Pengaplikasian Algoritma *Knuth-Morris-Pratt* dalam Teknik Kompresi Data

I Nyoman Prama Pradnyana - 13509032
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
prama.pradnyana@itb.ac.id

***Abstract*—Dewasa ini, penyimpanan data yang memakan ruang memori yang cukup besar sudah menjadi permasalahan umum. Salah satu metode yang digunakan untuk menangani permasalahan tersebut adalah teknik kompresi data. Teknik kompresi data yang paling umum dilakukan adalah kompresi teks dimana teks akan dimampatkan sehingga ukuran teks menjadi lebih kecil. Terdapat banyak algoritma yang menangani teknik kompresi ini, namun dalam makalah ini akan dicoba mengulas algoritma *Knuth-Morris-Pratt* yang diaplikasikan ke dalam teknik kompresi teks. Metodenya adalah algoritma *Knuth-Morris-Pratt* akan mengambil *pattern* terdepan pada suatu teks dan melakukan pencarian *pattern* pada teks tersebut. Jika terdapat *pattern* yang ditulis berulang dalam teks tersebut, akan disimpan dan diberikan *flag*. *Pattern* pada teks tersebut nantinya akan digantikan oleh *flag* tersebut. Hasilnya, ukuran teks akan berkurang karena *pattern* pada teks diganti dengan *flag*. Disamping membantu dalam teknik kompresi, algoritma *Knuth-Morris-Pratt* memiliki keunggulan tersendiri dari segi kecepatan pencarian kata sehingga teknik kompresi nantinya bisa menjadi lebih cepat.**

Index Terms- Knuth-Morris-Pratt, Pattern, Flag

I. PENDAHULUAN

Di masa perkembangan teknologi ini, penggunaan media pengolahan data sudah menjadi kebutuhan bagi berbagai kalangan masyarakat. Tidak hanya pegawai atau mahasiswa yang menggunakan media pengolahan data ini, pelajar sekolah dasarpun sudah mulai aktif menggunakan media pengolahan data ini. Salah satu yang paling sering dilakukan dalam media pengolahan data adalah pembuatan dokumen.

Pada awalnya, pengolahan data seperti pembuatan dokumen tidak menjadi permasalahan. Namun seiring dengan kebutuhan pengolahan data, timbul permasalahan baru yaitu mengenai penggunaan memori penyimpan data. Saat ini sudah banyak data yang memiliki ukuran yang sangat besar bahkan pembuatan dokumenpun dapat menghabiskan memori ketika dokumen tersebut memiliki ukuran yang besar.

Untuk menangani permasalahan tersebut, dicari berbagai metode untuk dapat melakukan pemangkasan ukuran data. Salah satu metode yang hingga saat ini sering digunakan adalah teknik kompresi data. Teknik kompresi bertujuan mengurangi ukuran data dengan melakukan modofikasi terhadap data tersebut. Modifikasi data ini dapat berupa pemampatan data yang bersifat redundan, pengurangan volume, ukuran dll. Namun, meskipun dilakukan modifikasi, teknik kompresi tidak melakukan penghancuran data yang dapat menyebabkan data rusak. Nantinya hasil ini dapat dikembalikan menjadi data asli ketika hendak dipakai.

Salah satu bentuk teknik kompresi yang akan dibahas dalam makalah ini adalah teknik kompresi teks. Kompresi teks yang akan dilakukan adalah menggabungkan kata-kata yang sama sehingga dapat mengurangi panjang teks. Dengan metode tersebut, ukuran teks akan berkurang. Misalkan teks "Adi mahasiswa Teknik Informatika. Di Teknik Informatika, Adi memiliki banyak teman." jika dilakukan kompresi, kata Adi, Teknik, dan Informatika akan digabung. Hasilnya adalah teks yang lebih pendek dan ukuran teks yang lebih kecil.

Pada dasarnya terdapat banyak cara melakukan kompresi teks. Namun dalam makalah ini akan digunakan satu algoritma yang dicoba diaplikasikan untuk membantu teknik kompresi data. Algoritma yang akan digunakan adalah algoritma *Knuth-Morris-Pratt* yaitu salah satu algoritma popular dalam pencocokan kata. Algoritma ini memiliki keunggulan dari segi kecepatan melakukan pencarian kata dalam teks sehingga nantinya proses kompresi menjadi lebih cepat.

Melalui makalah ini diharapkan dapat memberikan gambaran penggunaan algoritma *Knuth-Morris-Pratt* yang diaplikasikan pada teknik kompresi. Disamping itu melalui makalah ini diharapkan pengaplikasian algoritma ini dapat terus dikembangkan hingga bisa menjadi metode pilihan karena saat ini masih belum ada sumber yang memanfaatkan algoritma *Knuth-Morris-Pratt* ke dalam teknik kompresi.

II. DASAR TEORI

2.1 Penggunaan Bit Data

Pada dasarnya di setiap aplikasi penyimpan teks melakukan pengukuran teks berbeda-beda. Ukuran tersbut biasanya tergantung pada ukuran huruf, jenis huruf dan tambahan lainnya dalam teks. Contoh dalam format (.txt), untuk setiap karakter(jenis apapun dan ukuran apapun) pada file teks diberikan ukuran 1bytes. Sedangkan pada *Ms. Word* setiap karakter (*Calibri body* 11*pt*) pada file diberikan ukuran sebesar 22bytes.

Pada (.txt) file teks, berapapun ukuran teks, dan jenis apapun, untuk setiap satu karakter akan diberikan ukuran sebesar 1bytes. Karena kestabilan dari (.txt) maka pada makalah ini akan cenderung membahas dan mengambil contoh dari (.txt) meskipun pada prinsipnya bisa dilakukan pada media teks lain seperti *Ms. Word* dll.

2.2 Kompresi Data

Kompresi data adalah proses mengkodekan informasi menggunakan bit atau *information-bearing* unit yang lain yang lebih rendah daripada representasi data yang tidak terkodekan dengan suatu sistem encoding tertentu [DIG09]. Tujuan dari kompresi data adalah untuk merepresentasikan suatu data digital dengan sesedikit mungkin bit, tetapi tetap mempertahankan kebutuhan minimum untuk membentuk kembali data aslinya. Dalam teknik kompresi tidak disarankan untuk melakukan sembarang jenis kompresi dimana data hasil kompresi menjadi tidak dapat diubah kembali ke dalam bentuk aslinya. Namun demikian dalam teknik kompresi, terdapat beberapa hal yang menyebabkan pembentukan data hasil kompresi tidak persis sama dengan data awal seperti pada kompresi gambar dll.

Untuk membuat ukuran suatu data menjadi lebih kecil daripada ukuran data asli, diperlukan tahapan-tahapan untuk mengolah data tersebut. Salah satu tahapan tersebut adalah dengan menggunakan algoritma kompresi. Algoritma kompresi ini nantinya yang mengatur segala pengolahan data hingga dicapai hasil akhir yaitu data dengan ukuran yang lebih kecil.

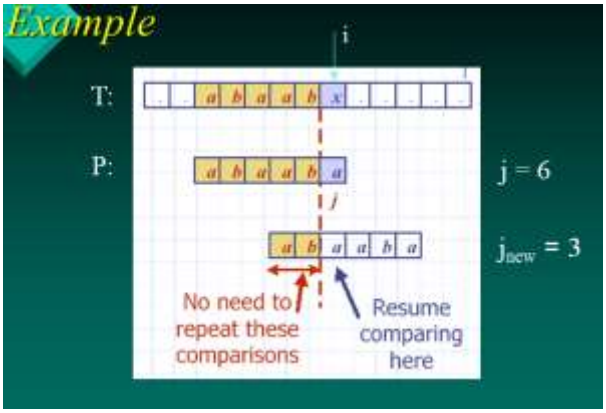
Adapun algoritma kompresi dibagi menjadi dua golongan yaitu kompresi *Lossy* dan kompresi *Lossless*.

Kompresi *Lossy* adalah algoritma yang tidak dimungkinkan untuk membentuk data asli yang tepat sama dari data yang sudah dikompresi. Dengan kata lain, terdapat beberapa detail data yang hilang selama proses kompresi. Contoh penggunaan algoritma *Lossy* pada teknik kompresi data gambar, suara dan video. Teknik kompresi dari gambar, suara dan video cenderung menghilangkan beberapa detail seperti detail warna yang tidak bisa dikembalikan persis sama dengan data awal.

Kompresi *Lossless* adalah algoritma yang dimungkinkan untuk membentuk data asli yang tepat sama dari data yang sudah dikompresi. Tidak ada informasi yang hilang selama proses kompresi dan dekompresi. Teknik ini digunakan jika data tersebut sangat penting, jadi tidak dimungkinkan untuk menghilangkan beberapa detail. Pada umunya teknik kompresi teks bersifat *Lossles* karena pada prinsipnya teknik kompresi teks menggunakan metode pemampatan ataupun pengubahan data menjadi kode ASCII. Dengan demikian sangat kecil dalam teknik kompresi teks menghilangkan detail data.

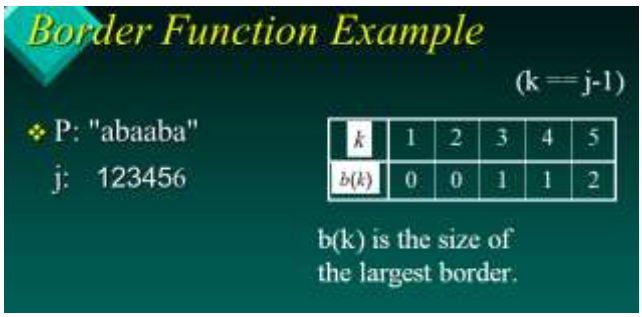
2.3 Algoritma *Knuth-Morris-Pratt*

Algoritma *Knuth-Morris-Pratt* adalah salah satu algoritma pencocokan kata dimana algoritma ini mengadopsi prinsip kerja algoritma *brute force*. Meskipun melakukan adopsi dari algoritma *brute force*, pencocokan kata dengan menggunakan algoritma ini jauh lebih cepat dibandingkan dengan pencarian menggunakan algoritma *brute force*. Perbedannya dengan algoritma *brute force* adalah algoritma *brute force* melakukan perpindahan pengecekan untuk setiap karakter dari teks, sedangkan algoritma *Knuth-Morris-Pratt* melakukan perpindahan berdasarkan informasi *pattern* dan tidak per-karakter sehingga pegeseran yang dilakukan algoritma *Knuth-Morris-Pratt* jauh lebih panjang dibandingkan algoritma *brute force*. Hal inilah yang membuat waktu pencarian dengan algoritma *Knuth-Morris-Pratt* jauh lebih cepat. Kompleksitas algoritma *Knuth-Morris-Pratt* adalah O(m+n) dimana O(m) adalah kompleksitas informasi data perpindahan, dan O(n) adalah kompleksitas algoritmanya sendiri.



Gambar 2.1 Penggunaan Algoritma Knuth-Morris-Pratt

Dalam prosesnya, algoritma *Knuth-Morris-Pratt* memiliki fungsi pembantu yaitu Fungsi Pinggiran. Fungsi Pinggiran ini nantinya menjadi literature dari banyak perpindahan yang akan dilakukan oleh algoritma *Knuth-Morris-Pratt*. Pada prinsipnya Fungsi Pinggiran hanya bergantung pada karakter-karakter di dalam pattern dan bukan pada karakter-karakter pada teks. Oleh karena itu fungsi ini dapat dilakukan sebelum dilakukan pencocokan kata.



Gambar 2.2 Fungsi Pinggiran

III. METODE

3.1 Persiapan Fungsi Pinggiran

Langkah pertama akan dibuat Fungsi Pinggiran yang akan digunakan dalam algoritma *Knuth-Morris-Pratt*. Metode pembuatannya adalah dengan melakukan pencarian secara bertahap dari pattern yang ada. Pengecekan dilakukan terhadap setiap karakter pada pattern tersebut. Pengecekan dilakukan dengan menghitung panjang maksimal karakter pinggiran yang sama. Hasil tersebut dimasukkan ke dalam array (fungsinya seperti tabel).

Ketika melakukan pengecekan terhadap *pattern*, terdapat hal penting yang perlu diperhatikan. Pertama, untuk karakter setiap pertama dari *pattern*, dikatakan tidak memiliki nilai panjang karakter, dengan kata lain untuk k=1, nilai panjang border dipastikan 0. Sedangkan yang kedua, dari jumlah pengecekan juga perlu diperhatikan dimana jumlah pengecekan dilakukan sebanyak (n-1) karakter dimana n merupakan banyak karakter pada pattern.

Pseudocode dari Fungsi Pinggiran

```
procedure HitungPinggiran(input m : integer,
P : array[1..m] of char, output b :
array[1..m] of integer)
{Menghitung nilai b[1..m] untuk pattern
P[1..m]}

Deklarasi
k, i : integer

Algoritma
b[1]← 0
q ← 2
k ← 0
for q ← 2 to m do
  while ((k>0) and (P[q] ←P[k+1])) do
    k ←b[k]
  endwhile
  if P[q]=P[k+1] then
    k ← k+1
  endif
  b[q]=k
endfor
```

Contoh pencarian Fungsi Pinggiran

Misalkan pattern P = “abaaba”
maka :

Untuk k=1, yaitu “a” tidak memiliki karakter yang sama sehingga nilainya 0
Untuk k=2, yaitu “ab” tidak memiliki karakter yang sama sehingga nilainya 0
Untuk k=3, yaitu “aba” memiliki satu karakter yang sama yaitu “a” pada ujung pattern, sehingga nilainya 1
Untuk k=4, yaitu “abaa” juga memiliki satu karakter yang sama yaitu “a” pada ujung pattern, sehingga nilainya 1
Untuk k=5 yaitu “abaab” memiliki dua buah karakter yang sama yaitu “ab” pada ujung pattern sehingga nilainya 2.
Hasil ini bisa dilihat pada gambar 2.2 Fungsi Pinggiran

3.2 Persiapan Algoritma Knuth-Morris-Pratt

Langkah selanjutnya adalah dengan membuat algoritma Knuth-Morris-Pratt. Metode pembuatannya adalah dengan melakukan pengecekan antara pattern dengan teks yang ada. Teks akan diubah menjadi list of teks dan dilakukan pengecekan terhadap pattern. Jika terdapat kata pada teks yang sama dengan pattern, maka posisi kata tersebut akan disimpan oleh algoritma dalam indeks dimana indeks adalah list of integer. Algoritma kemudian melanjutkan pencarian hingga akhir teks. Jika menemukan lagi, maka indeks akan ditambah.

Pseudocode algoritma Knuth-Morris-Pratt

```
procedure KMPsearch (input m, n : integer,
input P : array[1..m] of char, input T :
array[1..n] of char, output idx : integer)
{Mencari kecocokan pattern P di dalam teks T
dengan algoritma Knuth-Morris-Pratt. Jika
ditemukan P di dalam T, lokasi awal kecocokan
disimpan di dalam peubah idx
Masukan : pattern P yang panjangnya m dan teks
T yang panjangnya n. Teks T direpresentasikan
sebagai string (array of character)
Keluaran : posisi awal kecocokan (idx). Jika
ditemukan, idx = -1
}

Deklarasi
i, j : integer
ketemu : boolean
b : array[1..m] of integer

procedure HitungPinggiran(input m : integer,
P : array[1..m] of char, output b :
array[1..m] of integer)
{menghitung nilai b[1..m] untuk pattern
P[1..m]}

Algoritma
HitungPinggiran(m, P, b)
j ← 0
i ←1
ketemu ←false
while(i ←n and not ketemu) do
  while((j > 0) and (P[j+1] ←T[i]) do
    j ←b[j]
  endwhile
  if P[j+1] = T[I] then
    j←j + 1
  endif
  if j = m then
    ketemu = true
  else
    i ←i + 1
  endif
endwhile
if ketemu then
  idx ←i-m+1
else
  idx = -1
endif
```

3.3 Persiapan Algoritma Kompresi

Langkah terakhir adalah pembuatan algoritma kompresi. Tujuan dari algoritma kompresi ini adalah memproses hasil dari algoritma *Knuth-Morris-Pratt*. Pada dasarnya prinsip algoritma ini adalah meng-*update* data pada file asli dimana file baru tersebut terdapat hasil dari algoritma *Knuth-Morris-Pratt*. Langkah selanjutnya adalah dengan menuliskan ulang data asli dengan data baru dimana seluruh kata yang sama dengan hasil dari algoritma Knuth-Morris-Pratt akan diganti dengan nilai flag yang telah ditentukan sebelumnya.

Metode yang dilakukan adalah melakukan pengubahan data teks pada file menjadi *array of string* agar bisa dilakukan pengecekan oleh algoritma *Knuth-Morris-Pratt*. Selanjutnya dilakukan pengecekan terhadap kata pertama dan menjadikan kata pertama tersebut sebuah *pattern*. *Pattern* tersebut kemudian dijalankan dengan menggunakan algoritma *Knuth-Morris-Pratt*. Jika tidak ada kata yang sama pada teks dengan *pattern* tersebut, maka *pattern* diganti dengan kata berikutnya. Jika pada teks terdapat kata yang sama dengan *pattern*, maka kata tersebut akan dimasukkan ke dalam *array* kata dan diberikan *flag*. *Flag* adalah indeks dimana kata yang sama dengan *pattern* tersebut berada. Misalkan terdapat *pattern* ”prama” dan pada teks diketahui terdapat kata “prama” pada posisi 4, 6 dan 10. Maka pada *array* kata akan ditambahkan kata “prama!1” dimana !1 merupakan *flag* sebagai penunjuk !1 adalah kata “prama”. *Array* kata ini nantinya akan ditulis pada baris pertama *file*.

Setelah dilakukan pengecekan untuk seluruh kata, maka selanjutnya adalah penulisan data baru pada *file*. Penulisan ini dimulai dari baris ke dua karena pada baris pertama terdapat informasi *array* kata. Penulisan pada baris ke dua adalah *array of string* tadi, namun yang berbeda di sini adalah jika kata pada *array of string* terdapat pada array kata, maka kata tersebut akan diganti dengan *flag* yang sesuai. Misalkan pada contoh kata “prama” di atas dimana terdapat pada string ke 4, 6 dan 10, maka penulisannya menjadi

```
Prama!1
Xxx xxx xxx !1 xxx !1 xxx xxx xxx !1

dimana xxx adalah kata bukan prama.
```

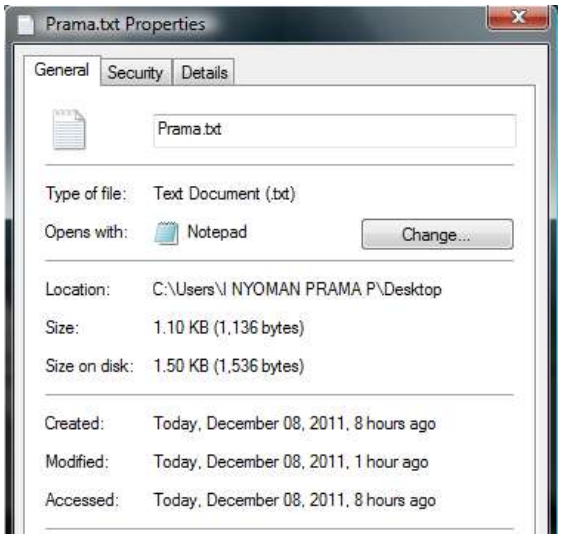
Dengan demikian, maka dapat dilakukan penghematan penggunaan kata dapat dilakukan penghematan penggunaan kata.

IV. PEMBAHASAN

3.1 Hasil

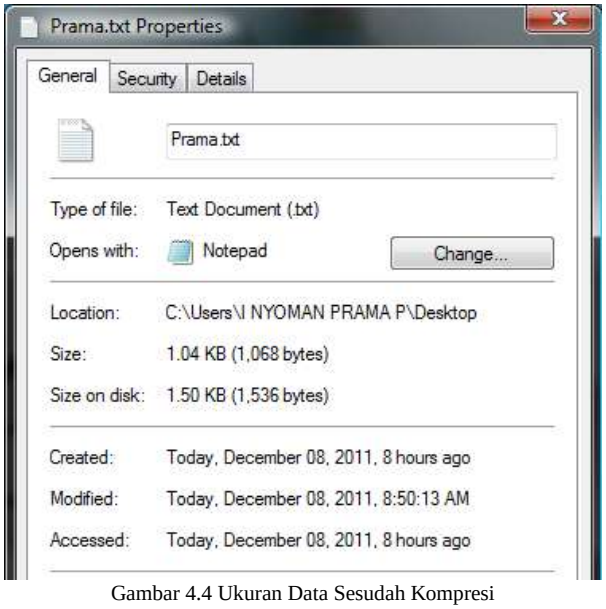
Berikut adalah hasil kompresi yang dilakukan terhadap bagian abstrak dari makalah ini.

Gambar 4.1 Sebelum Kompresi



Gambar 4.2 Ukuran Data Sebelum Kompresi

Gambar 4.3 Sesudah Kompresi



Gambar 4.4 Ukuran Data Sesudah Kompresi

3.2 Analisis Hasil

Pada bagian hasil di atas, terlihat bahwa terjadi pengurangan ukuran data yang awalnya berukuran 1.136 bytes, setelah dilakukan proses kompresi ukuran data menjadi 1.068 bytes. Perubahan yang terjadi ini akibat adanya pengelompokan kata yang sama pada bagian *array* kata. Dengan penggunaan *array* kata dapat dilakukan penghematan sebesar

Penghematan(*pattern*) = L(n-1)-x(n+1)
Dimana
L = panjang dari *pattern*, misalkan "prama" L=5
n = banyak *pattern* pada teks
x = panjang flag, misalkan "!1" x=2

Jika dilihat dari persamaan tersebut, tentu akan bermasalah ketika nilai L lebih kecil atau sama dengan x. Oleh karena itu pada algoritma *Knuth-Moris-Pratt* diberikan batasan pengecekan dimana untuk pattern berupa karakter dengan panjang 1,2 dan 3 tidak dilakukan pengecekan. Mengapa hanya 1, 2 dan 3 saja, hal ini dikarenakan panjang flag diasumsikan tidak lebih dari ratusan. Jadi untuk *pattern* "a", "an" dll harus dilewati tanpa dilakukan pengecekan lagi. Dengan demikian tidak masalah L lebih kecil atau sama dengan x bisa ditanggulangi. Terlebih lagi L cenderung jauh lebih banyak dibanding x dengan kata lain, terdapat jumlah L>3 yang lebih banyak daripada L<=3. Jadi dengan metode ini, masalah diatas dapat ditangani.

Pada gambar juga terlihat pengurangan memori cenderung sedikit. Meskipun perubahan masih dikatakan relatif sedikit, namun jika digunakan untuk data yang ukurannya besar, metode ini bisa sangat membantu. Dengan kata lain, semakin banyak data yang ada, maka peluang terjadinya pemampatan akan semakin besar. Tentu saja jika tidak dilakukan pemampatan, ukuran data akan tetap. Jadi yang mungkin dalam hal ini adalah pengurangan ukuran data atau ukuran data yag tetap.

Mengenai teknik dekompresi masih bisa dilakukan untuk mengembalikan data kembali seperti semula. Gamabaran teknik dekompresi adalah hasil dari kompresi tersebut akan terdapat dua bagian yaitu bagia *array* kata dan data yang sudah diberikan *flag*. Nantinya pada bagian *array* kata dan data akan disimpan kembali ke dalam *array of string*, kemudian dilakukan penghapusan data pada file. Setelah dilakukan penghapusan, akan ditulis kembali data pada *array of string* dimana dilakukan penggantian flag pada *array of string* dengan nilai *flag* pada *array kata*. Dengan demikian, tidak ada data yang akan hilang dan data hasil dekompresipun akan sama dengan data aslinya.

3.3 Analisis Penggunaan Algoritma

Terdapat sangat banyak algoritma pencocokan kata namun yang paling terkenal adalah algoritma *Knuth-Morris-Pratt* dan algoritma *Boyer-Moore*. Pada makalah ini digunakan algoritma *Knuth-Moris-Pratt* karena algoritma *Knuth-Morris-Pratt* tidak terpengaruh terhadap panjang dari *pattern* yang dicek sedangkan algoritma *Boyer-Moore* terpengaruh. Hal ini dapat dilihat dari kompleksitas *Boyer-Moore* dimana pada kompleksitas *Boyer-Moore* terdapat variable A dimana A adalah besar *alphabet* yang menandakan adanya pengaruh panjangnya kata terhadap kecepatan algoritma.

Selain perbandingan dilakukan dengan algoritma *Boyer-Moore*, juga dilakukan terhadap algoritma lain yang memang dikhususkan untuk menanangi kasus kompresi teks seperti algoritma LZ77 adalah O(BAn) dimana B adalah *hystory buffer* penyimpan, A adalah panjang *lookahead buffer*, dan n merupakan panjang *string* yang dikodekan[PRT09]. Sedangkan algoritma LZ78 adalah O(Cn) dimana C adalah besar *dictionary* yang dilakukan dan n merupakan panjang *string* yang dikodekan[PRT09]. Dengan perbandingan tersbut dirasa algoritma *Knuth-Morris-Pratt* mampu memberikan kecepatan kompresi yang tidak kalah dibandingkan dengan kedua algoritma tsb.

Disamping dilakukan perbandingan dengan algoritma lain, penggunaan algoritma *Knuth-Morris-Pratt* dilakukan pengubahan sedikit dimana jika pattern pada algoritma ini berupa string dengan panjang karakter 1, 2 dan 3 akan langsung di lewati tanpa dilakukan pengecekan (sudah dijelaskan pada bagian 3.2 analisis hasil). Selain itu penggunaan algoritma pada saat uji coba, dilakukan hanya untuk kata yang benar-benar *match*. Jadi jika pattern merupakan *substring* masih belum diberikan proses lanjutan. Namun jika ingin dikembangkan lagi, penggunaan sifat algoritma ini dalam *substring* akan menambah performa teknik kompresi dengan menggunakan algoritma ini.

V. KEKURANGAN DAN BATASAN

Pada makalah ini digunakan penelitian terhadap *file teks* (.txt) saja mengingat *file text* lebih stabil dan lebih mudah diamati pengurangan dan sebab-sebabnya. Selain itu makalah ini hanya menggunakan satu contoh sebagai uji coba. Contoh yang digunakanpun belum panjang dan memasukkan berbagai kondisi pengecekan. Hal ini tentunya belum cukup dan harus dilakukan uji coba yang lebih terhadap teknik ini.

Pada dasarnya penerapan algoritma ini masih belum sempurna karena belum ada sumber yang menggunakan algoritma ini dalam mengkompresi data. Penulis tertarik terhadap sistem kompresi data dan tercetus ide menggunakan algoritma ini sebagai salah satu metodenya. Setelah dicari di berbagai sumber, belum ada sumber yang menjelaskan penggunaan algoritma ini dalam teknik kompresi. Algoritma yang ada antara lain algoritma *Huffman*, LZ77, ZIV78 dll. Penulis yang mengerjakan dan melakukan analisis dalam waktu yang singkat tentu tidak menghasilkan hasil yang sangat baik. Meskipun demikian

diharapkan kedepannya metode ini bisa dikembangkan dan dilengkapi hingga bisa menjadi salah satu metode yang baik.

VI. KESIMPULAN

Algoritma *Knuth-Morris-Pratt* dapat digunakan sebagai salah satu metode teknik kompresi dengan melakukan pengcekan terhadap kata yang sama, kemudian dilakukan pemampatan.

REFERENCES

[RIN09] Munir, Rinaldi. (2009). *Diktat Kuliah IF3051 Strategi Algoritmia*. Bandung: Program Studi Teknik Informatika, Institut Teknologi Bandung.

[SUG07] Suyanto.ST.Msc.(2007).*Artificial-Intelligence*.Bandung:Informatika

[DIG09] NN. Digital Collection http://digilib.petra.ac.id/viewer.php?submit.x=9&submit.y=13&page=2&qual=high&submitval=prev&fname=%2Fjiunkpe%2Fs1%2Finfo%2F2009%2Fjiunkpe-ns-s1-2009-26404111-11745-deflate-chapter2.pdf Waktu : Rabu, 6 Desember 2011 [6:00]

[ANT05] Anton. Multimedia http://lecturer.ukdw.ac.id/anton/download/multimedia8.pdf Waktu : Rabu, 6 Desember 2011 [6:00]

[PRT09] Pratama,Andre. Skripsi Perbandingan Algoritma Kompresi LZ77, LZ78, LZW http://repository.usu.ac.id/bitstream/123456789/7860/1/10E00057.pdf Waktu : Rabu, 6 Desember 2011 [6:00]

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, 6 Desember 2011

I Nyoman Prama Pradnyana
13509032