Penggunaan Algoritma Penelusuran Graf dalam Persoalan Rekonstruksi String untuk Penyusunan Genom

Kevin Rizki Mohammad - 13518100
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13518100@std.stei.itb.ac.id

***Abstrak*—Makalah ini mempelajari penerapan algoritma penelusuran graf pada rekonstruksi string. Dengan menerima masukan sejumlah upastring dalam multiset komposisi dengan panjang setiap upastring sama, tujuan dari persoalan adalah merekonstruksi ulang string awal. Tujuan dari penelitian ini adalah melihat seberapa jauh batasan algoritma penelusuran graf Depth First Search, dalam menyelesaikan permasalahan rekonstruksi string. Kita akan tunjukkan bahwa persoalan rekonstruksi string dapat didekati dengan algoritma yang menyelesaikan persoalan pencarian lintasan Euler pada graf berarah. Didapatkan ternyata algoritma pencarian lintasan Euler yang kita susun ternyata tidak tuntas dalam menyelesaikan persoalan rekonstruksi string. Terdapat beberapa batasan yang bergantung dari seperti apa panjang upastring yang diterima.**

Kata kunci—penyusunan genom; rekonstruksi string; graf de Bruijn; depth-first search; lintasan Euler

I. PENDAHULUAN

Teks adalah salah satu jenis informasi yang biasa digunakan dalam berbagai kebutuhan penyimpanan, penggunaan, dan pengiriman informasi. Pada beberapa kasus, informasi terkadang tidak bisa didapatkan sebagai satu kesatuan utuh melainkan didapatkan dalam kepingan-kepingannya saja. Sebagai contoh bayangkan terdapat tumpukan beberapa koran yang serupa yaitu setiap koran adalah salinan koran yang lain. Kemudian koran itu dirobek hingga menjadi sobekan-sobekan kecil yang banyak. Tantangannya adalah dengan semua sobekan koran tersebut bisakah kita mendapatkan informasi yang semula ada pada koran tersebut.

Permasalahan tentang sobekan koran ini serupa dengan permasalahan yang ada pada genetika yaitu pada penyusunan genom materi genetik yang ada pada setiap makhluk hidup. Dalam membaca material genetik berupa barisan nukleotida, ahli genetik belum menemukan teknik untuk membaca seluruh barisan tersebut secara utuh seperti halnya membaca teks sebuah koran, melainkan mereka menggunakan teknik untuk membaca barisan-barisan pendek dengan ukuran yang jauh lebih pendek, yang disebut dengan istilah *reads*. Sehingga diperlukan suatu teknik komputasi untuk mendapatkan barisan genom yang asli diberikan barisan-barisan pendek hasil yang didapat dari barisan asli tersebut.

Makalah ini akan berusaha mempelajari sifat dari persoalan rekonstruksi string, dimana terdapat informasi multiset upastring komposisi sebuah teks dengan semua panjang upastring adalah sama. Tujuan dari persoalan tersebut adalah merekonstruksi ulang string teks dari hanya informasi komposisi nya saja.

Secara matematis untuk string w dengan panjang n dan sebuah bilangan bulat positif k, maka komposisi k-mer dari w, dinotasikan dengan $S_k(w)$, adalah multiset dari semua upastring w dengan panjang k, yaitu $S_k(w) = \{w_{1,k}, w_{2,k}, ..., w_{n-k+1,k}\}$, dimana $w_{i,k}$ adalah upastring $(w_i, w_{i+1}, ..., w_{i+k-1})$. Tujuan dari algoritma penyelesaian rekonstruksi string adalah mendapatkan kembali string w diberikan $S_k(w)$ dan bilangan bulat k. String W adalah semua string yang bisa direkonstruksi dari $S_k(w)$. $|W|$ dapat bernilai 0 hingga $|S_k(w)|$ faktorial. Sebagai batasan dalam makalah kali ini, kita tidak melakukan rekonstruksi untuk semua string pada W melainkan hanya bisa mendapatkan sebuah solusi saja.

Dalam menyusun persoalan rekonstruksi string untuk penyusunan genom, membuat beberapa asumsi tentang seperti apa multiset upastring yang kita miliki. Kita mengasumsikan setiap upastring memiliki panjang yang sama sehingga kita bisa menulis secara matematis tentang multiset komposisi k-mer nya. Lalu diasumsikan juga upastring yang terdapat dalam setiap kasus disini dihasilkan dari pembacaan *reads* yang tidak mengandung kesalahan. Kesalahan ini mungkin terjadi sebagaimana cara pembacaan yang sekarang mampu dilakukan dalam genetika masih belum sempurna. Ketiga, kita mengasumsikan semua upastring k-mer adalah meliputi semua upastring yang dihasilkan ketika suatu genom dikomposisikan. Artinya tidak ada upastring dari genom yang hilang.

Sudah banyak makalah yang membahas persoalan rekonstruksi string dari berbagai sudut pandang, seperti dilakukan oleh Marcovich dan Yaakobi pada [7]. Dari yang teoritis hingga membahas batasan dari penyelesaian masalah ini. Tujuan dari studi makalah kali ini adalah untuk melihat batasan algoritma penelusuran graf dasar Depth-First search pada rekonstruksi string teks genom. Diketahui ternyata,

menyelesaikan persoalan rekonstruksi string pada dasarnya sama dengan penyelesaian persoalan mencari lintasan Euler pada graf. Karena itu solusi yang akan disusun untuk menyelesaikan persoalan pada makalah ini akan menuangkan algoritma DFS pada persoalan pencarian lintasan Euler.

Makalah ini akan menyusun persoalan mulai dari definisi persoalan rekonstruksi string, pembentukan graf *de Bruijn* dari suatu multiset komposisi *k*-mer, dan bagaimana pendekatan algoritma DFS pada pencarian lintasan Euler dapat merekonstruksi string diberikan multiset upastring *k*-mer.

II. Dasar Teori

A. Persoalan Rekonstruksi String

Pada makalah kali ini akan digunakan alfabet untuk genom. Alfabet genom A adalah alfabet yang terdapat pada barisan nukleotida pada genom, dengan A = {"a", "c", "g", "t"}. Sebuah string $w \in A^n$ adalah string dengan simbol-simbolnya adalah anggota A dengan panjang bilangan bulat n.

Komposisi *k-mer* sebuah string w adalah multiset semua k-mer upastring dari w. Untuk mendapatkan multiset $S_k(w)$ dari string w dan bilangan bulat k dapat digunakan algoritma naif sebagai berikut.

Algoritma 1 Komposisi(w, k)
Input: Sebuah string w dengan ukuran n dan bilangan bulat k.
Output: Multiset $S_k(w)$.
1: set $S_k(w) = \{\}$
2: **for** semua $i = 0, \ldots, |w| + 1 - k$
3: $S_k(w) = S_k(w) \cup w_{i,k}$
4: **return** $S_k(w)$

Persoalan rekonstruksi string adalah balikan dari persoalan mencari komposisi k-mer dari string w. Definisi dari persoalannya secara komputasi adalah sebagai berikut.

Persoalan 1 Rekonstruksi String:
Merekonstruksi string w dari komposisi k-mer nya.
Input: Bilangan bulat k dan multiset S_k.
Output: String w yang komposisi k-mer nya adalah S_k.

Dalam persoalan rekonstruksi string ini, jumlah string w yang dihasilkan dapat bernilai 0. Itu berarti tidak ada string yang jika dikomposisikan akan mendapatkan S_k. Namun supaya kita bisa menguji perilaku algoritma yang akan kita susun, maka kita akan membuat masukan multiset S_k berasal dari algoritma 1. Sehingga paling tidak terdapat minimal satu buah solusi. Kita menambahkan parameter keberhasilan algoritma untuk persoalan 1 ialah harus memberikan string yang sama dengan masukan kasus uji.

B. Teori Graf dan Lintasan Euler

Graf adalah himpunan simpul dan sisi. Sebuah graf $G = (V, E)$ pada makalah ini akan direpresentasikan dengan senarai ketetanggan. Graf yang akan dibahas pada makalah kali ini adalah graf berarah (*directed graph*).

Lintasan Euler adalah lintasan yang melalui masing-masing sisi pada graf tepat satu kali. Suatu graf berarah dapat memiliki lintasan Euler atau tidak berdasarkan teorema 1.

Teorema 1. *Graf berarah G memiliki lintasan Euler jika dan hanya jika G terhubung dan paling banyak sebuah simpul dengan derajat masuk dikurang derajat keluar sama dengan 1 dan paling banyak sebuah simpul dengan derajat keluar dikurang derajat masuk sama dengan 1 dan semua simpul yang lain memiliki jumlah derajat masuk dan derajat keluar yang sama, atau graf G terhubung dan semua simpul memiliki jumlah derajat masuk dan derajat keluar yang sama.*

C. Graf de Bruijn

Sebagai pendahulu, string w dengan panjang n, kita akan menyebut *x-prefix* dari w adalah x buah karakter terdepan dari w, dan *x-suffix* dari w adalah x buah karakter paling belakang dari w. Dalam hal ini maka *x-prefix* dan *y-prefix* adalah anggota multiset $S_k(w)$ dengan $x = k - 1$.

Pada pembahasan selanjutnya, kita akan menyatukan persoalan rekonstruksi string dengan teori graf. Disini kita akan menggunakan graf de Bruijn berdasarkan [9].

Definisi 1. *Graf de Bruijn dari S_k adalah sebuah graf dengan total sisi sebanyak $|S_k|$ dimana setiap sisi merepresentasikan sebuah upastring k-mer dari S_k, dan simpul asal dan simpul tujuan akan diberi label berdasarkan berturut-turut k-1 prefix dan k-1 suffix dari upastring tersebut.*

Dengan asumsi bahwa pada S_k adalah hasil komposisi sebuah string tanpa ada upastring yang hilang, maka kita bisa menyusun teorema 2.

Teorema 2. *Untuk setiap graf de Bruijn dari S_k, hasil komposisi sebuah string w, dipastikan pada graf tersebut terdapat paling tidak sebuah lintasan Euler.*

D. Depth First Search

Secara umum, algoritma Depth First Search (DFS) adalah salah satu algoritma traversal pada graf. Seperti namanya, algoritma ini akan menelusuri graf sedalam mungkin hingga tidak ada simpul lagi yang bisa ditelusuri lebih dalam. DFS akan menelusuri simpul yang pertama kali ditemukan dan belum pernah ditelusuri. Untuk mencakup semua simpul yang ada pada graf, DFS akan melakukan runut balik ke simpul sebelumnya untuk melakukan penelusuran secara mendalam lagi. Penelusuran seperti ini terus dilakukan hingga semua

simpul yang terjangkau dari simpul asal telah tertelusuri. Ketika semua simpul dari asal telah tertelusuri dan masih terdapat simpul yang belum ditelusuri (karena tidak terjangkau) dari simpul asal, maka DFS akan memilih simpul asal baru dari simpul-simpul yang belum ditelusuri dan memulai penelusuran dari simpul tersebut. Algoritma akan selesai ketika semua simpul pada graf telah ditelusuri.

Algoritma global DFS untuk penelusuran graf adalah sebagai berikut. Dengan diberikan masukan *G* yaitu graf.

Algoritma 2.1 DFS(G)

Input: Graf *G*
Output: Urutan semua simpul yang dikunjungi
1: **for** semua vertex *u* pada *G*:
2: *visited*[*u*] = false
3: **for** semua vertex *u* pada *G*:
4: **if** not *visited*[*u*]
5: DFS-Util(*G, u, visited*)

Algoritma 2.2 DFS-Util(*G, u, visited*)

1: **write**(*u*)
2: *visited*[*u*] = true
3: **for** semua v ∈ G.adj(*u*):
4: **if** not *visited*[*v*]:
5: DFS-Util(*G, v, visited*)

Prosedur DFS pada ilustrasi di atas adalah sebagai berikut. Baris 1-2 menginisiasi larik *visited* untuk menandai semua simpul pada *G* belum dikunjungi. Baris 3-4 adalah menelusuri semua simpul pada *G*. Jika ada simpul yang belum pernah ditelusuri maka simpul tersebut akan menjadi simpul sumber untuk kemudian dilakukan prosedur DFS-Util pada baris 5.

Setiap simpul yang masuk ke DFS-Util akan ditandai telah ditelusuri untuk kemudian diiterasi semua simpul tetangganya. Jika ada simpul tetangga yang belum pernah dikunjungi, maka simpul tersebut akan ditelusuri tersebut hingga selesai. Proses runut balik adalah ketika semua pada baris 3-5 di DFS-Util semua simpul telah ditelusuri.

E. Persoalan Lintasan Euler

Diberikan graf berarah *G* yang memenuhi Teorema 1. Persoalan mencari lintasan Euler didefinisikan sebagai berikut.

Persoalan Lintasan Euler:
Mendapatkan lintasan dari simpul awal *v* ke suatu simpul tujuan *v* yang melalui setiap sisi pada *G* tepat satu kali.

Input: Graf berarah *G*.
Output: Sebuah urutan simpul-simpul yang adalah jika ditelusuri akan melewati setiap sisi pada *G* tepat satu kali.

III. Pendekatan Depth First Search Untuk Persoalan Rekonstruksi String

Pada bagian ini, kita akan mencoba memformulasikan algoritma DFS untuk menyelesaikan persoalan lintasan Euler untuk rekonstruksi string. Sebelum itu kita akan menyusun algoritma untuk membentuk graf de Bruijn dari multiset komposisi *k*-mer S_k yang diberikan. Setelah itu kita akan menggunakan algoritma DFS untuk menyelesaikan persoalan lintasan Euler. Terakhir lintasan Euler pada graf de Bruijn, kita akan merekonstruksi string sesuai definisi persoalan 1.

A. Menyusun Graf de Bruijn dari Multiset Komposisi K-mer

Algoritma untuk mendapatkan graf de Bruijn dari multiset komposisi *k*-mer adalah sebagai berikut.

Algoritma 4 De-Bruijn(S_k)

Input: S_k
Output: Graf de Bruijn *G* yang merepresentasikan S_k
1: Inisiasi *G* sebagai graf
2: **for** semua *upastring* pada S_k
3: tambahkan sisi "(*k*-1 prefix *upastring*, *k*-1 suffix *upastring*)" dengan label *upastring* pada graf.
4: **return** *G*

Ilustrasi dari pembentukan graf de Bruijn adalah sebagai berikut.

Gambar.1. Ilustrasi pembentukan graf de Bruijn dari multiset komposisi dengan *k*=3.

B. Menyusun Ulang String Awal Jika Diketahui Lintasan yang Ada Dari Graf de Bruijn

Sebelumnya kita telah membuat graf de Bruijn setelah mendapat masukan multiset komposisi *k*-mer. Sifat dari graf de Bruijn, dimana setiap sisi menyatakan upastring *k-mer,* memberikan informasi tentang seperti apa string yang bisa dibentuk dari multiset. Pembentukan ulang string ini menggunakan algoritma pemetaan dari lintasan pada graf de Bruijn ke string.

Algoritma 5 Lintasan-Ke-String(*L*)

Input: *L* adalah lintasan simpul-simpul pada graf de Bruijn
Output: String yang terbentuk dari lintasan
1: Inisiasi *Text* sebagai string kosong
2: ***k*** **=** panjang label pada simpul + 1
3: *Text* = L_0
4: **for** semua simpul pada *L*
5: *Text* = **concat**(*Text, 1 suffix pada label di simpul*)
6: **return** Text

C. *Algoritma DFS untuk Menyelesaikan Persoalan Lintasan Euler*

Untuk menyelesaikan persoalan lintasan Euler dengan algoritma DFS, kita harus memodifikasi algoritma umum DFS pada algoritma 2. Jika algoritma DFS umum menelusuri setiap simpul yang belum dikunjungi, algoritma DFS untuk lintasan Euler akan mengunjungi semua simpul sampai ditemukan simpul yang sisi keluarnya sudah dikunjungi semua. Dengan kata lain, DFS akan menelusuri sisi-sisi yang belum ditelusuri. Pada setiap simpul tetangga *j* yang dikunjungi dari simpul yang sedang ditelusuri *i*, DFS akan memberikan tanda sudah ditelusuri pada sisi dengan label (*i*, *j*). DFS akan melakukan runut balik pada suatu simpul hanya jika semua simpul keluar yang berasal dari simpul tersebut telah dikunjungi.

D. *Algoritma DFS untuk Mencari Lintasan Euler pada Graf de Bruijn*

Dengan prekondisi bahwa masukkan graf *G* pasti memenuhi teorema 1, algoritma pencarian lintasan Euler menggunakan DFS untuk menelusuri setiap sisi pada *G*.

Algoritma 6.1 Eulerian-Path(*G*)

Input: Graf berarah yang memenuhi Teorema 1.
Output: Lintasan Euler *L*
1: Inisiasi *L* sebagai senarai kosong.
2: **for** semua edge *e* pada *G*:
3: visited[*e*] = false
4: *start* = vertice pertama pada Q
5: **for** semua simpul *v* pada G:
6: **if** jumlah sisi masuk pada *v* == 0:
7: *start* = *v*
8: **break**
9: **if** jumlah sisi keluar – sisi masuk pada *v* == 1:
10: *start* = *v*
11: **break**
12: DFS-Euler-Path(*start, G, L, visited*):
13: **return** *L*

Algoritma 6.2 DFS-Euler-Path(*start, G, L, visited*)

1: **for** semua v ∈ G.adj(*start*):
2: **if not** *visited*[(*start, v*)]:
3: *visited*[(*start, v*)] = True
4: DFS-Euler-Path(*v, G, L, visited*)
5: **insert** *start* ke *L* dari depan.

Pada algoritma DFS yang disesuaikan untuk pencarian lintasan Euler ini, kita memodifikasi bagaimana kita memiliki simpul untuk memulai pencarian. Untuk kemungkinan bahwa graf *G* memenuhi teorema 1 dengan ada simpul yang derajat masuk dan derajat keluarnya nya berbeda, kita mencari simpul pada graf dimana derajat sisi keluarnya lebih besar dibanding derajat sisi masuknya atau pada simpul yang tidak memiliki derajat masuk. Pencarian simpul itu terdapat pada baris 4-11 algoritma 6.1. Perbedaan berikutnya adalah dalam mencari simpul berikutnya yang akan dikunjungi, DFS-Euler-Path mencari sisi keluar yang belum dikunjungi, bukan simpul yang belum dikunjungi. Langkahnya terdapat pada baris 1-4 algoritma 6.2. Itu kemungkinan DFS kembali ke simpul yang sama berbeda dengan algoritma umum DFS.

E. *Rekonstruksi String*

Permasalahan rekonstruksi string mengambil pendekatan penelusuran graf dengan membentuk graf de Bruijn yang merepresentasikan multiset S_k. Setelah itu untuk menyusun kembali string, kita harus mendapatkan lintasan Euler pada graf de Bruijn. Hal ini supaya string yang dibentuk tersusun dari semua upastring dari S_k. Karena itu kita menerapkan algoritma DFS untuk pencarian lintasan Euler untuk mendapat lintasan yang lintasan tersebut apabila disusun menjadi string, akan menghasilkan hasil rekonstruksi string.

Algoritma 7 Rekonstruksi-String(*S, k*)

Input: Bilangan bulat *k* dan multiset S_k.
Output: String *w* yang komposisi *k*-mer nya adalah S_k.
1: *G* = De-Bruijn(S_k)
2: *L* = Eulerian-Path(*G*)
3: *w* = Lintasan-Ke-String(*L*)
6: **return *w***

Algoritma 7 pada intinya menyatukan algoritma pembentukan graf de-Bruijn pada algoritma 4, pencarian lintasan euler dari algoritma 6, dan pembentukan string dari lintasan pada algoritma 5. String *w* pada algoritma 7 adalah string yang apabila dikomposisikan dengan algoritma 1 dan nilai *k* menghasilkan multiset komposisi *k*-mer S_k.

IV. Rekonstruksi String untuk Penyusunan genom

Pada bagian ini, kita akan mencoba algoritma rekonstruksi string yang sudah dibuat sebelumnya dengan berbagai data genom dan hasil *k*-mer komposisinya. Untuk menguji, kita akan melihat apakah algoritma rekonstruksi string mampu memberikan string genom yang sama dengan genom sebelum dikomposisikan.

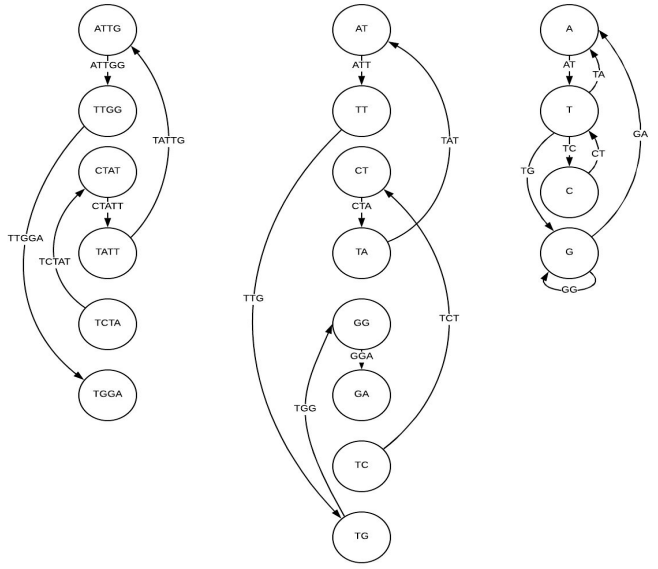
A. *Uji Coba Algoritma DFS untuk Mencari Lintasan Euler pada Beberapa Kasus*

Tabell I. Ujicoba Algoritma

Uji Coba	*N*	k	*Cocok (ya/tidak/ galgal)*
Uji Coba 1	9	5	Ya
Uji Coba 2	9	3	Ya
Uji Coba 3	9	2	Tidak
Uji Coba 4	20	10	Ya
Uji Coba 5	20	6	Ya
Uji Coba 6	20	3	Tidak

Uji Coba 7	50	25	Ya
Uji Coba 8	50	17	Ya
Uji Coba 9	50	10	Ya
Uji Coba 10	50	7	Ya
Uji Coba 11	50	5	Tidak
Uji Coba 12	50	3	Tidak
Uji Coba 13	100	50	Ya
Uji Coba 14	100	33	Ya
Uji Coba 15	100	10	Ya
Uji Coba 16	100	5	Tidak
Uji Coba 17	100	3	Tidak
Uji Coba 18	500	100	Ya
Uji Coba 19	500	10	Ya
Uji Coba 20	500	5	Tidak

Dalam uji coba yang dapat dilihat pada Tabel I bahwa sejumlah masukkan *k*-mer hasil komposisi string sepanjang *n* tidak memberikan keluaran rekonstruksi string yang sesuai. Hal ini akan terlihat jika kita coba merepresentasikan graf de Bruijn dari masukkan beberapa nilai buah *k*.



Gambar.2. Perbandingan graf de Bruijn. Multiset komposisi *k*—mer adalah untuk (dari kiri ke kanan) *k*=9, *k=3*, dan *k=2*. Didapatkan bahwa graf de Bruijn untuk *k*=2 terdapat lebih dari satu lintasan Euler.

Algoritma yang kita terapkan untuk menyelesaikan persoalan lintasan Euler tampaknya hanya mencoba mencari satu lintasan pertama yang ditemukan. Algoritma tidak berusaha mencari semua kemungkinan lintasan Euler yang mungkin berada pada graf.

B. Uji Coba untuk Merekonstruksi String dengan Ukuran String yang Sangat Panjang

Kita mendefinisikan panjang string yang sangat panjang sebagai batas panjang minimal algoritma ini tidak menghasilkan solusi. Penyelesaian masalah dengan algoritma DFS memberikan risiko tidak menghasilkan solusi mengingat sifat rekursif dari DFS dapat memberikan kompleksitas ruang yang besar. Tabel 2 adalah uji coba algoritma terhadap beberapa ukuran string.

TABELL II. UJICOBA ALGORITMA UNTUK UKURAN STRING YANG SANGAT PANJANG

Uji Coba	***N***	**k**	***Cocok (ya/tidak/ gagal)***
Uji Coba 1	700	100	Ya
Uji Coba 2	700	50	Ya
Uji Coba 3	700	10	Ya
Uji Coba 4	700	5	Tidak
Uji Coba 5	900	100	Ya
Uji Coba 6	900	50	Tidak
Uji Coba 7	900	10	Tidak
Uji Coba 8	900	5	Tidak
Uji Coba 9	1000	100	Ya
Uji Coba 10	1000	50	Tidak
Uji Coba 11	1000	10	Tidak
Uji Coba 12	1000	5	Gagal
Uji Coba 13	1200	100	Gagal
Uji Coba 14	1200	50	Gagal
Uji Coba 15	1200	10	Gagal
Uji Coba 16	1200	5	Gagal

Pada Tabel II, kita menemui beberapa kejanggalan terhadap apa yang kita bahas pada bagian A. Ternyata nilai dari *k* untuk panjang string tertentu bisa jadi memberikan hasil yang berbeda terhadap keberhasilan algoritma. Hal ini menyebabkan untuk ukuran string yang bertambah, nilai *k* diharapkan juga berada pada nilai yang cukup besar. Hal lain yang teramati dari Tabel II adalah sifat dari DFS sebagaimana yang kita jelaskan sebelumnya. Beberapa program tidak bisa mengimplementasikan algoritma dengan kompleksitas ruang yang besar. DFS melakukan pemanggilan fungsi tanpa batasan seberapa jauh. Hal ini lah yang menjadi batasan penerapan algoritma DFS untuk merekonstruksi string dengan ukuran yang sangat panjang.

V. Simpulan

Makalah ini mempelajari permasalahan rekonstruksi string dengan mengambil pendekatan strategi algoritma penelusuran graf. Dengan asumsi terkait sifat dari upastring pad multiset k-mer. Didapatkan bahwa pendekatan algoritma DFS menyelesaikan persoalan ini dengan menyelesaikan persoalan lintasan Euler pada graf de Bruijn, yaitu graf hasil representasi dari suatu multiset komposisi *k*-mer. Didapatkan bahwa algoritma mampu menyelesaikan persoalan untuk beberapa sifat dari nilai *k* yaitu untuk nilai *k* yang cukup besar, dan ukuran string yang tidak panjang. Kita melakukan pengamatan terhadap graf de Bruijn dari berbagai nilai *k* dan menemukan bahwa algoritma penelusuran graf tidak selalu memberikan jawaban yang tepat pada graf de Bruijn dengan lebih dari satu lintasan Euler.

Video Link at Youtube

Link tentang pembahasan sederhana dari makalah ini ada pada https://youtu.be/V7l8bti84aU. Jika link bermasalah coba buka pada link https://www.youtube.com/channel/UCO8aaWpG6chY0YyIIM05YLA.

Ucapan Terima Kasih

Penulis mengucapkan terima kasih kepada dosen mata kuliah IF2211 Strategi Algoritma, Ibu Masayu Leylia Khodra, Ibu Nur Ulfa Maulidevi, dan Bapak Rinaldi, atas ilmu dan bimbingannya selama 1 semester ini. Penulis juga mengucapkan terima kasih kepada orang tua dan seluruh keluarga dekat penulis atas dukungannya selama ini. Penulis juga mengucapkan terima kasih kepada teman-teman satu jurusan di Teknik Informatika baik yang satu angkatan maupun kakak tingkat yang telah membantu dalam berbagai hal terkait perkuliahan bersama di Teknik Informatika ITB.

Referensi

[1] P. Compeau, P. Pevzner, Bioinformatics Algorithms An Active Learning Approach, 2nd ed., vol. 1. San Diego: Active Learning Publishers, 2015, hal. 115-181.

[2] T. H. Cormen, C. E. Lerserson, R. L. Rivest, C. Stein, Introduction to Algorithms, 3rd ed. Cambridge: The MIT Press, 2009, hal. 603-611.

[3] A. Levitin, Introduction to The Design and Analysis of Algorithms, 3rd ed. New York: Pearson Education, 2012, hal. 122-124.

[4] K. H. Rosen, Discrete Mathematics and Its Applications, 7th ed. New York: Mc Graw Hillm 2012, hal. 641-706.

[5] M. Rinaldi. 2018. Diktat Kuliah IF2211 Strategi Algoritma. Program Studi Teknik Informatika ITB.

[6] M. Rinaldi. 2009. Matematika Diskrit, Bandung: Informatika Bandung.

[7] S. Marcovich, E. Yaakobi, 2019. "Reconstrucion of strings from their substrings spectrum", dipublikasi.

[8] Ingako. 2018. "Algorithms". https://ingako.gitbooks.io/algorithms/graph/euler.html. Diakses pada tanggal 4 Mei 2020 pukul 9.00 WIB.

[9] G. Golovnev, A. S. Kukikov, I. Mahjlin, 2013. "Approximation shortest superstring problem using de bruijin graphs", New York University, dipublikasi.

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.

Jakarta, 4 Mei 2020

Kevin Rizki Mohammad 13518100