Penerapan *Huffman Coding* pada Teks Proklamasi Indonesia

Francesco Michael Kusuma - 13522038
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13522038@std.stei.itb.ac.id

***Abstract*— Pertukaran informasi yang pesat dalam era digital menuntut pendekatan efisien dalam representasi dan penyimpanan data. Huffman Coding, sebagai metode kompresi data yang telah terbukti efektif, menjadi fokus penelitian ini. Makalah ini mengeksplorasi penerapan Huffman Coding pada teks Proklamasi Indonesia, sebuah dokumen bersejarah yang memiliki nilai kultural dan nasional yang tinggi. Tujuan utama penelitian ini adalah untuk mengevaluasi potensi Huffman Coding dalam mengurangi ukuran data teks Proklamasi tanpa mengorbankan integritas makna dan pesan sejarah yang terkandung dalam teks tersebut.**

***Kata kunci*— *Huffman Coding*, Kompresi, Teks Proklamasi Indonesia**

I. PENDAHULUAN

Pada zaman modern, berbagi informasi adalah bagian penting dari kehidupan sehari-hari. Representasi dan penyimpanan data yang efisien semakin penting karena jumlah data yang dikirim semakin banyak. Teknik kompresi data adalah salah satu cara untuk mencapai efisiensi ini. Huffman Coding adalah teknik pengkodean yang diciptakan oleh David A. Huffman pada tahun 1952 dan muncul sebagai salah satu metode yang berhasil untuk mengurangi jumlah redundansi yang ada dalam representasi data biner.

Dalam makalah ini, kami akan melihat bagaimana teknik Huffman Coding dapat diterapkan pada teks Proklamasi Indonesia, sebuah dokumen bersejarah yang sangat penting bagi bangsa ini. Diharapkan bahwa penerapan metode ini pada teks Proklamasi akan memungkinkan untuk mengurangi ukuran data tanpa mengorbankan integritas dan keaslian teks.

Memanfaatkan Huffman Coding pada teks Proklamasi tidak hanya mempertimbangkan efisiensi representasi bit, tetapi juga mencoba memahami bagaimana penerapan metode kompresi ini dapat membantu menyimpan, mengirim, dan mengelola dokumen bersejarah. Analisis akan dilakukan untuk mengevaluasi seberapa baik Huffman Coding dapat mengoptimalkan ukuran file teks Proklamasi tanpa menghilangkan informasi yang penting.

Diharapkan bahwa penelitian ini akan memberikan pemahaman yang lebih baik tentang kemungkinan penerapan Huffman Coding pada teks sejarah dan menunjukkan bagaimana metode ini relevan dengan pengelolaan dokumen digital saat ini. Selain itu, hasilnya akan berkontribusi pada pengembangan teknik kompresi data yang lebih canggih dan berkelanjutan.

II. TEORI DASAR

A. Proklamasi Kemerdekaan Indonesia

Soekarno dan Mohammad Hatta berkumpul untuk membuat rencana yang akan mengubah sejarah Indonesia pada pagi yang bersejarah tanggal 17 Agustus 1945, di tengah kegembiraan dan ketegangan perjuangan nasional. Kondisi politik saat itu mengalami banyak perubahan. Belanda menjajah Indonesia selama berabad-abad. Setelah Perang Dunia II, Jepang yang sebelumnya menduduki Indonesia juga kalah.

Gambar 2.1 Jepang menyerah kepada Sekutu pada 2 September 1945

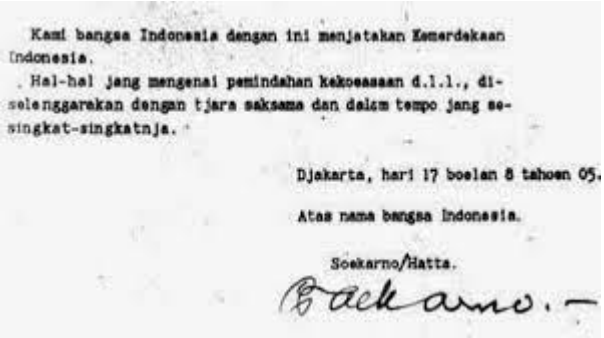
Setelah perang berakhir dan kekuasaan menjadi kosong setelah Jepang pergi, perjuangan kemerdekaan mendapat momentum penting. Soekarno dan Hatta memahami pentingnya mengumumkan kemerdekaan Indonesia kepada dunia meskipun situasinya sulit.

Soekarno dan Hatta menghabiskan banyak waktu untuk berunding sebelum akhirnya berdiri di depan mikrofon. Teks yang mereka baca harus lebih dari sekadar pidato; itu harus menunjukkan semangat perjuangan, keteguhan, dan harapan baru bagi bangsa yang telah lama merindukan kemerdekaan.

Bahasa yang akan digunakan dipertimbangkan secara menyeluruh selama proses penyusunan teks Proklamasi. Sebuah cerita yang kuat dan penuh arti dibuat dengan kata-kata yang

dipilih dengan hati-hati. Penghormatan terhadap prinsip-prinsip Pancasila, yang berfungsi sebagai dasar keyakinan negara yang baru dibentuk, terjadi.

Soekarno dan Hatta memastikan bahwa semua pihak yang terlibat setuju dengan teks tersebut sebelum akhirnya membacanya dengan penuh emosi. Melibatkan tokoh-tokoh nasionalis lainnya yang berkontribusi besar pada perjuangan merebut kemerdekaan, serta sesama pemimpin perjuangan.


Kami bangsa Indonesia dengan ini menjatakan Kemerdekaan Indonesia.
Hal-hal jang mengenai pemindahan kekoeasaan d.l.l., diselenggarakan dengan tjara saksama dan dalam tempo jang sesingkat-singkatnja.

Djakarta, hari 17 boelan 8 tahoen 05.

Atas nama bangsa Indonesia.

Soekarno/Hatta.

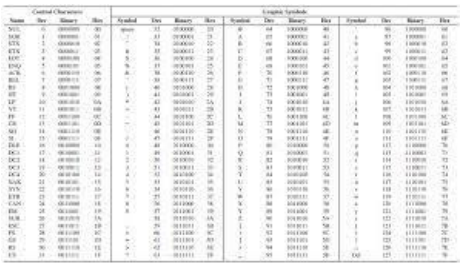
Gambar 2.2 Teks Proklamasi Kemerdekaan Indonesia

Di Jalan Pegangsaan Timur 56, Soekarno membacakan Teks Proklamasi di depan para pemimpin nasionalis dan rakyat Indonesia dengan suara yang penuh semangat ketika matahari mulai menyinari Jakarta. Di seluruh negeri, suara kebebasan bergema dengan kata-kata, "Kami, bangsa Indonesia, dengan ini menyatakan kemerdekaan Indonesia."


Gambar 2.3 Pembacaan Teks Proklamasi

Proklamasi menunjukkan semangat dan keberanian rakyat Indonesia untuk menentang penjajahan apa pun. Teks tersebut tidak hanya berfungsi sebagai dasar hukum untuk pembentukan Republik Indonesia, tetapi juga berfungsi sebagai simbol semangat perjuangan yang akan terus menginspirasi dan dikenang oleh generasi berikutnya.

B. Kode ASCII


Gambar 2.4 Kode ASCII

Kode ASCII (American Standard Code for Information Interchange), yang dibuat pertama kali pada tahun 1960, menawarkan metode konsisten untuk mengonversi berbagai karakter, angka, dan simbol menjadi bentuk bilangan yang dapat diproses komputer dan perangkat elektronik lainnya.

Pada awalnya, Kode ASCII hanya memiliki 128 karakter, termasuk huruf besar dan kecil, angka, dan beberapa karakter khusus, seperti tanda baca, dan setiap karakter diwakili oleh bilangan bulat dari 0 hingga 127. Bilangan biner, oktal, atau heksadesimal dapat digunakan untuk menunjukkan angka ini.

Dalam Kode ASCII, karakter utama, seperti huruf A-Z, a-z, dan 0-9, ditempatkan dalam kumpulan nilai tertentu. Misalnya, angka 65 menunjukkan huruf besar "A", dan angka 97 menunjukkan huruf kecil "a". Ini memberikan dasar untuk pemrograman multibahasa dan manipulasi karakter.

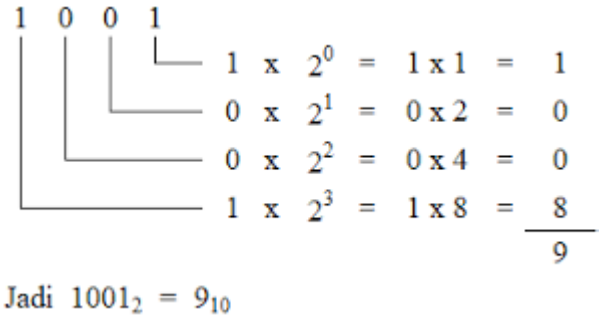
Kode ASCII juga mencakup karakter kontrol seperti Enter (newline), Tab, dan Carriage Return, yang digunakan untuk memformat teks dan mengatur tata letak.

Namun, karena kemajuan teknologi dan kebutuhan untuk mendukung berbagai karakter dan simbol dalam berbagai bahasa, Kode ASCII diperluas menjadi Extended ASCII. Perluasan ASCII memungkinkan lebih banyak karakter, termasuk simbol khusus dan karakter aksen yang ada dalam beberapa bahasa.

Unicode mendukung berbagai karakter dari berbagai bahasa dan budaya, memberikan solusi yang lebih luas untuk representasi karakter di dunia digital yang semakin global, meskipun Kode ASCII masih banyak digunakan.

Pengembang perangkat lunak yang memahami Kode ASCII dapat dengan mudah mengedit dan memproses teks dalam berbagai konteks. Di sisi lain, Kode ASCII masih merupakan dasar pemrosesan karakter pada komputer dan perangkat elektronik kontemporer.

C. Sistem Biner



Gambar 2.5 Sistem Biner

Sistem bilangan biner, atau sering disebut sebagai sistem biner, adalah dasar dari semua komputasi dan pengolahan informasi komputer. Sistem ini menggunakan basis dua, yang berarti bahwa hanya terdiri dari dua digit, yaitu 0 dan 1.

Dalam matematika, sistem biner memiliki dasar yang sangat mendasar. 0 dan 1 adalah simbol nilai biner dan non-biner

matematika. Konsep ini kemudian digunakan dalam bidang TI dan komputasi.

Setiap digit dalam sistem bilangan biner memiliki nilai yang merupakan pangkat dari dua. Di sini, digit paling kanan menunjukkan 2 0 (dengan pangkat 0), dan digit sebelah kiri menunjukkan 2 1 (dengan pangkat 1), dan seterusnya. Sebagai contoh, Anda dapat menggambarkan bilangan biner 1011 sebagai 1x2~3 + 0x2^2 + 1x2~1 + 1x2|0, yang setara dengan bilangan desimal 11.

Teknologi digital dan elektronika terkait erat dengan sistem biner. Bit, atau angka biner, adalah unit terkecil dalam penyimpanan data komputer. Bit dapat memiliki dua nilai, yaitu 0 atau 1. Arus listrik diwakili oleh 1 dan ketiadaan arus listrik diwakili oleh 0.

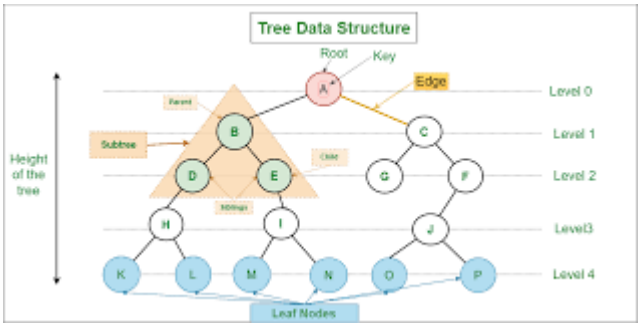
Dalam sistem biner, operasi matematika melibatkan prinsip-prinsip dasar matematika, tetapi mereka hanya memiliki dua digit: 0 dan 1. Penjumlahan, pengurangan, perkalian, dan pembagian memiliki aturan khusus yang sesuai dengan basis dua.

Representasi biner digunakan untuk memproses data dan instruksi dalam komputasi. Kode biner digunakan untuk menggambarkan semua karakter komputer, seperti huruf, angka, dan simbol. Sebuah kode yang dikenal sebagai ASCII (American Standard Code for Information Interchange) menggunakan representasi biner untuk setiap karakter.

Sistem biner sangat efektif dalam menampilkan dan memproses data digital karena memiliki dua digit yang memungkinkan banyak kombinasi dan permutasi.

Seluruh komputasi modern bergantung pada sistem biner. Memahami bagaimana komputer menyimpan, memproses, dan mengirimkan data sangat penting. Pada awalnya, sistem biner mungkin terasa asing bagi beberapa orang, tetapi sekarang menjadi bahasa yang digunakan di seluruh dunia komputasi yang memungkinkan manusia dan perangkat berkomunikasi satu sama lain.

D. Pohon

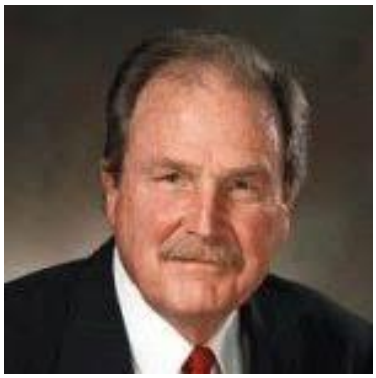


Gambar 2.6 Pohon

Pohon adalah struktur data yang terdiri dari simpul yang terhubung dengan aturan oleh tepi (edge) dan tidak mengandung suatu sirkuit di dalamnya. Satu simpul berfungsi sebagai simpul induk (atau simpul atas), dan simpul-simpul lainnya berfungsi sebagai simpul anak-anak (atau simpul bawah) dari simpul induk. Pohon dapat digunakan untuk mewakili struktur hierarkis dalam berbagai konteks, seperti struktur keluarga, struktur file dalam sistem operasi, atau struktur hierarki dalam ilmu komputer. Pada pohon, terdapat beberapa istilah dan konsep yang terkait engan pohon, yaitu

a. Simpul (Node):
   Simpul merupakan elemen-elemen dasar dari pohon. Setiap simpul mewakili suatu entitas atau nilai.
b. Tepi (Edge):
   Tepi adalah garis yang menghubungkan simpul-simpul dalam pohon. Tepi menunjukkan hubungan hierarkis antara simpul-simpul.
c. Akar (Root):
   Akar adalah simpul paling atas dalam pohon, menjadi induk dari semua simpul lainnya. Setiap pohon memiliki satu akar.
d. Daun (Leaf):
   Daun adalah simpul-simpul yang tidak memiliki anak (simpul bawah). Daun berada di level terbawah dalam pohon.
e. Simpul Anak (Child):
   Simpul anak adalah simpul yang terhubung langsung dengan simpul induk. Sebuah simpul induk dapat memiliki beberapa simpul anak.
f. Simpul Induk (Parent):
   Simpul induk adalah simpul yang memiliki simpul anak. Simpul induk mengontrol dan mendominasi simpul-simpul anaknya.
g. Tingkat (Level):
   Tingkat pohon menunjukkan jarak antara suatu simpul dengan akar. Akar berada di tingkat 0, anak-anak langsung dari akar berada di tingkat 1, dan seterusnya.
h. Tinggi (Height):
   Tinggi pohon adalah panjang terpanjang dari akar ke daun terjauh. Tinggi pohon mencerminkan sejauh mana pohon tersebut berkembang.

E. Huffman Coding (Kode Huffman)

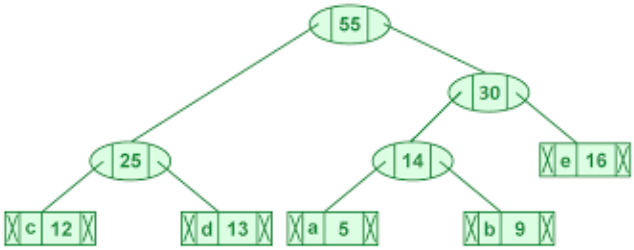

Gambar 2.7 David A. Huffman

Diciptakan oleh David A. Huffman pada tahun 1952, coding Huffman adalah salah satu algoritma kompresi data yang mendasarkan prinsipnya pada distribusi frekuensi karakter dalam teks atau data. Algoritma ini sangat efektif dalam mengurangi ukuran data tanpa kehilangan informasi yang terkandung di dalamnya.

Menghitung berapa kali setiap karakter muncul dalam data adalah langkah pertama dalam kode Huffman. Selanjutnya, aturan berikut digunakan untuk membentuk pohon Huffman: tingkat yang lebih tinggi diberikan kepada simpul dengan frekuensi yang lebih rendah, dan tingkat yang lebih rendah diberikan kepada simpul dengan frekuensi yang lebih tinggi. Saat ini, seluruh simpul tergabung menjadi satu simpul akar.

Setiap karakter diberi kode biner setelah pembentukan Pohon Huffman, yang terdiri dari jalur yang mengalir dari akar menuju titik yang mewakili karakter tersebut. Nilai 0 diberikan pada jalur menuju anak kiri, dan nilai 1 diberikan pada jalur menuju anak kanan. Tidak ada kode biner yang merupakan awalan dari kode biner yang lain karena kode biner yang dihasilkan memiliki karakteristik yang sama.

Proses encoding (kompresi) menggantikan setiap karakter dalam data dengan kode biner yang sesuai. Proses decoding (dekompressi) membaca kode biner dan mengikuti jalur pada Pohon Huffman untuk mengembalikan karakter ke bentuk aslinya, yang menunjukkan efisiensi Huffman Coding.



Gambar 2.8 Huffman Coding

Karena memastikan bahwa karakter yang paling sering muncul diwakili dengan kode biner yang paling pendek, Huffman Coding dianggap sebagai metode kompresi data yang terbaik. Dengan demikian, Huffman Coding meminimalkan jumlah bit yang dibutuhkan untuk merepresentasikan seluruh data.

Huffman Coding sangat bergantung pada distribusi frekuensi karakter dalam data; jika distribusinya merata, metode ini dapat memberikan tingkat kompresi yang tinggi. Ini karena fakta bahwa metode ini sering digunakan dalam berbagai aplikasi, termasuk transmisi data, kompresi file, dan penyimpanan informasi.

Jika distribusi karakter dalam data tidak menunjukkan pola frekuensi yang jelas, Huffman Coding mungkin tidak efektif. Dalam kasus seperti itu, metode kompresi lain mungkin lebih cocok.

Konsep Huffman Coding sangat mendalam dalam bidang kompresi data, dan penjelasan ini mencakup beberapa elemen penting yang membantu kita memahami bagaimana algoritma ini bekerja dan bagaimana ia dapat digunakan dengan efisiensi dalam berbagai situasi.

III. PEMBAHASAN

A. Langkah Penyederhanaan

Tahap penyederhanaan teks adalah sebagai berikut :

1) Menghitung jumlah frekuensi tiap karakter
2) Membuat pengkodean dengan Kode Huffman
3) Menghitung persentase kompresi

B. Konstruksi Kode Huffman

Teks asli Proklamasi ditulis ulang menjadi “PROKLAMASI Kami bangsa Indonesia dengan ini menjatakan kemerdekaan Indonesia. Hal-hal yang mengenai pemindahan kekoeasaan d.l.l., diselenggarakan dengan tjara saksama dan dalam tempo jang sesingkat-singkatnja. Djakarta, hari 17 boelan 8 tahoen 05. Atas nama bangsa Indonesia.Soekarno/Hatta.”. Total panjang karakternya adalah 290 karakter. Tabel frekuensi kemunculan tiap karakter ditampilkan pada table 3.1.

Karakter	Frek	Karakter	Frek	Karakter	Frek
Spasi	34	U	0	p	2
A	3	V	0	q	0
B	0	W	0	r	6
C	0	X	0	s	13
D	1	Y	0	t	10
E	0	Z	0	u	0
F	0	a	51	v	0
G	0	b	3	w	0
H	2	c	0	x	0
I	4	d	11	y	1
J	0	e	21	z	0
K	2	f	0	0	1
L	1	g	11	1	1
M	1	h	4	2	0
N	0	i	12	3	0
O	1	j	5	4	0
P	1	k	11	5	1
Q	0	l	7	6	0
R	1	m	9	7	1
S	2	n	33	8	1
T	0	o	9	9	0
.	8	,	2	/	1
-	2				

Tabel 3.1 Frekuensi kemunculan karakter

Setelah dihitung banyaknya tiap karakter dari kalimat tersebut, akan dicari panjang kalimat tersebut dalam biner menggunakan kode ASCII. Pengubahan ini akan menggunakan kode *Python* sebagai berikut

```python
# mengubah string ke kode biner
def konvertBiner(uji):
    # list berisi kode biner per karakter
    binary_list = []

    # pengubahan per karakter
    for char in uji:
        binary_list.append(bin(ord(char))[2:].zfill(8))

    return ''.join(binary_list)


if __name__ == "__main__":
    uji = input("Masukkan string yang ingin diubah : ")
    print()
    print(f"Panjang binernya adalah : {len(konvertBiner(uji))}")
    print()
    print(f"Kode binerya adalah : ")
    print(konvertBiner(uji))
```

Gambar 3.1 Kode Python mengubah ke kode biner dengan kode ASCII

dan menghasilkan

```
Masukkan string yang ingin diubah : PROKLAMASI Kami bangsa Indonesia dengan ini menjatakan kemerdekaan In
donesia. Hal-hal yang mengenai pemindahan kekoeasaan d.l.l., diselenggarakan dengan tjara saksama dan dal
am tempo jang sesingkat-singkatnja. Djakarta, hari 17 boelan 8 tahoen 05. Atas nama bangsa Indonesia.Soek
arno/Hatta.

Panjang binernya adalah : 2320

Kode binernya adalah :
01010000010100100100111101001011010011000100000101001101010000010101001101001001000000100101110110000101
11011010110100100100000011000101100001011011100110011101110011011000010010000001001001011011100110010001
10111101101110011001010111001101101001011000010010000011001000110010110111001100111011000010110111000101
00000011010010110111001101001000000110110101100101011011100110011010000110001101110100011000010111010110
00010110111000100000011010110110010101101101101100101011100100110010001100101011010110111000010110000101101
11000100000010010010110111001100100011011110110111001100101011100110110100101100001001011100000100101
00011000010110110000101101011010000110000101100101000000111000101100001011010010110000100101110010000010010
10110001010111001100011011101100100111000110110000101100101101100010010000011001000110001011010110100001011
01100100011000010110100001100001011011000100000110101101100101100101100101110110010101100001011100110
11000010110000101101110001000000110010000101110011010000101110011011000010111001101100001011000010010011000
10100101110011011001010111011000110010101100111001110011001110110001011100100110000101101100100000101011
01110001000000110010001011011100110010000011001000010111001101100001011100110110000101100100000011000100
00010011000000110101001011100010000010000101110011010000101110011011000010111001101100001011000010010011000
00001100000011010100101110001000000100000101110011010000101110011011000010111001101100001011000010010011000
01100001011100101110011000010000011001000010111001101100001011100110110000101100100000011000100000101110100
01100001011100101110011000010000011001000010111001101100001011100110110000101100100000011000100000101110101
10000100100000011000100110000101101110011001110011
00110110000100100000010010010110111001100100011011
11011011100110010101110011011010010110000100101110
01010011011011110110010101101101100001011100110111
01000110000100101110
```

Namun, ini semua bisa disederhanakan dengan Kode Huffman menjadi lebih sederhana. Pembuatan Kode Huffman dibantu dengan kode *Python* berikut

```python
import heapq

# node dari tree
class Node:
    def __init__(self, frekuensi, simbol, kiri = None, kanan = None):
        # frekuensi dari suatu karakter karakter
        self.frekuensi = frekuensi

        # isi simbol pada node
        self.simbol = simbol

        # lokasi node tujuan pointer kiri
        self.kiri = kiri

        # lokasi node tujuan pointer kanan
        self.kanan = kanan

        # arah pohon
        self.huff = ''

    # membandingkan frekuensi yang ada pada kedua node
    def __lt__(self, nxt):
        return self.frekuensi < nxt.frekuensi

def printNodes(node, val = ""):
    newVal = val + str(node.huff)

    if(node.kiri):
        printNodes(node.kiri, newVal)
    if(node.kanan):
        printNodes(node.kanan, newVal)

    if(not (node.kiri or node.kanan)):
        print(f"{node.simbol} -> {newVal}")
```

Gambar 3.3 Kode Python untuk Huffman Coding(Bagian 1)

Dapat dilihat bahwa panjang kode binernya adalah 2320 dan memiliki kode biner

01010000010100100100111101001011010011000100000101001101010000010101001101001001000000100101110110000101
11011010110100100100000011000101100001011011100110011101110011011000010010000001001001011011100110010001
10111101101110011001010111001101101001011000010010000011001000110010110111001100111011000010110111000101
00000011010010110111001101001000000110110101100101011011100110011010000110001101110100011000010111010110
00010110111000100000011010110110010101101101101100101011100100110010001100101011010110111000010110000101101
11000100000010010010110111001100100011011110110111001100101011100110110100101100001001011100000100101
00011000010110110000101101011010000110000101100101000000111000101100001011010010110000100101110010000010010
10110001010111001100011011101100100111000110110000101100101101100010010000011001000110001011010110100001011
01100100011000010110100001100001011011000100000110101101100101100101100101110110010101100001011100110
11000010110000101101110001000000110010000101110011010000101110011011000010111001101100001011000010010011000
10100101110011011001010111011000110010101100111001110011001110110001011100100110000101101100100000101011
01110001000000110010001011011100110010000011001000010111001101100001011100110110000101100100000011000100
00010011000000110101001011100010000010000101110011010000101110011011000010111001101100001011000010010011000
01100001011100101110011000010000011001000010111001101100001011100110110000101100100000011000100000101110100
01100001011100101110011000010000011001000010111001101100001011100110110000101100100000011000100000101110101
10000100100000011000100110000101101110011001110011
00110110000100100000010010010110111001100100011011
11011011100110010101110011011010010110000100101110
01010011011011110110010101101101100001011100110111
01000110000100101110

```python
if __name__ == "__main__":
    # karakter yang ada
    karakter = [' ', 'A', 'D', 'H', 'I', 'K', 'L', 'M',
                'O', 'P', 'R', 'S','a', 'b', 'd', 'e',
                'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n',
                'o', 'p', 'r', 's', 't', 'y', '0', '1',
                '5', '7', '8', '.', ',', '/', '-']

    # frekuensi tiap karakter
    frekuensi = [34, 3, 1, 2, 4, 2, 1, 1, 1, 1, 1, 2, 51,
                 3, 11, 21, 11, 4, 12, 5, 11, 7, 9, 33, 9,
                 2, 6, 13, 10, 1, 1, 1, 1, 1, 1, 8, 2, 1, 2]

    # node yang ada
    Nodes = []

    # convert ke node untuk pohon
    for i in range(len(karakter)):
        heapq.heappush(Nodes, Node(frekuensi[i], karakter[i]))

    while len(Nodes) > 1:
        kiri = heapq.heappop(Nodes)
        kanan = heapq.heappop(Nodes)

        # untuk ke arah kiri beri simbol 0
        kiri.huff = 0

        # untuk ke arah kanan beri simbol 1
        kanan.huff = 1

        newNode = Node(kiri.frekuensi + kanan.frekuensi, kiri.simbol + kanan.simbol, kiri, kanan)

        heapq.heappush(Nodes, newNode)

    printNodes(Nodes[0])
```

Gambar 3.4 Kode Python untuk Huffman Coding(Bagian 2)

Kode *Python* tersebut menghasilkan output yang menunjukkan hasil pengkodean sebagai berikut

```
i -> 0000
5 -> 0001000
K -> 0001001
A -> 000101
l -> 00011
s -> 0010
. -> 00110
M -> 00111000
P -> 00111001
/ -> 00111010
1 -> 00111011
h -> 001111
n -> 010
  -> 011
- -> 1000000
D -> 10000010
L -> 10000011
p -> 1000010
y -> 10000110
0 -> 10000111
I -> 100010
7 -> 10001100
```

Gambar 3.5 Hasil Huffman Coding (Bagian 1)

```
8 -> 10001101
S -> 1000111
o -> 10010
m -> 10011
H -> 1010000
, -> 1010001
R -> 10100100
O -> 10100101
b -> 1010011
t -> 10101
e -> 1011
k -> 11000
j -> 110010
r -> 110011
d -> 11010
g -> 11011
a -> 111
```

Gambar 3.6 Hasil Huffman Coding (Bagian 2)

Dengan Kode Huffman, karakter-karakter tersebut dapat diubah menjadi

Karakter	Kode	Karakter	Kode	Karakter	Kode
Spasi	011	U	-	p	10000010
A	000101	V	-	q	-
B	-	W	-	r	110011
C	-	X	-	s	0010
D	10000010	Y	-	t	10101
E	-	Z	-	u	-
F	-	a	111	v	-
G	-	b	1010011	w	-
H	1010000	c	-	x	-
I	100010	d	11010	y	10000110
J	-	e	1011	z	-
K	0001001	f	-	0	10000111
L	10000011	g	11011	1	00111011
M	00111000	h	001111	2	-
N	-	i	0000	3	-
O	10100101	j	110010	4	-
P	00111001	k	11000	5	0001000
Q	-	l	00011	6	-

R	10100100	m	10011	7	10001100
S	1000111	n	010	8	10001101
T	-	o	10010	9	-
.	00110	,	1010001	/	0011110
-	1000000				

Tabel 3.2 Kode Huffman

Dari hasil pengkodean tersebut, jika diterapkan dengan cara yang sama seperti kode biner sebelumnya, akan menghasilkan bit dengan panjang

3x34 + 6x3 + 8x1 + 7x2 + 6x4 + 7x2 + 8x1 + 8x1 + 8x1 + 8x1 + 8x1 + 7x2 + 5x8 + 7x2 + 3x51 + 7x3 + 5x11 + 4x21 + 5x11 + 6x4 + 4x12 + 6x5 + 5x11 + 5x7 + 5x9 + 3x33 + 5x9 + 7x2 + 7x2 + 6x6 + 4x13 + 5x10 + 8x1 + 8x1 + 8x1 + 7x1 + 8x1 + 8x1 + 8x1 = 1258 bit.

C. Analisis Kapasitas

Dari perhitungan sebelumnya, bit biner menggunakan kode ASCII membutuhkan 2320 bit. Dengan menggunakan Kode Huffman, bit biner yang dibutuhkan adalah 1258 bit. Dari data ini, penyederhanaan yang dilakukan adalah

$$\frac{2320 - 1258}{2320} . 100\ \% = \ 45{,}776\ \%$$

Ini berarti bit dalam ASCII terkompresi sebesar 45,776 % jika menggunakan Kode Huffman. Ini berarti pengkodean dengan Kode Huffman sangat efektif karena hamper bisa mengompres data hampir setengahnya.

IV. KESIMPULAN

Kode Huffman digunakan untuk menggambarkan teks asli Proklamasi, yang panjangnya 290 karakter. Penelitian ini menunjukkan bahwa representasi biner Kode ASCII membutuhkan lebih banyak bit biner untuk menyimpan teks tersebut.

Dengan menggunakan Kode Huffman, panjang bit biner yang diperlukan hanya sekitar 54,224% dari panjang bit biner yang diperoleh dengan menggunakan Kode ASCII. Kode Huffman mampu mengompres data teks Proklamasi hingga sekitar 45,776%. Hasil ini menunjukkan bahwa, dengan memanfaatkan frekuensi munculnya karakter, Kode Huffman adalah metode kompresi yang efektif untuk mengurangi ukuran data teks.

Dalam penelitian ini, kami melihat bagaimana Kode Huffman dapat digunakan untuk mengelola data teks, terutama teks dengan pola kemunculan karakter tertentu. Karena efisiensi kompresinya yang tinggi, Kode Huffman adalah pilihan yang bagus untuk digunakan dalam kompresi, transmisi, dan penyimpanan data di komputer kontemporer.

V. LAMPIRAN

Kode program yang digunakan untuk penelitian dapat dilihat pada https://github.com/FrancescoMichael/Makalah-Matematika-Diskrit.git .

VI. UCAPAN TERIMA KASIH

Puji Tuhan kepada Tuhan Yang Maha Esa karena atas Rahmat dan berkat-Nya, makalah berjudul “” dapat diselesaikan dengan baik. Terima kasih juga diberikan kepada dosen pengampu mata kuliah IF2120 Matematika Diskrit Dr. Nur Ulfa Maulidevi atas bimbingan selama perkuliahan.

DAFTAR PUSTAKA

[1] Munir, RInaldi. 2020. Matematika Diskrit Revisi Ketujuh. Bandung: Penerbit Informatika

[2] Munir, Rinaldi, Pohon (Bagian 1 &2), (2022), diakses di https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Pohon2020-Bag1.pdf and https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2021-2022/Pohon2021-Bag2.pdf

[3] Yağmur Çiğdem Aktaş (2021), Huffman Encoding & Python Implementation, diakses di Medium https://towardsdatascience.com/huffman-encoding-pythonimplementation-8448c3654328

[4] Proklamasi Kemerdekaan Indonesia diakses di Proklamasi Kemerdekaan Indonesia (menlhk.go.id)

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, 10 Desember 2023

Francesco Michael Kusuma 13522038