Analisis Efisiensi Huffman Encoding terhadap Kompresi Sekuens DNA dengan Variasi Distribusi Basa Nitrogen

Rhenaldy Cahyadi Putra - 13524039
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: rhenitb@gmail.com , 13524039@std.stei.itb.ac.id

***Abstract*— Seiring meningkatnya jumlah data genomik, pengolahan dan penyimpanan sekuens DNA menjadi tantangan utama di bidang bioinformatika. Sekuens DNA yang besar dan kaya akan pola berulang membuat teknik kompresi menjadi solusi penting. Penelitian ini membahas penerapan algoritma Huffman untuk mengompresi sekuens DNA, dengan fokus pada pengaruh variasi distribusi basa nitrogen (A, C, G, T) terhadap efisiensi kompresi. Hasil menunjukkan bahwa distribusi simbol sangat memengaruhi panjang total hasil kompresi, memberi tahu kita akan pentingnya pemilihan metode berdasarkan karakteristik data.**

Keywords—Huffman Encoding, Sekuens DNA, Basa Nitrogen.

I. PENDAHULUAN

Perkembangan teknologi sekuensing DNA dalam beberapa dekade terakhir telah menghasilkan volume data genomik yang sangat besar. Informasi genetik suatu organisme tersimpan dalam bentuk sekuens DNA, yang merupakan rangkaian dari empat jenis basa nitrogen: Adenin (A), Sitosin (C), Guanin (G), dan Timin (T). Walaupun hanya terdiri dari empat simbol, panjang sekuens DNA bisa mencapai jutaan hingga miliaran pasangan basa. Sebagai contoh, genom beberapa virus memiliki ukuran hingga 500.000 pasangan basa, sementara genom manusia mencapai lebih dari 3 miliar pasangan basa.

Nantinya, sekuens DNA ini dianalisis untuk memahami penyakit genetik, mengembangkan pengobatan berbasis genom (personalized medicine), hingga melakukan identifikasi forensik dan pemetaan pohon keluarga. Dalam praktiknya, data DNA juga sering dibagikan antar laboratorium dan lembaga penelitian, sehingga kebutuhan akan penyimpanan yang efisien menjadi semakin penting. Di sinilah kompresi data berperan. Kompresi bertujuan untuk mengurangi ukuran file tanpa kehilangan informasi penting, sehingga penyimpanan dan transmisi data menjadi lebih hemat sumber daya.

Salah satu metode kompresi yang banyak digunakan adalah Huffman encoding. Algoritma ini bekerja dengan memberikan kode biner lebih pendek kepada simbol yang lebih sering muncul, dan kode lebih panjang kepada simbol yang lebih jarang. DNA biasanya disimpan dalam format teks seperti FASTA, yang menyajikan sekuens sebagai string panjang dari huruf A, C, G, dan T. Hal ini membuat data DNA tampak serupa dengan teks biasa, tetapi memiliki karakteristik khusus: hanya empat simbol unik dan potensi pola distribusi yang sangat bergantung pada jenis organisme. Oleh karena itu, penerapan Huffman encoding pada data DNA memiliki tantangan dan karakteristik yang unik dibandingkan dengan aplikasi kompresi teks konvensional.

Distribusi simbol dalam sekuens DNA sangat beragam antar organisme. Penelitian ini akan menguji efisiensi Huffman Encoding pada tiga kategori umum: dominasi simbol tunggal (homopolymer), distribusi merata, dan distribusi tidak merata. Kategori ini dipilih untuk mencerminkan variasi biologis nyata dalam sekuens DNA

Penelitian ini bertujuan untuk menganalisis bagaimana variasi distribusi basa nitrogen memengaruhi efisiensi kompresi sekuens DNA menggunakan Huffman encoding. Hasil penelitian diharapkan dapat memberikan perspektif bagi peneliti yang ingin memahami tantangan teknis di bidang bioinformatika, khususnya terkait pengolahan data genomik dalam skala besar.

II. LANDASAN TEORI

A. Pohon

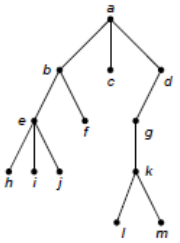
Pohon adalah graf tak-berarah yang terhubung dan tidak memiliki sirkuit. Dengan kata lain, sebuah pohon adalah graf yang menyambungkan seluruh simpul tanpa membentuk lingkaran, dan memiliki tepat *n-1* sisi jika terdapat *n* simpul.

Dalam praktiknya, terdapat berbagai bentuk dan varian dari struktur pohon yang masing-masing memiliki ciri khas dan kegunaan tersendiri. Beberapa konsep yang relevan dalam konteks algoritma kompresi Huffman, antara lain adalah pohon berakar dan pohon biner.

1. Pohon berakar (*rooted tree*)

Pohon berakar (*rooted tree*) adalah pohon yang satu buah simpulnya diperlakukan sebagai akar, dan sisi-sisinya diberi

arah dari akar menuju simpul-simpul lainnya sehingga membentuk graf berarah. Dengan adanya akar, struktur pohon menjadi hierarkis dari atas ke bawah, memudahkan penentuan posisi dan relasi antar simpul.



GAMBAR 1. CONTOH GAMBAR POHON BERAKAR
(SUMBER: PPT BAHAN KULIAH IF1220 – POHON BAGIAN 2)

Dalam pohon berakar dikenal beberapa istilah, seperti:

- Anak (child) dan orangtua (parent): simpul yang terhubung secara langsung. Pada contoh gambar, b,c,d adalah anak-anak dari simpul a, dan sebaliknya, simpul a adalah orang tua b,c,d.
- Lintasan (path): urutan simpul yang saling terhubung langsung melalui sisi dalam suatu pohon atau graf. Misalnya, lintasan a ke c adalah a,c.
- Upapohon (subtree): bagian dari pohon yang terdiri dari suatu simpul beserta seluruh keturunannya.
- Saudara kandung (sibling): anak-anak dari simpul orangtua yang sama.
- Daun (leaf): simpul yang tidak memiliki anak.
- Simpul dalam (internal node): simpul yang memiliki satu atau lebih anak.
- Aras (level): jarak dari akar ke simpul tertentu, dihitung berdasarkan jumlah sisi.
- Tinggi (height): aras maksimum dalam pohon.

Struktur ini menjadi dasar penting dalam berbagai algoritma berbasis pohon, termasuk Huffman Encoding yang membangun pohon secara bertingkat dari akar hingga ke daun.

2. Pohon Biner

Pohon biner adalah pohon berakar khusus di mana setiap simpul memiliki paling banyak dua anak. Anak-anak tersebut dibedakan sebagai anak kiri (*left child*) dan anak kanan (*right child*), sehingga pohon biner termasuk dalam jenis pohon terurut (*ordered tree*). Pohon biner memiliki beberapa bentuk khusus:

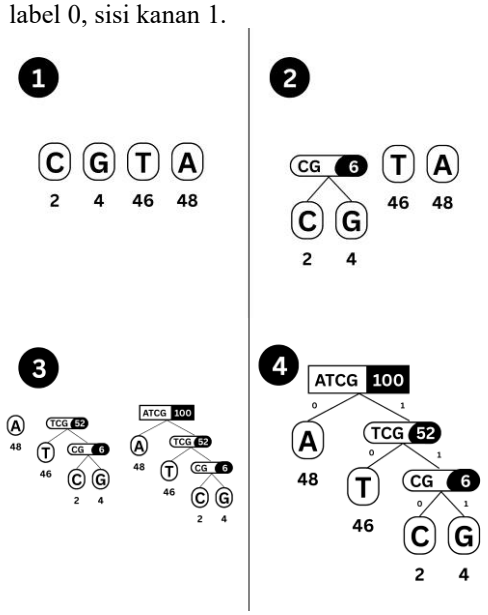
- Pohon biner penuh (*full binary tree*) adalah pohon biner di mana setiap simpul dalam memiliki tepat dua anak.
- Pohon biner lengkap (complete binary tree) adalah pohon biner di mana seluruh aras, kecuali mungkin aras terakhir, terisi penuh dan semua simpul berada sejauh mungkin di sisi kiri.
- Pohon biner seimbang (*balanced binary tree*) adalah pohon di mana tinggi upapohon kiri dan kanan tidak berbeda lebih dari satu.

Dalam penelitian ini, pohon biner digunakan sebagai struktur dasar dalam pembentukan pohon Huffman, di mana setiap simpul daun merepresentasikan satu simbol, dan setiap jalur dari akar ke daun menghasilkan kode biner yang unik berdasarkan frekuensi kemunculan simbol.

B. Algoritma Huffman

Algoritma Huffman adalah algoritma kompresi data yang menggunakan prinsip pengkodean variabel berdasarkan frekuensi kemunculan simbolnya. Algoritma ini dikembangkan oleh David A. Huffman pada tahun 1952, dan menjadi salah satu metode paling populer dalam kompresi data *lossless*. Prinsip dasar dari algoritma Huffman adalah memberikan kode yang lebih pendek untuk simbol yang lebih sering muncul, dan kode yang lebih panjang untuk simbol yang lebih jarang muncul. Dengan cara ini, total panjang data terkompresi dapat diminimalkan. Proses pembentukan kode Huffman terdiri dari beberapa langkah:

1. Hitung frekuensi kemunculan setiap simbol dan urutkan dari kecil ke besar.
2. Ambil dua simpul dengan frekuensi terendah, gabungkan menjadi simpul orangtua baru dengan frekuensi hasil penjumlahan keduanya. Masukkan simpul baru ke urutan.
3. Ulangi proses sampai hanya tersisa satu simpul (akar).
4. Telusuri pohon dari akar ke daun untuk menentukan kode Huffman masing-masing simbol. Sisi kiri diberi label 0, sisi kanan 1.



GAMBAR 2. TAHAPAN ALGORITMA HUFFMAN (SUMBER: PENULIS)

Setelah didapatkan kode Huffman, kita dapat melakukan perhitungan rasio kompresi Rasio kompresi mengukur seberapa kecil hasil kompresi dibanding data aslinya, dengan menggunakan persamaan sebagai berikut:

$$Rasio\ Kompresi = \frac{jumlah\ bit\ setelah\ kompresi}{jumlah\ bit\ sebelum\ kompresi} \times 100\%$$

Sebagai contoh, data yang awalnya bernilai 800 bit jika dikompresi menjadi 160 bit, maka memiliki rasio 160/800 x 100% = 20%, yang berarti file kompresi hanya 20% aslinya (semakin kecil semakin baik).

C. Sekuens DNA

Deoxyribonucleic acid (DNA) adalah molekul pembawa informasi genetik pada semua makhluk hidup. Struktur DNA berbentuk rantai panjang yang terdiri atas empat jenis basa nitrogen, yaitu:

- Adenin (A)
- Sitosin (C)
- Guanin (G)
- Timin (T)

Urutan dari keempat basa ini membentuk sekuens DNA, yang menyimpan instruksi biologis untuk membentuk dan mengatur fungsi organisme. Dalam representasi digital, sekuens DNA biasanya dituliskan sebagai string karakter seperti ACGTAGCT..., di mana setiap huruf mewakili satu basa. Adapun beberapa ciri penting dari sekuens DNA yang berkaitan dengan kompresi data antara lain:

- Ukuran data genomik sangat besar. Genom manusia, misalnya, mengandung lebih dari 3 miliar pasangan basa, sehingga dibutuhkan metode penyimpanan dan transmisi yang efisien.
- Simbol yang digunakan dalam DNA sangat terbatas, hanya terdiri dari empat karakter yaitu A, C, G, dan T. Hal ini berbeda dengan teks biasa yang bisa menggunakan ratusan jenis simbol.
- DNA memiliki pola yang berulang, terutama dalam bentuk homopolymer, yaitu deretan basa yang sama muncul berturut-turut seperti AAAAAA atau TTTTTT. Pola seperti ini dapat dimanfaatkan oleh algoritma kompresi tertentu untuk meningkatkan efisiensi.

III. Analisis Algoritma Huffman

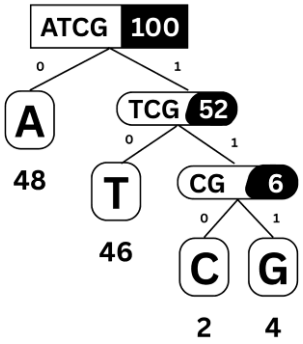
A. Analisis Efisiensi Huffman Encoding dalam Kompresi Sekuens DNA

Untuk mengevaluasi efisiensi algoritma Huffman pada data DNA, dilakukan pengujian terhadap tiga potongan sekuens DNA sepanjang 100 karakter yang diambil dari genom *Candidatus Carsonella ruddii* (Sumber: file FASTA CP003541.1).

Ketiga potongan ini dipilih mewakili kondisi distribusi basa nitrogen yang berbeda, yakni: (1) distribusi tidak seimbang mengandung homopolymer panjang, (2) distribusi seimbang antara A, C, G, dan T, serta (3) distribusi tidak seimbang tanpa homopolymer. Masing-masing potongan dianalisis untuk menentukan frekuensi kemunculan tiap simbol, pembangunan pohon Huffman, serta total ukuran bit hasil kompresi. Kompresi Huffman bekerja dengan memberikan kode biner yang lebih pendek untuk simbol yang lebih sering muncul.

1. Kasus Tidak Seimbang (Dengan Homopolymer)
Potongan ini didominasi oleh huruf A dan T yang muncul secara berturut-turut dalam urutan panjang, membentuk pola homopolymer.

```
AATAAAAAAATTTTTTTTTAATAGATTATTTAAA
AAAATTTATAAATAAATCAATATATAAAAGTTCT
AGAATTATAATTTTATTTTATTGTTTAAATA
```



GAMBAR 3. POHON HASIL PEMBENTUKAN KODE HUFFMAN UNTUK KASUS 1. (SUMBER: PENULIS)

Basa	Frekuensi	8-bit ASCII (Representasi awal)	Kode Huffman (Representasi baru)
A	48	01000001	0
T	46	01000111	11
G	4	01000011	101
C	2	01010100	100

TABEL 1. HASIL HUFFMAN ENCODING PADA KASUS 1: DISTRIBUSI TIDAK SEIMBANG DENGAN HOMOPOLYMER

Jumlah bit setelah encoding dihitung dengan mengalikan frekuensi setiap simbol dengan panjang kode Huffman-nya, yaitu:

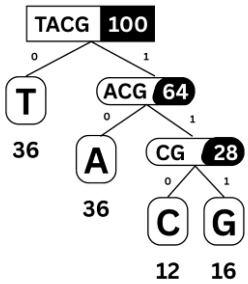
$$(48 \times 1) + (46 \times 2) + (4 \times 3) + (2 \times 3) = 158\ bit$$

Ukuran awal sekuens adalah 800 bit (100 karakter × 8 bit). Maka, rasio kompresi yang dicapai adalah:

$$Rasio\ Kompresi = \frac{158}{800} \times 100\ \% = 19{,}75\%$$

2. Kasus Distribusi Seimbang
Potongan ini memiliki jumlah simbol yang relatif merata.

CAACTTCAGCTCCAGGATGTGATGAGCCGACATC
GAGGTGCCAAACATTGCCGTCGATATGAACTCTT
GGGCAATATTAGCCTGTTATCCCCGGAGTACC



GAMBAR 4. POHON HASIL PEMBENTUKAN KODE HUFFMAN UNTUK KASUS 2. (SUMBER: PENULIS)

Basa	Frekuensi	8-bit ASCII (Representasi awal)	Kode Huffman (Representasi baru)
A	25	01000001	10
T	24	01000111	00
G	24	01000011	01
C	27	01010100	11

TABEL 2. HASIL HUFFMAN ENCODING PADA KASUS 2: DISTRIBUSI SEIMBANG

Pada kasus ini, semua simbol direpresentasikan dengan 2 bit. Jumlah bit setelah encoding adalah:

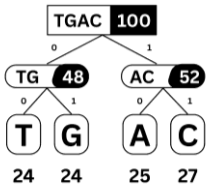
$$(25 \times 2) + (24 \times 2) + (24 \times 2) + (27 \times 2) = 200\ bit$$

Ukuran awal sekuens adalah 800 bit (100 karakter × 8 bit). Maka, rasio kompresi yang dicapai adalah:

$$Rasio\ Kompresi = \frac{200}{800} \times 100\ \% = 75{,}00\%$$

3. Kasus Tidak Seimbang (Tanpa Homopolymer)
Potongan ini juga didominasi oleh huruf A dan T, namun kemunculannya tersebar dan tidak berurutan, sehingga tidak membentuk homopolymer.

ATGAATAATATAATTATAAGTAAAGCAACTCCTG
AAGGTTTTTCTTCAATTTGCGTTGTAAGAGTGTCT
GGAGAAAATGCTTCAAAATTTATTAAACCTT



GAMBAR 5. POHON HASIL PEMBENTUKAN KODE HUFFMAN UNTUK KASUS 3. (SUMBER: PENULIS)

Basa	Frekuensi	8-bit ASCII (Representasi awal)	Kode Huffman (Representasi baru)
A	36	01000001	10
T	36	01000111	0
G	16	01000011	111
C	12	01010100	110

TABEL 3. HASIL HUFFMAN ENCODING PADA KASUS 3: DISTRIBUSI TIDAK SEIMBANG TANPA HOMOPOLYMER

Jumlah bit setelah encoding adalah:

$$(36 \times 2) + (36 \times 1) + (16 \times 3) + (12 \times 3) = 192\ bit$$

Ukuran awal sekuens adalah 800 bit (100 karakter × 8 bit). Maka, rasio kompresi yang dicapai adalah:

$$Rasio\ Kompresi = \frac{192}{800} \times 100\ \% = 24{,}00\%$$

Dengan demikian, dari hasil kompresi ketiga kasus ini, kita telah mendapatkan rasio kompresi sebagai berikut.

No	Kasus Distribusi Basa Nitrogen	Rasio Kompresi
1	Tidak seimbang, dengan Homopolymer	19,75%
2	Seimbang	25,00%
3	Tidak seimbang, tanpa Homopolymer	24,00%

TABEL 4. PERBANDINGAN RASIO KOMPRESI ANTAR KASUS DISTRIBUSI BASA NITROGEN

Rasio kompresi terbaik diperoleh pada sekuens dengan distribusi basa nitrogen yang sangat tidak seimbang (kasus 1), di mana simbol A dan T muncul jauh lebih sering dibanding simbol lainnya. Hal ini sejalan dengan prinsip dasar Huffman Encoding yang memberikan keuntungan maksimal saat frekuensi simbol sangat timpang. Namun, perlu diperhatikan

bahwa meskipun potongan pada kasus 1 mengandung homopolymer panjang, efisiensi kode Huffman tidak secara langsung disebabkan oleh bentuk pengulangan tersebut, melainkan oleh dominasi jumlah simbol tertentu (A dan T).

Dengan demikian, keberadaan homopolymer dalam data tidak berpengaruh secara khusus pada hasil kode Huffman, dan keuntungannya hanya muncul apabila pengulangan tersebut juga berkontribusi terhadap peningkatan frekuensi simbol. Hal ini menunjukkan bahwa *Huffman Encoding* memiliki keterbatasan dalam mengenali pola struktur lokal seperti homopolymer secara eksplisit, dan karenanya kurang optimal jika digunakan sebagai satu-satunya metode untuk kompresi sekuens DNA.

Selain itu, efisiensi Huffman Encoding juga turun signifikan ketika distribusi simbol DNA bersifat seimbang, seperti terlihat pada kasus 2. Dalam kondisi di mana A, C, G, dan T muncul dengan frekuensi hampir sama, panjang kode Huffman cenderung seragam dan tidak banyak memberikan penghematan dibanding representasi 8-bit ASCII. Hal ini selaras dengan temuan Bakr dan Sharawi (2013), yang menyatakan bahwa Huffman encoding tidak mampu mengompresi DNA secara signifikan ketika peluang kemunculan simbolnya tidak jauh berbeda satu sama lain.

B. Keterbatasan Huffman Encoding dalam Kompresi Sekuens DNA

Dari hasil percobaan sebelumnya, dapat disimpulkan bahwa meskipun algoritma Huffman memiliki keunggulan dalam menghasilkan representasi biner yang efisien berdasarkan frekuensi simbol, terdapat beberapa keterbatasan yang membuatnya kurang optimal dalam konteks kompresi sekuens DNA secara keseluruhan.

Pertama, algoritma Huffman tidak dapat mengenali pola biologis berulang seperti homopolymer secara eksplisit, karena hanya mempertimbangkan frekuensi kemunculan simbol tunggal. Artinya, Huffman gagal memanfaatkan struktur berulang seperti "AAAAAAA" atau "CGCGCG" yang umum dijumpai dalam sekuens DNA.

Kedua, Huffman encoding juga tidak efektif pada sekuens dengan distribusi simbol seimbang, yaitu ketika keempat basa (A, C, G, T) memiliki frekuensi yang hampir sama. Dalam kondisi ini, panjang kode Huffman untuk masing-masing simbol menjadi hampir seragam, dan rasio kompresi tidak jauh berbeda dengan representasi awal 8-bit. Hal ini menunjukkan keterbatasan utama Huffman dalam skenario di mana struktur statistik tidak terlalu mencolok. Bakr dan Sharawi (2013) juga menegaskan bahwa Huffman cenderung gagal dalam menghasilkan rasio kompresi yang signifikan untuk data DNA semacam ini.

Oleh karena itu, diperlukan pendekatan kompresi lanjutan yang mampu mengenali pola lokal atau pengulangan, seperti Run-Length Encoding (RLE). Kombinasi Huffman dengan algoritma lain diharapkan dapat meningkatkan efisiensi secara keseluruhan, khususnya pada struktur DNA yang kompleks.

C. Analisis Efisiensi Penggunaan Algoritma Huffman dengan Run-Length Encoding (RLE)

Hasil analisis efisiensi Huffman Encoding dalam kompresi sekuens DNA menunjukkan bahwa Huffman Encoding memiliki keterbatasan dalam mengenali pola pengulangan lokal seperti homopolymer yang umum ditemukan pada sekuens DNA. Salah satu pendekatan yang dapat digunakan untuk mengatasi kekurangan tersebut adalah Run-Length Encoding (RLE). RLE merupakan metode kompresi sederhana yang menggantikan simbol yang berulang secara berturut-turut dengan satu simbol dan jumlah pengulangannya. Dalam konteks sekuens DNA, pola seperti AAAAAA akan dikodekan sebagai A6, sehingga tidak perlu menyimpan enam karakter 'A' secara eksplisit.

Pada makalah ini, saya akan melakukan eksplorasi menggunakan RLE yang sederhana, yaitu dengan langsung menggantikan urutan simbol yang berulang menjadi representasi *[simbol][jumlah]* yang dilanjutkan dengan Kompresi Huffman. Pendekatan ini dapat menunjukkan bagaimana deteksi pola lokal dapat mendukung efisiensi Huffman Encoding dalam kompresi sekuens DNA.

1. Implementasi Run-Length Encoding (RLE)

Untuk menguji efisiensi kompresi dengan Run-Length Encoding, digunakan sebuah program C sederhana yang terdiri dari dua bagian utama: fungsi runLengthEncodeDNA() dan fungsi main().

Fungsi runLengthEncodeDNA() akan mengubah sekuens DNA menjadi bentuk terkompresi dengan prinsip RLE, yaitu menggantikan deretan simbol berulang secara berurutan menjadi pasangan [simbol][jumlah]. Adapun fungsi main() digunakan untuk menerima masukan sekuens DNA dari pengguna melalui scanf(), lalu memanggil fungsi kompresi tersebut, dan akhirnya mencetak hasil sebelum dan sesudah dikompresi ke layar. Berikut adalah implementasi algoritma Run-Length Encoding dalam bahasa C yang digunakan dalam eksperimen ini:

DNArle.c > main()

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>

void runLengthEncodeDNA(const char *input, char *output) {
    int len = strlen(input);
    int count = 1;
    int outIdx = 0;

    for (int i = 1; i <= len; i++) {
        if (input[i] == input[i - 1]) {
            count++;
        } else {
            output[outIdx++] = input[i - 1];

            // Tambahkan angka hanya jika count > 1
            if (count > 1) {
                outIdx += sprintf(output + outIdx, "%d", count);
            }

            count = 1;
        }
    }

    output[outIdx] = '\0';
}
```

```c
int main() {
    char dna[1000];             // buffer input sekuens DNA
    char compressed[1000];      // panjang hasil kompresi paling buruk = sekuens DNA awal

    printf("Masukkan sekuens DNA: ");
    scanf("%999s", dna);

    runLengthEncodeDNA(dna, compressed);

    printf("Input DNA     : %s\n", dna);
    printf("Hasil kompres : %s\n", compressed);

    return 0;
}
```

GAMBAR 6 DAN 7. IMPLEMENTASI RUN-LENGTH ENCODING. (SUMBER: PENULIS)

```
Masukkan sekuens DNA: AAATTTCCCGGAAGGTTTTA
Input DNA     : AAATTTCCCGGAAGGTTTTA
Hasil kompres : A3T3C3G2A2G2T4A
```

GAMBAR 8. TAMPILAN TERMINAL PROGRAM (SUMBER: PENULIS)

Melalui program ini, didapatkan hasil Run-Length Encoding untuk tiga potongan sekuens DNA *Candidatus Carsonella ruddii* yang telah kita gunakan pada analisis sebelumnya. Berikut adalah hasil RLE dari ketiga potongan tersebut.

1. Kasus Tidak Seimbang (Dengan Homopolymer)

Input DNA:
AATAAAAAAATTTTTTTTTTAATAGATTATTTAAA
AAAATTTATAAATAAATCAATATATAAAAGTTCT
AGAATTATAATTTTATTTTATTGTTTAAATA

Hasil kompres:
A2TA7T10A2TAGAT2AT3A7T3ATA3TA3TCA2TATA
TA4GT2CTAGA2T2ATA2T4AT4AT2GT3A3TA

2. Kasus Distribusi Seimbang

Input DNA:
CAACTTCAGCTCCAGGATGTGATGAGCCGACATC
GAGGTGCCAAACATTGCCGTCGATATGAACTCTT
GGGCAATATTAGCCTGTTATCCCCGGAGTACC

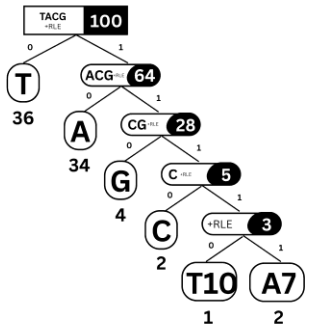
Hasil kompres:
CA2CT2CAGCTC2AG2ATGTGATGAGC2GACATCG
AG2TGC2A3CAT2GC2GTCGATATGA2CTCT2G3CA2
TAT2AGC2TGT2ATC4G2AGTAC2

3. Kasus Tidak Seimbang (Tanpa Homopolymer)

Input DNA:
ATGAATAATATAATTATAAGTAAAGCAACTCCTG
AAGGTTTTTCTTCAATTTGCGTTGTAAGAGTGTCT
GGAGAAAATGCTTCAAAATTTATTAAACCTT

Hasil kompres:
ATGA2TA2TATA2T2ATA2GTA3GCA2CTC2TGA2G2
T5CT2CA2T3GCGT2GTA2GAGTGTCTG2AGA4TGCT
2CA4T3AT2A3C2T2

Sebagai pembanding dengan metode Huffman biasa, kita akan mencoba melakukan *Selective Huffman Encoding* pada kasus 1 (distribusi tidak seimbang dengan homopolymer). Dengan melakukan encoding pada A,C,G,T serta A7 dan T10 secara terpisah, akan didapatkan:



GAMBAR 9. POHON HASIL PEMBENTUKAN KODE HUFFMAN UNTUK KASUS 1 DENGAN RUN-LENGTH ENCODING. (SUMBER: PENULIS)

Basa	Frekuensi	8-bit ASCII (Representasi awal)	Kode Huffman (Representasi baru)
A	34	01000001	10
T	36	01000111	0
G	4	01000011	110
C	2	01010100	1110
A7	2	7x A (56-bit)	11111
T10	1	10x T (80-bit)	11110

TABEL 5. HASIL HUFFMAN ENCODING PADA KASUS 1 DENGAN RLE: DISTRIBUSI TIDAK SEIMBANG DENGAN HOMOPOLYMER

Jumlah bit setelah encoding adalah:

$$(34 \times 2) + (36 \times 1) + (4 \times 3) + (2 \times 4) + (2 \times 5) + (1 \times 5) = 139\ bit$$

Ukuran awal sekuens adalah 800 bit (100 karakter × 8 bit). Maka, rasio kompresi yang dicapai adalah:

$$Rasio\ Kompresi = \frac{139}{800} \times 100\ \% = 17{,}37\%$$

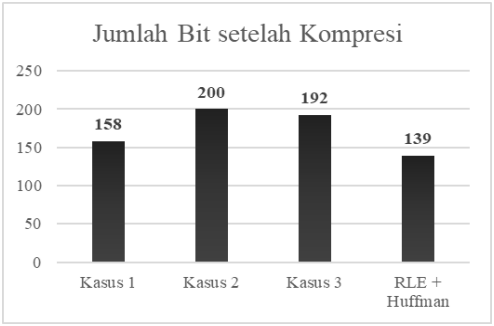
D. Perbandingan Rasio Kompresi

Rasio kompresi terbaik diperoleh saat kombinasi metode digunakan. Pada sekuens DBA dengan homopolymer yang dominan, RLE membantu mereduksi deretan simbol yang berulang, lalu Huffman memperpendek representasi biner dari simbol dan pasangan *[simbol][jumlah]*.

No	Kasus Distribusi Basa Nitrogen	Metode Kompresi	Rasio Kompresi

1	Tidak seimbang, dengan Homopolymer	Huffman	19,75%
2	Seimbang	Huffman	25,00%
3	Tidak seimbang, tanpa Homopolymer	Huffman	24,00%
4	Tidak seimbang, dengan Homopolymer	Huffman dan RLE	17,37%

TABEL 6. PERBANDINGAN RASIO KOMPRESI ANTAR METODE HUFFMAN DAN HUFFMAN DENGAN RLE



GAMBAR 10. PERBANDINGAN JUMLAH BIT HASIL KOMPRESI ANTAR METODE HUFFMAN DAN HUFFMAN DENGAN RLE (SUMBER: PENULIS)

IV. KESIMPULAN

Penelitian ini menunjukkan bahwa efisiensi kompresi sekuens DNA menggunakan algoritma Huffman sangat dipengaruhi oleh distribusi frekuensi basa nitrogen. Pada kasus dengan distribusi tidak seimbang dan didominasi oleh simbol tertentu, Huffman Encoding mampu menghasilkan rasio kompresi yang jauh lebih baik dibandingkan dengan kasus distribusi seimbang, di mana panjang kode biner setiap simbol menjadi hampir seragam dan tidak memberikan penghematan signifikan. Namun, Huffman Encoding memiliki keterbatasan dalam mengenali struktur lokal seperti homopolymer karena hanya bekerja berdasarkan frekuensi simbol tunggal, bukan pola berulang.

Untuk mengatasi hal tersebut, eksplorasi penggunaan metode Run-Length Encoding (RLE) memberikan hasil yang lebih efisien dalam kasus-kasus dengan banyak pengulangan simbol. Kombinasi RLE dengan Huffman terbukti mampu meningkatkan efisiensi dibandingkan penggunaan Huffman saja. Hal ini menunjukkan bahwa pendekatan hibrid yang menggabungkan keunggulan deteksi pola lokal dan pengkodean berbasis frekuensi dapat menjadi solusi yang lebih optimal dalam kompresi data genomik. Dengan demikian, pemilihan metode kompresi sekuens DNA haruslah mempertimbangkan karakteristik distribusi dan struktur data DNA yang akan dikompresi.

V. UCAPAN TERIMA KASIH

Terima kasih saya panjatkan kepada Tuhan Yang Maha Esa karena atas rahmat dan kasih karunia-Nya, makalah ini dapat diselesaikan dengan maksimal. Ucapan terima kasih yang sebesar-besarnya saya sampaikan kepada dosen pengampu mata kuliah IF1220, Bapak Dr. Ir. Rinaldi, M.T., yang telah membagikan ilmu serta memberikan banyak perspektif baru bagi saya di dunia Informatika. Tak lupa, saya mengucapkan terima kasih kepada keluarga dan teman-teman saya yang telah memberikan dukungan moral selama proses pengerjaan makalah ini. Penulis berharap makalah ini dapat menjadi referensi yang berguna, baik bagi pelajar lain yang ingin memahami materi terkait, maupun bagi penulis sendiri di masa mendatang.

LINK VIDEO YOUTUBE

https://youtu.be/WvXUEuATqNM

REFERENCES

[1] Munir, Rinaldi. 2024. "Pohon (Bagian 1)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/23-Pohon-Bag1-2024.pdf, diakses 16 Juni 2025.

[2] Munir, Rinaldi. 2024. "Pohon (Bagian 2)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/24-Pohon-Bag2-2024.pdf, diakses 16 Juni 2025.

[3] European Nucleotide Archive. 2012. "Candidatus Carsonella ruddii genome, CP003541.1". https://www.ebi.ac.uk/ena/browser/view/CP003541, diakses 17 Juni 2025.

[4] Al-Okaily, A., Almarri, B., Al Yami, S., & Huang, C. H. 2017. "Toward a Better Compression for DNA Sequences Using Huffman Encoding". Journal of Computational Biology, 24(4), 280–288. https://doi.org/10.1089/cmb.2016.0151, diakses 17 Juni 2025.

[5] Bakr, N. S., & Sharawi, A. A. 2013. "DNA Lossless Compression Algorithms: Review". *American Journal of Bioinformatics Research*, 3(3), 72–81. https://doi.org/10.5923/j.bioinformatics.20130303.04, diakses 18 Juni 2025.

[6] University of Massachusetts Lowell. "Theory of Data Compression". https://faculty.uml.edu/jweitzen/16.548/classnotes/Theory%20of%20Data%20Compression.htm, diakses 20 Juni 2025.

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, 20 Juni 2025

Rhenaldy Cahyadi Putra 13524039