IMPLEMENTASI DAN ANALISIS PERBANDINGAN ANTARA PENGKODEAN LZ78 DAN SHANNON FANO PADA KOMPRESI DATA TEKS

Nita Christina Saputro, Sri Suwarno, R.Gunawan Santosa
Jurusan Teknik Informatika. Fakultas Teknik
Universitas Kristen Duta Wacana, Yogyakarta

Abstrak:

Algoritma Shannon Fano merupakan algoritma kompresi data yang mengkodekan tiap karakter dengan menggunakan beberapa rangkaian bit. Pembentukan bit yang mewakili masing-masing karakter dibuat berdasarkan frekuensi kemunculan tiap karakter. Sedangkan algoritma LZ78 merupakan algoritma kompresi data yang menggunakan kamus yang dibentuk setiap ada 1 input karakter baru. Output dari algoritma LZ78 adalah berupa serangkaian token. Melalui analisis yang dilakukan pada penelitian ini diketahui bahwa algoritma Shannon Fano menghasilkan rasio dan waktu kompresi yang lebih baik daripada algoritma LZ78 pada data teks yang memiliki frekuensi kemunculan karakter yang merata. Sedangkan algoritma LZ78 menghasilkan rasio kompresi yang lebih baik pada data teks di mana frekuensi kemunculan tiap-tiap karakter besar.

Kata Kunci: *data compression, shannon fano, lz78*

1. Pendahuluan

Ukuran *file* yang semakin besar menuntut para pemakai komputer untuk mencari berbagai macam cara agar dapat menyimpan sejumlah besar *file* dalam media penyimpanan yang terbatas. Untuk itu, ukuran dari *file* harus dimampatkan agar ukurannya menjadi lebih kecil. Teknik memampatkan data ini disebut dengan teknik kompresi data.

Dari berbagai macam algoritma kompresi data yang ada, penulis ingin membandingkan keefektifan penggunaan algoritma Shannon Fano dan LZ78 pada data teks. Melalui penelitian ini diharapkan dapat ditemukan kelebihan dan kekurangan dari masing-masing algoritma tersebut di atas dalam mengkompresi data teks.

Penelitian ini akan dilakukan terhadap file teks berekstensi .txt dengan menggunakan sistem yang dibuat. File teks akan dikompresi dengan menggunakan algoritma LZ78 dan Shannon Fano untuk kemudian dibandingkan rasio kompresi dan kecepatan kompresi. File teks yang telah dikompresi kemudian akan didekompresi untuk memastikan tidak ada data yang hilang.

2. Landasan Teori

2.1 Algoritma Shannon Fano

Menurut Salomon (2007), kompresi data adalah proses pengubahan serangkaian data input menjadi data output yang mempunyai ukuran lebih kecil. Algoritma Shannon Fano ditemukan dan dikembangkan oleh Claude Shannon dan Robert Fano. Algoritma Shannon Fano mengkodekan tiap karakter yang ada dalam serangkaian data input dengan menggunakan rangkaian beberapa bit, di mana karakter yang sering muncul dikodekan dengan rangkaian bit yang lebih pendek dibandingkan karakter yang jarang muncul.

Berikut ini adalah langkah-langkah kompresi dengan menggunakan algoritma Shannon Fano:

1. Buat pohon Shannon dengan cara membaca semua karakter di dalam teks untuk menghitung probabilitas kemunculan tiap karakter.
2. Karakter-karakter ini kemudian disusun dari yang mempunyai probabilitas kemunculan paling besar ke karakter yang mempunyai probabilitas kemunculan paling kecil.
3. Bagi dua serangkaian karakter ini menjadi 2 subset yang mempunyai jumlah probabilitas yang sama atau hampir sama.
4. Semua simbol dalam subset pertama diberi kode 0 sedangkan simbol pada subset lainnya diberi kode 1.
5. Ketika pada sebuah subset hanya terdapat 2 macam simbol, kode mereka dibedakan dengan cara memberikan 1 bit lagi ke masing-masing simbol.
6. Lakukan langkah nomor 3, 4, dan 5 pada tiap-tiap subset secara rekursif sampai tidak ada lagi subset yang tersisa.

Dan berikut ini adalah algoritma untuk melakukan dekompresi pada serangkaian kode-kode tersebut:

1 Baca bit pertama dari serangkaian kode yang dihasilkan.

2 Jika bit tersebut ada dalam pohon Shannon, maka bit tersebut diterjemahkan menjadi karakter yang sesuai dengan bit tersebut.

3 Jika bit tersebut tidak ada dalam pohon Shannon, gabungkan bit tersebut dengan bit selanjutnya dalam rangkaian kode, cocokkan dengan tabel hasil pengkodean.

4 Lakukan langkah 3 sampai ada rangkaian bit yang cocok dengan pohon Shannon, terjemahkan rangkaian bit tersebut menjadi karakter yang sesuai.

5 Baca bit selanjutnya dan ulangi langkah 2, 3 dan 4 sampai rangkaian kode habis.

2.2 Algoritma LZ78

LZ78 termasuk dalam algoritma yang menggunakan metode kamus. Algoritma ini mengubah tiap karakter data input menjadi sebuah token menggunakan sebuah kamus. Kamus terdiri dari 2 buah *field*. *Field* pertama digunakan sebagai index dari karakter, *field* kedua digunakan untuk karakter.

Token berisikan 2 buah *field*, *field* pertama adalah sebuah pointer yang mengacu pada index kamus, *field* kedua berisi kode dari karakter. Berikut ini adalah langkah langkah pengkodean dengan menggunakan algoritma LZ78:

1. Kamus selalu dimulai dengan index 0 dan karakter *NULL*
2. Baca karakter pertama dalam teks, misal P.
3. Jika karakter yang ditunjuk belum ada pada kamus, maka:
   a. Masukkan P di *field* kedua dalam kamus dan beri index.
   b. *Field* pertama pada token diisi angka 0, sedangkan *field* kedua pada token diisi karakter “P”.
4. Jika karakter yang ditunjuk sudah ada pada kamus, maka:
   a. Baca karakter selanjutnya, misal C.
   b. Cari apakah dalam kamus sudah ada karakter PC.
   c. Jika rangkaian karakter tersebut sudah ada, maka lakukan langkah (a) dan (b) sampai rangkaian karakter yang ditunjuk tidak ada dalam kamus.
   d. Jika rangkaian karakter tersebut tidak ada dalam kamus, maka:
      - Masukkan rangkaian karakter di field kedua dalam kamus dan beri index.
      - Field pertama pada token diisi index kamus tempat rangkaian karakter sebelum karakter terakhir berada, sedangkan field kedua diisi karakter terakhir dalam rangkaian karakter tersebut.
5. Baca karakter selanjutnya dan lakukan langkah 3 dan 4 sampai pointer berada pada karakter terakhir dalam string.

Berikut ini adalah algoritma untuk melakukan dekompresi pada serangkaian token-token tersebut:

1. Baca token pertama dari serangkaian token-token yang ada.
2. Cocokkan token tersebut dengan kamus yang telah terbentuk.
3. Ubah token menjadi karakter yang sesuai.
4. Baca token selanjutnya dan lakukan langkah 2 dan 3 sampai token habis.

2.3 Pengukuran Performa

Menurut Sayood (2006), sebuah algoritma kompresi dapat dievaluasi dengan berbagai macam tolak ukur, antara lain:

- Kompleksitas algoritma
- Memori yang diperlukan untuk mengimplementasikan algoritma tersebut
- Kecepatan performa algoritma tersebut
- Banyaknya data yang dapat dikompresi
- Kemiripan data asli dengan data hasil rekonstruksi

Cara yang dapat digunakan untuk mengukur seberapa baik sebuah algoritma kompresi bekerja adalah dengan menghitung rasio kompresinya. Menurut Sayood (2006), rasio kompresi adalah perbandingan antara jumlah bit yang diperlukan untuk merepresentasikan data sebelum dikompresi dengan jumlah bit yang diperlukan untuk merepresentasikan data setelah dikompresi.

Rasio kompresi sebuah data dapat juga direpresentasikan dengan menggunakan rumus pada Gambar 2.3.1 sebagai berikut:

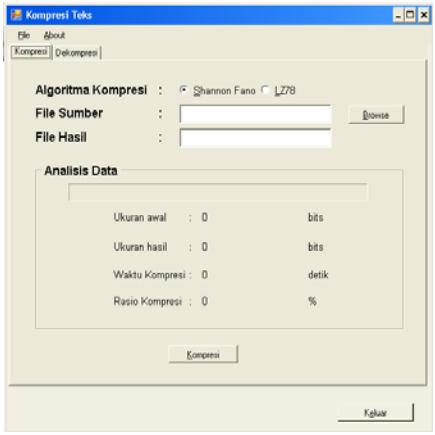
$$\text{Rasio kompresi} = 100\% - \left(\frac{\textit{output stream}}{\textit{input stream}} \times 100 \right)$$

Gambar 1 Rumus Rasio Kompresi

3. Pembahasan

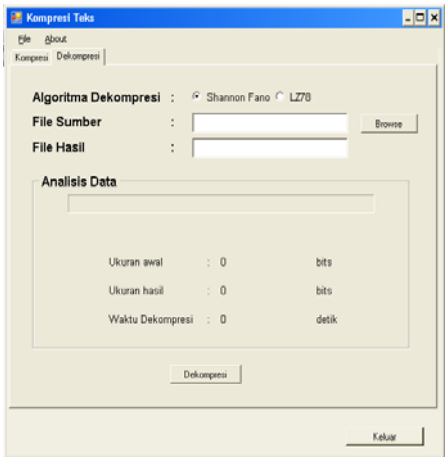
3.1 Antar Muka Program

Dalam sistem yang dibuat, terdapat dua buah form. Form pertama yaitu form untuk melakukan proses kompresi dan dekompresi sedangkan form kedua adalah form About. Form ini berisi identitas dari pembuat sistem. Form pertama terdiri dari dua buah tab. Tab yang pertama, sebagaimana terlihat dalam Gambar 2 adalah tab untuk melakukan proses kompresi. Berikut ini adalah hasil *screenshot* dari sistem yang dibuat:



Gambar 2 Tab Proses Kompresi

Tab kedua, sebagaimana terlihat dalam Gambar 3 adalah tab untuk melakukan proses dekompresi



Gambar 3 Tab Proses Dekompresi

Prosedur untuk melakukan proses dekompresi samap seperti prosedur untuk melakukan proses kompresi.

3.2 Analisis Sistem

Pada penelitian ini, hasil yang akan dibandingkan meliputi waktu kompresi dan dekompresi serta rasio kompresi. Pada saat proses pengujian, program akan berjalan sendiri tanpa ada *background* program lain yang berjalan. Sedangkan yang menjadi objek uji coba dari sistem yang dibuat adalah:

1. *Canterbury Corpus,* yaitu sekumpulan file teks yang sudah secara luas digunakan untuk membandingkan algoritma kompresi data. *Canterbury Corpus* dikembangkan pada tahun 1977

sebagai versi baru dari *Calgary Corpus.* Dalam *Canterbury Corpus* terdapat 11 file seperti terlihat dalam Tabel 1 berikut ini:

Tabel 1 File dalam *Canterbury Corpus*

Nama File	Kategori	Ukuran File (Bytes)
alice29.txt	English text	152089
asyoulik.txt	Shakespeare	125179
cp.html	HTML source	24603
fields.c	C source	11150
grammar.lsp	LISP source	3721
kennedy.xls	Excel spreadsheet	1029744
Icet10.txt	Technical writing	426754
plrabn12.txt	Poetry	481861
ptt5	CCITT test set	513216
sum	SPARC Executable	38240
xargs.1	GNU manual page	4227

Berhubung sistem yang dibuat hanya dapat menerima input berupa file berekstensi .txt, maka hanya akan dipakai 4 buah file dalam *Canterbury Corpus* sebagai objek uji coba. Yaitu file alice29.txt, asyoulik.txt, Icet10.txt, dan plrabn12.txt. *File* teks ini dapat di-*download* di alamat http://corpus.canterbury.ac.nz/resources/cantrbry.zip.

2. *Artificial Corpus,* yaitu sekumpulan file berekstensi .txt yang mempunyai suatu kondisi khusus, sehingga hasil kompresi bisa memunculkan hasil yang terburuk *(worst case scenario)* karena dalam file tersebut hanya terdapat sedikit atau bahkan sama sekali tidak ada pengulangan karakter dan file dengan ukuran yang sangat kecil. File ini dapat di-*download* di alamat http://corpus.canterbury.ac.nz/resources/artificl.zip. Artificial Corpus terdiri dari 4 buah file seperti tampak dalam Tabel 2 berikut ini:

Tabel 2 File dalam *Artificial Corpus*

Nama File	Keterangan	Ukuran File (Bytes)
a.txt	Hanya berisi satu karakter 'a'	1
aaa.txt	Karakter 'a' diulangi sebanyak 100000 kali	100000
alphabet.txt	Mengulangi karakter alfabet yang ada hingga mencapai 100000 karakter	100000
random.txt	100000 karakter yang dipilih secara acak dari [a-z|A-Z|0-9||]	100000

Perbandingan hasil dan rasio kompresi dengan menggunakan algoritma Shannon Fano dan LZ78 terhadap 4 buah file teks yang terdapat dalam *Canterbury Corpus* dapat dilihat dalam Tabel 3 berikut ini:

Tabel 3 Perbandingan Hasil dan Rasio Kompresi *Canterbury Corpus*

Nama File	Asli (bit)	Shannon Fano		LZ78	
		Hasil (bit)	Rasio (%)	Hasil (bit)	Rasio (%)
alice29.txt	1216712	705536	42.013	814824	33.031
asyoulik.txt	1001432	609632	39.124	711608	28.941
Icet10.txt	3414032	2011168	41.091	2122160	37.840
plrabn12.txt	3854888	2208632	42.706	2504016	35.043

Dari Tabel 4 dapat dilihat bahwa secara keseluruhan, algoritma Shannon Fano memberikan rasio kompresi yang lebih tinggi dibandingkan algoritma LZ78.

Tabel 4 Perbandingan Waktu Kompresi dan Dekompresi *Canterbury Corpus*

Nama File	Shannon Fano		LZ78	
	Waktu Kompresi (detik)	Waktu Dekompresi (detik)	Waktu Kompresi (detik)	Waktu Dekompresi (detik)
alice29.txt	0.464	0.033	56.215	118.711
asyoulik.txt	0.355	0.029	39.933	70.593
Icet10.txt	1.363	0.167	398.2	611.393
plrabn12.txt	1.556	0.091	557.277	879.13

Dari Tabel 4 terlihat bahwa proses kompresi dan dekompresi data dengan menggunakan Algoritma Shannon Fano membutuhkan waktu yang jauh lebih cepat dibandingkan proses kompresi data dengan menggunakan algoritma LZ78. Algoritma LZ78 membutuhkan waktu kompresi yang lebih lama karena untuk setiap 1 input karakter, LZ78 akan mencarinya di dalam kamus. Proses pencarian ini menyebabkan waktu kompresi dengan menggunakan algoritma LZ78 relatif lebih lama.

Tabel 5 Perbandingan Hasil dan Rasio Kompresi *Artificial Corpus*

Nama File	Sumber (bit)	Shannon Fano		LZ78	
		Kompresi		Kompresi	
		Hasil (bit)	Rasio (%)	Hasil (bit)	Rasio (%)
a.txt	8	88	-1000	96	-1100
aaa.txt	800000	100080	87.49	10744	98.657
alphabet.txt	800000	477528	40.309	59208	92.599
random.txt	800000	602760	24.655	922616	-15.327

Hasil penelitian terhadap rasio kompresi untuk kedua algoritma ini dapat dilihat pda table 5. Rasio kompresi menunjukkan bilangan negatif apabila proses kompresi atau pemampatan data gagal. Dalam artian, ukuran dari *file* teks tidak lebih kecil melainkan semakin besar. Algoritma LZ78 tampak memberikan performa yang baik pada *file* yang memiliki banyak pengulangan karakter. Pada *file* a.txt kedua algoritma memberikan rasio kompresi berupa bilangan negatif, hal ini dikarenakan *file* hasil kompresi membutuhkan ruang lebih banyak untuk

menyimpan data-data yang diperlukan untuk melakukan proses dekompresi daripada data itu sendiri. Sedang pada file random.txt, rasio kompresi dengan menggunakan algoritma LZ78 menunjukkan bilangan negatif, ini berarti proses kompresi gagal, ukuran *file* justru membengkak. Hal ini dikarenakan pada *file* random.txt, jenis karakter yang digunakan lebih banyak dengan tingkat pengulangan karakter yang lebih sedikit, sehingga node yang tebentuk semakin banyak dan menyebabkan ukuran *file* membengkak.

Tabel 6 Perbandingan Waktu Kompresi dan Dekompresi File dalam *Artificial Corpus*

Nama File	Shannon Fano		LZ78	
	Waktu Kompresi (detik)	Waktu Dekompresi (detik)	Waktu Kompresi (detik)	Waktu Dekompresi (detik)
a.txt	0.055	0.003	0.001	0.012
aaa.txt	0.052	0.007	0.366	0.66
alphabet.txt	0.273	0.019	0.548	3.348
random.txt	0.354	0.021	85.228	108.274

Dari Tabel 6 terlihat bahwa secara garis besar, proses kompresi data dengan menggunakan algoritma Shannon Fano membutuhkan waktu yang lebih singkat daripada proses kompresi data dengan menggunakan algoritma LZ78. Tapi tampak ada pengecualian terhadap file teks a.txt yang di dalamnya hanya terdapat satu karakter tunggal karena pada algoritma LZ78, proses pencarian lebih singkat daripada algoritma Shannon Fano.

5. Kesimpulan

Kesimpulan yang dapat ditarik setelah dilakukan uji coba terhadap sistem yang dapat melakukan proses kompresi dan dekompresi data dengan menggunakan algoritma Shannon Fano dan LZ78 pada sejumlah data teks adalah:

- Untuk file teks yang memiliki jumlah jenis karakter yang lebih banyak dengan tingkat pengulangan karakter yang lebih kecil, algoritma Shannon Fano memberikan rasio kompresi yang lebih tinggi daripada algoritma LZ78.
- Untuk file teks yang memiliki jumlah jenis karakter yang lebih sedikit dengan tingkat pengulangan karakter yang lebih tinggi, algoritma LZ78 memberikan rasio kompresi yang lebih tinggi daripada algoritma Shannon Fano
- Algoritma Shannon Fano membutuhkan waktu kompresi dan dekompresi yang lebih sedikit dibandingkan algoritma LZ78.

- Algoritma Shannon Fano juga dapat mengkompresi file gambar dengan baik.

Algoritma LZ78 tidak dapat mengkompresi file gambar dengan baik. Hal ini terjadi karena pada file gambar terdapat karakter *nothing.*

Daftar Pustaka

[1] Bell, T.C., Whitten, I., Cleary, J. (1990). *Text Compression.* New Jersey: Prentice Hall, Inc.

[2] Kurniawan, B., Neward, T. (2002). *VB.NET Core Classes In A Nutshell.* Cambridge: O'Reilly Media, Inc.

[3] Nelson, M. (1996). *The Data Compression Book 2nd Ed.* New York: M & T Books.

[4] Rickyanto, I. (2003). *Membuat Aplikasi Windows dengan Visual Basic.NET.* Jakarta: PT Elex Media Komputindo.

[5] Roman, S., Petrusha, R., Lomax, P. (2001). *VB.NET Language In A Nutshell 1st Ed.* Cambridge: O'Reilly Media, Inc.

[6] Salomon, D. (2007). *Data Compression The Complete Reference 4th Ed.* London: Springer.

[7] Sayood, K. (2006). *Introduction To Data Compression 3rd Ed.* San Fransisco: Morgan Kaufmann.