati kali [6]. Suatu graf bisa saja memiliki lintasan Euler dan Hamilton tertutup, atau lebih dikenal dengan istilah sirkuit.

Sampai saat ini, belum ada algoritma yang mangkus dalam proses penyelesaian masalah graf Euler dan Hamilton. Salah satu algoritma yang pasti menemukan solusi adalah algoritma *brute force*, seperti yang telah dijelaskan di poin sebelumnya.

C. *Pengurutan Genom*

Pengurutan genom adalah proses untuk mendapatkan informasi genom secara utuh dari potongan-potongan genom yang didapat [8]. Proses pengurutan genom mirip dengan menyusun sebuah *puzzle*, perbedaannya adalah *puzzle* dicari pasangannya dengan melihat basa nukleotida yang *overlap*. *Overlap* artinya potongan genom pertama memiliki suffiks yang sama dengan prefiks dari potongan genom kedua.

Pengurutan genom menjadi sangat penting dalam penelitian kesehatan saat ini karena genom menyimpan informasi genetik dari suatu spesies atau dalam hal ini kita sebut saja manusia. Dengan adanya informasi genetik tersebut para peneliti dapat mendapatkan bagian mana dari genom yang menentukan sifat genetik suatu manusia [8]. Dengan didapatkannya informasi ini, para peneliti nantinya dapat melakukan rekayasa genetik untuk pembuatan obat-obatan baru yang akan membawa perkembangan dalam dunia kesehatan. Selain itu, pengurutan genom juga bermanfaat dalam mengetahui sistem kerja gen dalam mengatur tubuh manusia [8].

Proses pengurutan genom tidak dapat dilakukan dalam sekejap, tetapi harus dipotong-potong menjadi bagian yang lebih kecil tersebut terlebih dahulu, kemudian baru diurutkan kembali [8]. Ada dua pendekatan dalam proses pengurutan genom ini, yang pertama adalah pendekatan *"clone-by-clone*" dan yang kedua adalah pendekatan *"whole-genome shotgun*".

Pendekatan "*clone-by-clone*" memecah genome menjadi potongan-potongan yang berukuran relatif besar, potongan ini disebut dengan *clone*. Disebut relatif besar karena ukuran dari potongan ini adalah sekitar 150.000 pasangan basa [8]. Kemudian potongan-potongan tersebut disusun untuk

membentuk kembali genom.

Berbeda dengan pendekatan "*clone-by-clone*", pendekatan *"whole-genome shotgun*" memecah genom menjadi pecahan-pecahan yang kecil [8]. Kemudian menyatukannya kembali dengan melihat *overlap*.

Pendekatan *clone-by-clone* terlihat lebih mudah namun sulit dalam proses pemetaannya. Sebaliknya, pendekatan *whole genome shotgun* dapat memecah genom dengan cara yang mudah, tetapi memiliki waktu yang lama dalam menyatukan potongan-potongan itu kembali.

Pada praktiknya, proses pengurutan genom masih dapat mengalami kesalahan. Hal ini dikarenakan pada saat proses pengurutan, komputer tidak mengetahui posisi asli dari potongan-potongan tersebut pada awalnya, sehingga bisa jadi pada proses pengurutan genom ini terjadi kesalahan peletakkan potongan genom. Untuk mengatasi masalah ini, program pada mesin pengurut genom dapat menghitung probabilitas error pada hasil pengurutan genom [8].

Apabila proses pengurutan ini tidak memenuhi batas standar error yang telah dihitung, maka genom akan dipecah dan diurutkan kembali. Sampai saat ini, penentuan validitas hasil urutan genom tidak dapat dilakukan oleh komputer [8]. Karena itu, dibutuhkan orang tertentu untuk melakukan finalisasi hasil penngurutan genom.

III. Proses Pengurutan Genom dengan Algoritma Brute Force

Seperti yang sudah dijelaskan pada subbab sebelumnya, genom adalah informasi genetik yang disusun dari empat basa nukelotida yaitu Adenin, Thymin, Sitosin, dan Guanin. Ada 3 milyar informasi yang terdapat di dalamnya, dan dalam subbab ini akan dibahas bagaimana cara mendapatkan kemungkinan-kemungkinan tersebut dengan menggunakan algoritma *brute force*. Ada dua pendekatan dalam mengurutkan genom, yaitu pendekatan tanpa graf dan pendekatan menggunakan graf.

A. Pendekatan tanpa Graf

Dalam pendekatan tanpa graf ini, program akan mengecek setiap potongan-potongan genom dan menyusun semua kemungkinan yang ada. Misalkan kita memiliki sebuah genom sederhana TAATGCCATGGGATGTT. Buatlah sebuah komposisi k-mer (k adalah sebuah bilangan asli yang menandakan panjang dari potongan genom) dengan k = 3 [9]. Hal pertama yang kita lakukan adalah memecah-mecah genom tersebut menjadi sebuah 3-mer (TAA, AAT, ATG, TGC, GCC, CCA, CAT, ATG, TGG, GGG, GGA, GAT, ATG, TGT, dan GTT).

Setelah kita mendapatkan semua 3-mer dari genom tersebut, kita bisa menyusunnya kembali menjadi satu string panjang yang merepresentasikan sebuah genom. Namun, perlu diperhatikan bahwa ketika kita mendapatkan kumpulan 3-mer tersebut, kita tidak mengetahui posisi awal mereka [9]. Atau dengan kata lain, kita tidak tahu harus diletakan di indeks ke berapa 3-mer tersebut.

Untuk mengatasi masalah ini, hal yang dapat dilakukan adalah mengurutkan 3-mer tersebut sesuai dengan aturan kamus [9]. Proses pengurutan k-mer ini dapat memanfaatkan algoritma *brute force*. Salah satu teknik pengurutan dengan algoritma *brute force* yang sering digunakan adalah *bubble sort* [5]. Proses pengurutan dengan *bubble sort* akan mengiterasi seluruh elemen pada himpunan elemen. Untuk setiap elemen yang bersebelahan jika elemen yang lebih akhir bernilai lebih kecil (asumsi urutan pengurutan adalah membesar), maka elemen tersebut ditukar dengan elemen setelahnya. Proses penukaran ini akan terus diulang sejumlah kuadrat dari jumlah elemen dalam himpunan elemen.

Setelah dirutkan sesuai dengan urutan kamus, maka algoritma akan dimulai dengan mengambil 3-mer pertama, dalam contoh kasus ini, AAT adalah 3-mer pertama. Dengan algoritma *brute force*, program akan mencari semua 3-mer yang memiliki prefix AT, misalnya ATG. Namun, karena ada tiga 3-mer yang memiliki prefix AT, kita bisa memilih salah satu saja. Demikian program ini akan terus berlanjut sampai seluruh genom berhasil diurutkan kembali.

Permasalahan kedua adalah pada contoh, kita memulai dengan AAT, padahal genom yang sebenarnya dimulai dengan TAA. Untuk mengatasi hal ini, algoritma *brute force* saja tidak akan menyelesaikan persoalan, karena pada kenyataannya, genom terdiri dari 3 milyar informasi. Karena itu, diperlukan sebuah DNA referensi yang sudah pernah diambil datanya untuk suatu percobaan [10]. Jika tidak mirip, maka program akan terus mencari sampai tingkat kemiripan yang diizinkan oleh para peneliti (memeriksa *error*).

Permasalahan yang ketiga adalah tidak ditemukan kelanjutan dari k-mer dalam proses pengurutan genom [9]. Untuk permasalahan ini, ada dua solusi yang dapat dijalankan, yang pertama adalah menambahkan algoritma *backtrack* untuk mengganti k-mer yang paling terakhir. Solusi kedua adalah dengan mengganti pendekatan kita dengan menggunakan pendekatan graf [9], yang akan dibahas setelah poin ini. Untuk solusi pertama, cara penyelesaiannya mirip dengan menyelesaikan *Sudoku*. Ketika tidak ditemukan lagi solusi yang valid, maka kita "mundur" dan mengganti solusi sebelumnya dengan solusi lain yang valid. Jika masih tidak ditemukan maka, program akan terus mundur. Dan mengganti solusi sampai ditemukan solusi.

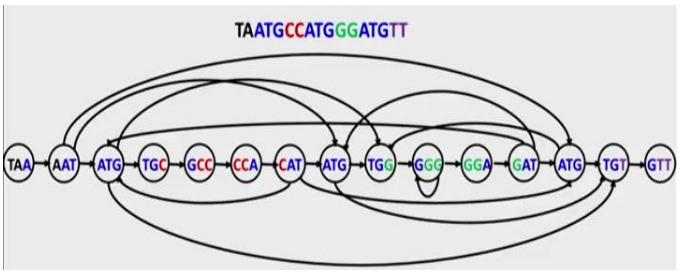
Pada praktiknya algoritma *backtrack* ini kurang disukai para peneliti, karena cukup memakan waktu dalam penyelesaiannya. Karena itu dipilihlah solusi kedua, yaitu solusi dengan menggunakan pendekatan graf untuk pengurutan genom.

B. Pendekatan dengan Graf

Dalam pendekatan dengan graf sendiri, ada dua cara yang dapat dijadikan solusi untuk mengurutkan suatu genom. Solusi pertama adalah dengan memetakan masalah pengurutan genom ke masalah lintasan Hamilton. Sedangkan solusi kedua adalah dengan memetakan masalah pengurutan genom ke masalah lintasan Euler [9]. Maka masalah pertama yang dilalui adalah bagaimana cara menggambarkan graf tersebut.

Pada solusi pertama, yaitu dengan memetakan masalah ke lintasan Hamilton, kita memosisikan setiap 3-mer sebagai

simpul dari suatu graf [9]. Dengan diposisikannya 3-mer sebagai simpul, kita bisa menghubungkan semua simpul yang berhubungan dengan konsep yang sama seperti pendekatan tanpa graf. Setiap simpul dihubungkan sedemikian rupa sehingga membentuk graf berarah yang merepresentasikan sebuah genom. Dalam pembentukan graf ini, kita tidak menggunakan graf tidak berarah karena belum tentu prefix dan suffix dari suatu 3-mer berhubungan bolak-balik. Misalnya AAT dan ATG. Kita bisa menghubungkan AAT ke ATG, tapi tidak sebaliknya.



Gambar 2 Pemodelan Genom Sebagai Graf dengan Kaidah Lintasan Hamilton

Sumber : https://class.coursera.org/assembly-001/lecture [9]

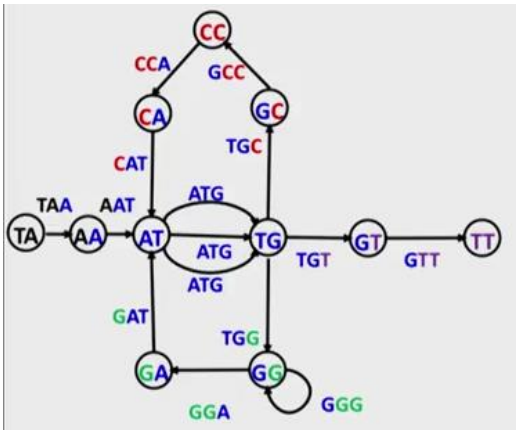
Untuk membentuk sisi-sisi graf, kita bisa menggunakan algoritma *brute force*. Secara umum, langkah algoritma *brute force* dalam menghubungkan setiap simpul yang ada adalah sebagai berikut : 1. Telusuri setiap sisi yang ada di dalam lingkup graf. 2. Jika ditemukan sisi yang valid (prefixnya sama dengan suffix simpul yang sedang dicek), maka hubungkan kedua siimpul. 3. Ulangi langkah 1 sampai tidak ada lagi simpul yang dapat dihubungkan. Dengan langkah tersebut, maka dipastikan graf yang merepresentasikan genom sudah terbentuk.

Setelah graf berhasil terbentuk, maka proses pengurutan genom yang dilakukan adalah menelusuri graf tersebut dan menentukan jalur yang tepat agar genom yang telah dipecah dapat dibentuk kembali. Seperti yang telah disebutkan, cara penelusuran graf ini mengikuti kaidah lintasan Hamilton, artinya kita mencari jalur yang melalui setiap simpulnya tepat satu kali saja.

Kemudian pada solusi kedua kita menggunakan kaidah lintasan Euler. Untuk membentuk graf yang demikian, maka yang perlu kita lakukan adalah merepresentasikan 3-mer sebagai "bobot" sisi. Karena 3-mer direpresentasikan sebagai sisi, maka simpul tidak lagi berisi 3-mer tersebut, melainkan berisi prefix dari 3-mer jika simpul tersebut merupakan simpul asal, dan berisi suffix dari 3-mer jika simpul merupakan simpul terminal [9]. Sebagai contoh kita memiliki dua simpul sebut saja v1 dan v2. Ada sebuah sisi yang menghubungkan kedua simpul tersebut bernilai AAT. Maka AA diisikan ke v1, sedangkan v2 diisikan dengan AT.

Proses pembentukan simpul-simpul ini juga dapat dilakukan dengan algoritma *brute force*, di mana kita membuat semua simpul yang mungkin dari k-mer yang telah disediakan. Setelah dibentuk setiap simpulnya, maka proses selanjutnya adalah menghubungkan simpul-simpul yang sudah dibuat sesuai dengan k-mer yang tersedia dengan tidak lupa memperhatikan arah dari sisi yang dibuat.

Tentunya akan ada simpul-simpul yang bernilai sama, namun bersisian dengan sisi yang berbeda. Untuk kasus yang demikian, kita bisa meleburkan simpul-simpul tersebut menjadi satu simpul saja [9]. Karena menjadi satu simpul, bukan berarti busur yang berbeda jadi menghilang. Busur yang berbeda tersebut akan tetap bersisian dengan simpul yang telah disatukan, dan akan tetap bersisian dengan simpul terminalnya. Agar lebih jelas, dapat dilihat pada gambar 3 berikut.



Gambar 3 Pemodelan Genom Sebagai Graf dengan kaidah Lintasan Euler

Sumber : : https://class.coursera.org/assembly-001/lecture [9]

Graf yang telah merepresentasikan sebuah genom adalah graf yang setiap simpulnya tidak memiliki duplikat lagi. Graf yang demikian disebut dengan graf De Bruijn [9]. Untuk menelusuri graf De Bruijn, kita gunakan kaidah lintasan Euler seperti yang telah disebutkan di atas. Untuk mengurutkan sebuah genom, kita perlu menelusuri graf De Bruijn, dengan mencari jalur yang melalui setiap sisinya tepat satu kali saja.

Untuk menelusuri kedua graf yang telah dibentuk, baik melalui kaidah lintasan Hamilton maupun melalui kaidah lintasan Euler, kita dapat menggunakan algoritma *brute force*. Algoritma *brute force* dalam penelusuran graf ini mirip dengan proses penyelesaian persoalan *Travelling Salesman Problem (*TSP). Dengan metode *exhaustive search* [5], maka langkah-langkah yang dilakukan adalah mengenumerasi semua lintasan Hamilton/Euler dari graf representasi genom. Kemudian kita mengevaluasi apakah ada langkah yang valid atau tidak. Tahap ini bisa dilewati karena seharusnya selalu ada langkah yang valid, karena graf dibentuk dari genom utuh yang dipecah. Setelah menemukan setiap langkah yang valid, maka kita mengevaluasi manakah genom yang memiliki tingkat *error* paling sedikit.

Namun demikian, algoritma *brute force* kini tidak digunakan lagi untuk menyelesaikan permasalahan graf De Bruijn [9]. Hal ini dikarenakan untuk graf dengan permasalahan lintasan Euler telah ditemukan algoritma yang lebih efisien dari algoritma *brute force*. Berbeda dengan lintasan Hamilton, yang belum ditemukan algoritma yang lebih baik dari *brute force* untuk menemukan solusinya. Karena permasalahan inilah, pengurutan genom saat ini selalu menggunakan graf De Bruijn, yang berarti menggunakan kaidah lintasan Euler.

IV. Kelebihan Dan Kelemahan Algoritma Brute Force dalam Pengurutan Genom

Berdasarkan pembahasan mengenai penerapan algoritma *brute force* pada graf di subbab sebelumnya, maka dapat ditemukan kelebihan dan kekurangan dari algoritma *brute force* dalam proses pengurutan genom.

A. Kelebihan Algoritma Brute Force

Algoritma *brute force* adalah algoritma yang mencoba setiap kemungkinan yang ada untuk menemukan solusi, maka salah satu kelebihan dari algoritma *brute force*, terutama dalam proses pengurutan genom ini adalah pasti menghasilkan solusi yang valid [5]. Solusi yang valid dalam hal ini artinya, algoritma *brute force* dapat menemukan jalur dari lintasan Hamilton/Euler dari graf genom. Sesulit atau sebanyak apapun simpul dari graf genom, algoritma *brute force* pasti dapat menemukan solusi yang valid dari persoalan pengurutan genom ini.

Kelebihan lain dari algoritma *brute force* adalah implementasinya yang sederhana [5]. Telah diuraikan di subbab sebelumnya bahwa algoritma untuk menelusuri graf sangatlah pendek dibandingkan dengan algoritma lainnya. Dengan demikian, secara implementasi, algoritma *brute force* adalah solusi yang paling mudah, karena tidak diperlukan komputasi yang rumit dalam setiap langkahnya.

Kelebihan ini dapat dimanfaatkan untuk para pemula di bidang bioinformatika atau di bidang informatika sendiri. Pemanfaatan algoritma *brute force* bagi para pemula biasanya diterapkan untuk pembelajaran. Kesederhanaan algoritma *brute force* menyebabkan algoritma ini relatif mudah dipahami untuk para pemula, terutama dalam persoalan graf. Sampai saat ini, sebagian besar persoalan graf masih menjadi persoalan yang sulit dipecahkan. Dengan memahami algoritma *brute force* ini, maka masalah efisiensi waktu akan menjadi masalah berikutnya bagi para peneliti di proses pengurutan genom ini.

Kelebihan yang ketiga adalah algoritma *brute force* merupakan algoritma yang layak untuk beberapa persoalan [5], dalam kasus ini adalah lintasan dan sirkuit Hamilton. Seperti yang telah diuraikan di atas, persoalan lintasan dan sirkuit Hamilton masih merupakan persoalan yang sulit. Karena belum ditemukan algoritma yang lebih cepat dari algoritma *brute force*, maka bisa dikatakan bahwa algoritma *brute force* adalah satu-satunya jalan untuk menyelesaikan persoalan lintasan dan sirkuit Hamilton.

Selain persoalan Hamilton, algoritma *brute force* juga dapat diterapkan dalam proses pembentukan graf. Dalam hal ini algoritma *brute force* juga menghasilkan solusi yang benar, sehingga algoritma *brute force* merupakan algoritma yang layak untuk membentuk sebuah graf, baik graf Hamilton atau graf De Bruijn.

Karena kepastiannya dalam menemukan solusi, terutama dalam permasalahan graf Hamilton, algoritma *brute force* bisa dijadikan standar algoritma untuk menyelesaikan permasalahan graf Hamilton ini [5]. Yang dimaksud standar adalah algoritma *brute force* merupakan algoritma tercepat untuk menyelesaikan persoalan graf Hamilton, sehingga setiap pemrogram harus membuat algoritma yang kecepatannya lebih besar atau sama dengan algoritma *brute force* ini.

Namun demikian, algoritma *brute force* memiliki kelemahan utama, yang menyebabkan algoritma *brute force* kurang disukai para peneliti dalam bidang bioinformatika dan informatika. Kelemahan-kelemahan tersebut akan dibahas di poin berikutnya.

B. Kelemahan Algoritma Brute Force

Salah satu kekurangan dari algoritma *brute force* yang paling sering ditemui adalah permasalahan efisiensi waktu dan ruang [5]. Dalam bidang pemrograman, metode *brute force* jarang digunakan karena membutuhkan waktu yang lama dalam penyelesaian masalahnya. Misalnya untuk penyelesaian masalah dengan graf Hamilton ini. Graf Hamilton pada proses pengurutan genom bisa jadi memiliki jutaan simpul mengingat informasi yang tersimpan dalam genom sekitar 3 milyar informasi genetik. Bayangkan berapa waktu yang dibutuhkan untuk mengenumerasi jalur yang tepat dari graf Hamilton yang memiliki jutaan simpul ini. Jika kita asumsikan satu pembuatan satu simpul membutuhkan waktu 1 detik, maka kita memerlukan 1 juta detik (sekitar 278 hari) untuk membentuk graf Hamilton. Dan belum lagi kita harus menelusuri graf Hamilton tersebut.

Karena itu, dalam proses pengurutan genom ini graf Hamilton tidak dipilih sebagai solusi mengingat waktu eksekusi untuk graf Hamilton ini sangatlah lama. Meskipun algoritma *brute force* adalah algoritma yang valid dalam menghasilkan solusi untuk permasalahan graf Hamilton, tetapi algoritma *brute force* juga dianggap tidak valid karena tidak menyelesaikan masalah dalam waktu yang cepat [5].

Selain mengenai masalah kompleksitas ruang dan waktu, algoritma *brute force* memiliki kelemahan dari segi kreativitas dalam teknik penyelesaian masalah pengurutan genom ini [5]. Kreativitas ini juga kembali menyangkut ke masalah kompleksitas waktu dan ruang tadi. Saat ini peneliti di bidang informatika sedang berlomba-lomba untuk mencari algoritma yang efisien baik dari segi waktu maupun ruang. Jika seorang pemrogram hanya menguasai penyelesaian masalah dengan algoritma *brute force* saja, atau dengan kata lain masalah-masalah yang kecil, maka pemrogram tidak akan maju dan menjadi pemrogram yang baik.

Kemudian kelemahan ketiga adalah kelemahan dari segi kebenaran penyusunan genom. Dengan menggunakan algoritma *brute force*, ternyata masih dapat terjadi kesalahan dalam proses pengurutan genom [9]. Pada tingkat yang lebih lanjut dalam pengurutan genom, program untuk proses pengurutan genom ini memerlukan perhitungan matematis dalam menentukan protein yang akan diurutkan nantinya. Hanya dengan algoritma *brute force*, tanpa pengecekan di setiap langkahnya, pengurutan protein ini bisa menghasilkan solusi yang salah [9].

Solusi yang dimaksud mirip dengan isitilah isotop pada kimia, yaitu dua unsur berbeda yang memiliki jumlah proton yang sama [11]. Pada biologi juga terdapat protein dengan massa tertentu. Tanpa pengecekan di setiap langkahnya, maka

urutan protein yang dihasilkan bisa jadi berbeda dari yang diinginkan karena program hanya mengecek total massanya saja.

Hal ini tentunya tidak diharapkan oleh para peneliti di HGP. Tentunya hasil yang diinginkan adalah hasil yang sesempurna mungkin untuk keperluan studi. Dengan demikian kelemahan algoritma *brute force* yang lain dalam bidang ini adalah ketidakmampuannya untuk menyelesaikan satu masalah tanpa bergantung pada algoritma yang lain. Dengan kata lain, algoritma *brute force* harus mengalami modifikasi yang cukup besar, sehingga malah menghilangkan kesederhanaan algoritma *brute force* tersebut.

C. Solusi dari Kelemahan Algoritma Brute Force

Hingga saat ini, para peneliti terus berusaha untuk mencari algoritma baru untuk menggantikan *brute force* dalam HGP ini. Dan sebagian besar algoritma tersebut telah ditemukan sehingga waktu untuk menyelesaikan HGP ini jauh lebih cepat dibandingkan dengan menggunakan algoritma *brute force* saja.

Salah satu contoh algoritma berkaitan dengan HGP yang sudah dibuktikan lebih cepat dari *brute force* adalah algoritma Knuth-Morris-Pratt (KMP) dan Booyer-Moore (BM) dalam proses pencocokan string [12]. Pencocokan string dalam proses pengurutan genom terjadi pada proses pencocokkan prefix dan suffix. Kemudian algoritma *sorting* dalam HGP juga bisa menggunakan *insertion sort, quick sort, heap sort*, dll yang jauh lebih cepat dibandingkan *bubble sort* [12].

Pada beberapa masalah graf di HGP, digunakan juga algoritma *branch and bound*, yaitu dalam proses pengurutan protein yang merupakan kelemahan dari algoritma *brute force*, seperti yang telah dijelaskan di atas [9]. Algoritma *branch and bound* ini akan memastikan protein yang sesuai genom aslinya agar tidak terjadi kesalahan dalam diagnosis fungsional protein.

Masih banyak lagi algoritma-algoritma lain yang dapat menggantikan *brute force* untuk menyelesaikan HGP ini, terutama dalam permasalahan graf, kecuali untuk masalah graf Hamilton. Karena tidak ada algoritma yang efisien, maka para peneliti HGP tidak menggunakan graf Hamilton, melainkan graf De Bruijn.

V. KESIMPULAN

Permasalahan-permasalahan dalam HGP adalah permasalahan-permasalahan yang dapat diselesaikan dengan algoritma *brute force*. Algoritma *brute force* merupakan algoritma yang paling tepat untuk menyelesaikan masalah HGP karena kepastiannya dalam menyelesaikan sebagian besar masalah dalam HGP. Namun demikian, algoritma *brute force* memiliki kelemahan utama, yaitu waktu eksekusi yang lama. Waktu eksekusi yang lama ini dapat berakibat pada biaya penelitian yang makin besar, karena itu algoritma *brute force* mulai ditinggalkan dalam penyelesaian masalah di HGP. Untuk menggantikan algoritma *brute force*, para peneliti telah menggantikan algoritma penyelesaian beberapa masalah dalam HGP dengan algoritma lain, seperti KMP dan BM dalam pencocokan string, *insertion sort* pada proses pengurutan k-mer, dan algoritma *branch and bound* dalam proses pengurutan protein yang kompleks.

REFERENSI

[1] BioPlanet. 2013. "*What is Bioinformatics"*. Diakses dari http://www.bioplanet.com/what-is-bioinformatics/ pada 6 Mei 2016 pukul 17.43.

[2] Broad Institute. 2016. "*Genome*". Diakses dari http://www.broadinstitute.org/education/glossary/genome pada 6 Mei 2016 pukul 21.04.

[3] National Human Genome Research Institute. 2015. *"All About The Human Genome Project (HGP)*". Diakses dari https://www.genome.gov/10001772/all-about-the--human-genome-project-hgp/ pada 6 Mei 2016 pukul 20.49.

[4] Yourgenome.org. 2014. *"What is a Genome"*. Diakses dari http://www.yourgenome.org/facts/what-is-a-genome pada 6 Mei 2016 pukul 20.17.

[5] Munir, Rinaldi. 2009. *Diktat Kuliah IF2211 Strategi Algoritma*. Institut Teknologi Bandung : Bandung.

[6] Munir, Rinaldi. *Diktat Kuliah IF2120 Matematika Diskrit*. Institut Teknologi Bandung : Bandung.

[7] Rosen, Kenneth H. 2012. *Discrete Mathematics and Its Applications.* McGraw-Hill : New York.

[8] Genome News Network. 2003. *"Genome Sequencing*". Diakses dari http://www.genomenewsnetwork.org/resources/whats_a_genome/Chp2_1.shtml pada 6 Mei 2016 pukul 23.26.

[9] Compeau, Philip. 2014. "*Genome Sequencing (Bioinformatics II*". Diakses dari https://class.coursera.org/assembly-001/lecture pada 6 Mei 2016 pukul 17.32.

[10] 454 Life Science. Tanpa Tahun. "*How is Genome Sequencing Done?*". Diakses dari www.454.com pada 7 Mei 2016 pukul 17.21.

[11] Dictionary.com. 2016. "*IsotopI"*. Diakses dari http://www.dictionary.com/browse/isotope pada 7 Mei 2016 pukul 21.32.

[12] Kasahara, Masahiro, Shinichi Morishita. 2006. *Large Scale Genome Sequence Processing.* Imperial College Press : London.

VI. PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 7 Mei 2016

ttd

Martino Christanto Khuangga - 13514084