BAB 2
TINJAUAN PUSTAKA

2.1 Plagiarisme

2.1.1 Pengertian Plagiarisme

Plagiarisme merupakan tindakan menjiplak, mencuri atau mengambil ide, hasil karya atau tulisan orang lain, baik seluruh, sebagian besar maupun sebagian kecil, untuk jadi ide atau karya tulisan sendiri tanpa menyebutkan nama penulis dan sumber aslinya[1].

2.1.2 Jenis Plagiarisme

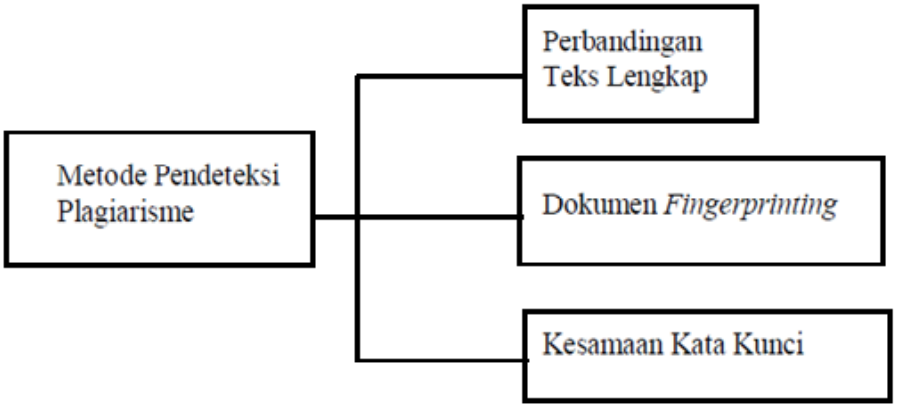
Berdasarkan sejumlah pola atau modus operansi tersebut, paling sedikit ada empat jenis plagiarisme[1].

a. Plagiarisme total yaitu tindakan plagiasi yang dilakukan seorang penulis dengan cara menjiplak atau mencuri hasil karya orang lain seluruhnya dan mengklaim sebagai karyanya sendiri.
b. Plagiarisme parsial yaitu tindakan plagiasi yang dilakukan sesorang penulis dengan cara cara menjiplak sebagian hasil karya orang lain untuk menjadi hasil karyanya sendiri. Biasanya, dalam plagiasi jenis ini seorang penulis mengambil pernyataan, landasan teori, sampel, metode analisis, pembahasan dan atau kesimpulan tertentu dari hasil karya orang lain menjadi karyanya tanpa menyebutkan sumber aslinya.
c. Auto-plagiasi (*self-plagiarisme*) yaitu plagiasi yang dilakukan seorang penulis terhadap karyanya sendiri, baik sebagian maupun seluruhnya. Misalnya, ketika menulis suatu artikel ilmiah seorang penulis meng-*copy paste* bagian-bagian tertentu dari hasil karyanya dalam suatu buku yang sudah diterbitkan tanpa menyebut sumbernya
d. Plagiarisme antarbahasa yaitu plagiasi yang dilakukan seorang penulis dengan cara menerjemahkan suatu karya tulis yang berbahasa asing ke dalam bahasa

Indonesia. Kemudian, penulis menjadikan hasil terjemahan tersebut sebagai hasil karyanya tanpa menyebut sumbernya. Modus operandinya hampir mirip dengan jenis plagiasi total dan plagiasi parsial. Asumsinya, para pembaca tidak akan tahu bahwa artikel tersebut adalah hasil terjemahan karena berbeda bahasa.

2.2 Metode Pendeteksian Kesamaan Dokumen

Metode pendeteksian kesamaan dokumen dibagi menjadi tiga bagian yaitu metode perbandingan teks lengkap, metode dokumen *fingerprinting*, dan metode kesamaan kata kunci [3]. Metode pendeteksi plagiarism dapat dilihat pada Gambar 2.1.



Gambar 2.1 Metode Pendeteksi Plagiarisme

Penjelasan dari masing-masing metode dan algoritma pendeteksi kesamaan dokumen sesuai *Gambar 2.1* yaitu:

1. Perbandingan teks lengkap. Metode ini diterapkan dengan membandingkan semua isi dokumen. Dapat diterapkan untuk dokumen yang besar. Pendekatan ini membutuhkan waktu yang lama tetapi cukup efektif, karena kumpulan dokumen yang diperbandingkan adalah dokumen yang disimpan pada penyimpanan lokal.

Metode perbandingan teks lengkap tidak dapat diterapkan untuk kumpulan dokumen yang tidak terdapat pada dokumen lokal. Algoritma yang digunakan pada metode ini adalah algoritma *Brute-Force*, algoritma *edit distance*, algoritma *Boyer Moore* dan algoritma *lavenshtein distance.*

2. Dokumen *Fingerprinting*. Dokumen *fingerprinting* merupakan metode yang digunakan untuk mendeteksi keakuratan salinan antar dokumen, baik semua teks yang terdapat di dalam dokumen atau hanya sebagian teks saja. Prinsip kerja dari metode dokumen *fingerprinting* ini adalah dengan menggunakan teknik *hashing*. Teknik *hashing* adalah sebuah fungsi yang mengkonversi setiap string menjadi bilangan. Misalnya Rabin-Karp, Winnowing dan Manber.
3. Kesamaan Kata Kunci. Prinsip dari metode ini adalah mengekstrak kata kunci dari dokumen dan kemudian dibandingkan dengan kata kunci pada dokumen yang lain. Pendekatan yang digunakan pada metode ini adalah teknik dot.

2.3 Algoritma

Algoritma adalah suatu perintah yang berisi langkah-langkah untuk menyelesaikan masalah [5]. Algoritma berasal dari nama tokoh ilmuan islam pada masa itu yaitu Abu Ja'far Muhammad Ibu Musa Al Khawārizmi yang hidup sekitar abad ke-9. Dengan karya bukunya yang terkenal yaitu Al Jabar Wal Muqabala yang berarti "Buku Pemugaran dan Pengurangan".

Pada awalnya kata algoritma adalah istilah yang merujuk kepada aturan-aturan aritmetis untuk menyelesaikan persoalan dengan menggunakan bilangan numerik arab (sebenarnya dari India, sepertitertulis pada judul di atas). Pada abad ke-18, istilah ini berkembang menjadi algoritma, yang mencakup semua prosedur atau urutan langkah yang jelas dan diperlukan untuk menyelesaikan suatu permasalahan.
Berikut adalah contoh bagaimana algoritma dapat menyelesaikan masalah :

- Terdapat dua buah gelas yang berisi air yang berwarna merah dan gelas yang berisi air berwarna biru. Kita asumsikan gelas yang berwarna merah adalah gelas

A dan gelas yang berwarna biru adalah gelas B. Jika isi gelas A ingin kita pindahkah ke dalam gelas B dan isi gelas B berpindah ke gelas A agar tidak merubah warna masing-masing gelas tentukan cara bagaimana mana gelas tersebut dapat berpindah tempat tanpa merubah isi dalam gelas tersebut.

Penyelesaiannya :

1. tambahkan gelas kosong yang diasumsikan sebagai gelas C.
2. Pindahkan isi gelas A ke dalam gelas kosong C sehingga sekarang gelas A kosong dan isi gelas C adalah air berwarna merah.
3. Setelah itu pindahkan isi gelas B ke dalam gelas A sehingga isi gelas B kosong dan berpindah ke gelas A.
4. Pindahkan isi gelas C yang berisi air berwarna merah kedalam gelas B yang sudah kosong karena isinya sudah berpindah ke dalam gelas A.
5. Hasil akhirnya adalah A berisi air berwarna biru dan B berisi air berwarna merah.

2.3.1 Dasar Algoritma

Dasar algoritma terdiri dari beberapa tahapan. Tahapan-tahapan ini merupakan tindakan yang sering digunakan seperti sequence, pemilihan dan perulangan.

2.3.1.1 Sequence

Algoritma merupakan runtunan satu atau lebih instruksi, yang berarti bahwa :

a. Tiap instruksi di kerjakan satu persatu;
b. Tiap instruksi dilaksanakan tepat sekali; tidak ada instruksi yang di ulang;
c. Urutan instruksi yang dilaksanakan pemroses sama dengan urutan instruksi sebagaimana yang tertulis didalam teks algoritmanya
d. Akhir dari instruksi terakhir adalah akhir algoritma.

Langkah 1

Tuangkan isi gelas A ke dalam gelas C



Langkah 2

Tuangkan isi gelas B ke dalam gelas A

Langkah 3

Tuangkan isi gelas C ke dalam gelas B

2.3.1.2 Pemilihan

Adakalanya sebuah program dihadapkan pada suatu kondisi dimana kondisi tersebut menentukan alur program yang akan kita buat.

Contoh 1

Jika keran air kita tutup maka

Air tidak akan keluar

Pernyataan diatas dapat kita tulis dalam pernyataan kondisional sebagai berikut :

If kondisithen Aksi

Jika kita tulis :

If keran air di tutup then Air tidak akan keluar

If keran air di buka then Air akan keluar

Dalam bahasa Indonesia If berarti jika dan then berarti maka dimana setiap kondisi bisa menghasilkan nilai benar atau salah.

2.3.1.3 Perulangan

Jika di banding dengan manusia kelebihan komputer adalah tidak mengenal kata lelah jika mengerjakan pekerjaan yang sama secara berulang kali. Contoh algoritma yang kurang cerdas untuk menuliskan kata sebanyak 100 x sebagai berikut:

Program menulis_kata;

{Menuliskan kata sebanyak 100 kali}

Deklarasi

{Tidak ada}
Algoritma
Tulis ‘Saya akan belajar yang rajin’;
Tulis ‘Saya akan belajar yang rajin’;
Tulis ‘Saya akan belajaryang rajin’;
Tulis ‘Saya akan belajar yang rajin’;
Tulis ‘Saya akan belajar yang rajin’;
……..

Alangkah baiknya jika pengulangan tersebut dibuat secara otomatis yaitu dengan menggunakan algoritma perulangan sebagai berikut :

Program menulis_kata;
{Menuliskan kata sebanyak 100 kali}
Deklarasi
Algoritma
For i dari 1 sampai 100
Tulis ‘Saya akan belajar yang rajin’;
End for

Algoritma diatas akan mengulangi kata ‘Saya akan belajar yang rajin’ sampai 100 kali, apabila perulangan sudah terpenuhi maka perulangan pun akan berhenti.

2.3.2 Stuktur Algoritma

Berikut ini merupakan salah satu struktur algoritma yang dipakai sebagai patokan yaitu:

a. Bagian kepala (Header)
   Pada bagian ini memuat nama algoritma serta informasi atau keterangan algoritma

b. Bagian Deklarasi (Definisi Variable)

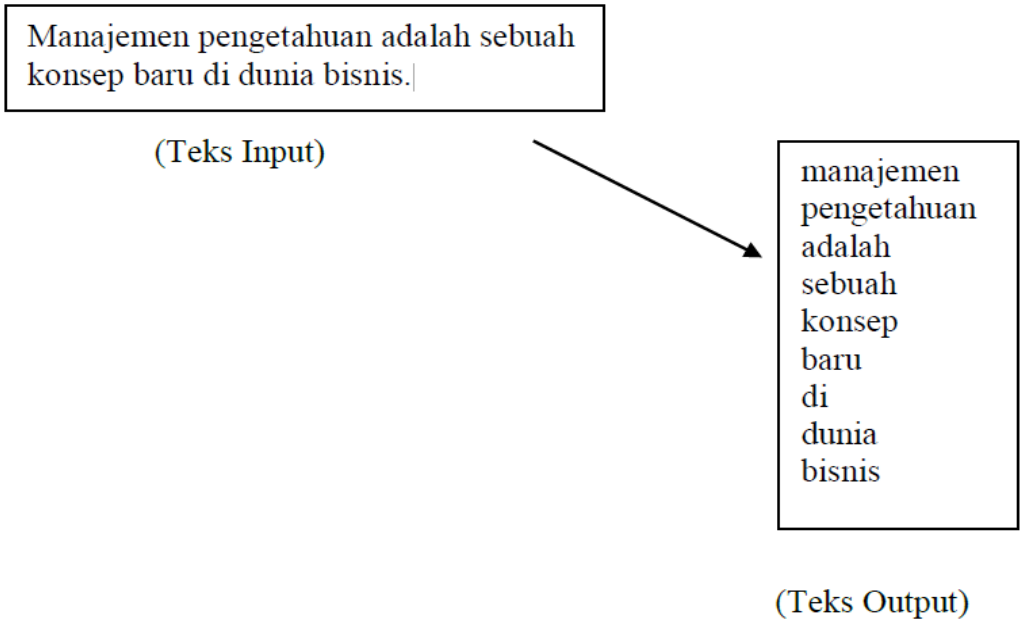
   Pada bagian ini memuat definisi nama variable, nama tetapan, nama procedure, nama fungsi, dan tipe data yang digunakan oleh algoritma

c. Bagian Deskripsi (Rincian Langkah)

   Pada bagian ini memuat langkah-langkah penyelesaian masalah termasuk beberapa perintah seperti baca data, tampil data, perulangan yang mengubah data input menjadi output [6]

2.4 Tokenizing

Tahap *tokenizing / parsing* adalah tahap pemotongan string input berdasarkan tiap kata yang menyusunnya. Karakter selain huruf dihilangkan dan dianggap *delimiter*. *Tokenizing* biasanya digunakan pada tahap *preprocessing* sehingga kata-kata pada sebuah dokumen dibagi menjadi beberapa kata sesuai dengan delimeter pembagi kata yang telah ditentukan. *Tokenizing* sangat berguna ketika sebuah program pengolah teks memerlukan data sebuah kata yang tersusun dan terbagi menjadi array. Proses *tokenizing* dapat dilihat pada Gambar 2.3 [7] :

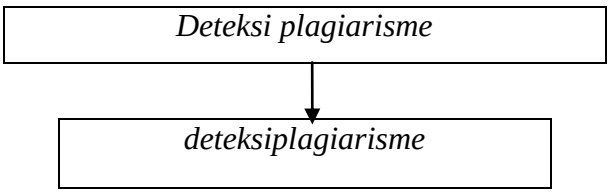


Gambar 2.2 Tokenizing

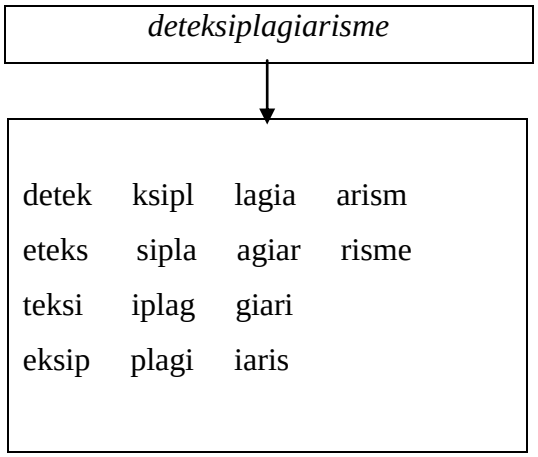
2.5 Algoritma Winnowing

Algoritma winnowing merupakan algoritma dokumen *fingerprinting* yang digunakan untuk mendeteksi salinan dokumen dengan menggunakan teknik *hashing* [4]. Input dari algoritma ini adalah dokumen teks yang diproses sehingga menghasilkan output berupa kumpulan nilai-nilai *hash*, nilai *hash* merupakan nilai numerik yang terbentuk dari perhitungan ASCII tiap karakter . Kumpulan-kumpulan nilai *hash* tersebut selanjutnya disebut *fingerprint*. *Fingerprint* inilah yang digunakan dalam deteksi penjiplakan [8].

Langkah awal dalam penerapan algoritma *Winnowing* adalah membuang karakter-karakter dari isi dokumen yang tidak relevan misal tanda baca spasi dan simbol lain sehingga yang terbaca hanya karakter *string*. Sebagai contoh:



Langkah kedua, isi dokumen yang telah dilakukan pembersihan selanjutnya dilakukan pembentukan rangkaian gram, dimana k =5.



Langkah ketiga dari rangkaian gram yang telah terbentuk dibentuk nilai *hash* dengan nilai ASCII tiap karakter. Perhitungan nilai hash menggunakan metode *Rolling Hash* dan menghasilkan nilai *hash* dari setiap gram sebagai berikut:

12281, 12658, 13536, 12532, 13161, 13579, 12895, 13275, 12706, 11988, 12498, 12580, 12334, 13532

Setelah mendapatkan nilai hash langkah selanjutnya adalah pembagian nilai hash menurut *window* dengan besar *window w* = 4

{**12281**, 12658, 13536, 12532} {12658, 13536, **12532**, 13161}
{13536, 12532, 13161, 13579} {12532, 13161, 13579, **12895**}
{13161, 13579, 12895, 13275} {13579, 12895, 13275, **12706**}
{12895, 13275, 12706, **11988**} {13275, 12706, 11988, 12498}
{12706, 11988, 12498, 12580} {11988, 12498, 12580, **12334**}
{12498, 12580, 12334, 13532}

Langkah ke lima adalah memeilih nilai hash terkecil dari setiap window untuk dijadikan nilai fingerprints dari dokumen tersebut:

12281, 12532, 12895, 12706, 11988, 12334

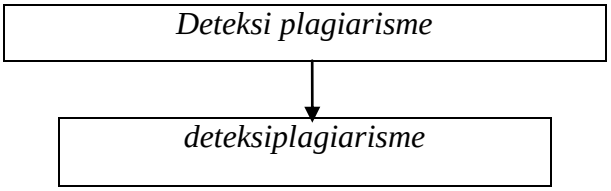
Implementasi algoritma winnowing yang efisien sebaiknya menyimpan posisi *fingerprints* yang telah didapatkan (Elbegbayan, 2005), dimana posisi dimulai dari 0 dari nilai hash pada gram yang terbentuk sebelumnya [*fingerprints, posisi*].

[12281,0] [12532,3] [12895,6] [12706,8] [11988,9] [12334,12]

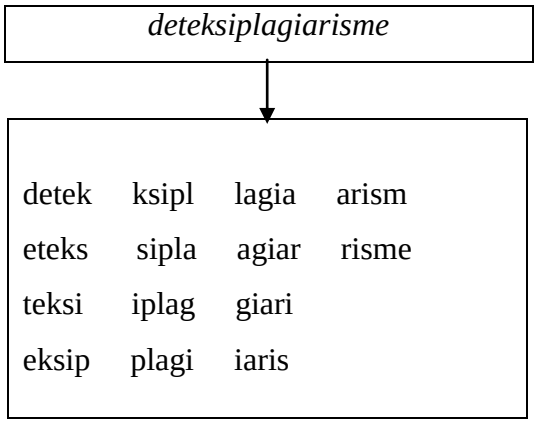
2.6 Algoritma Manber

Algoritma Manber melakukan pemilihan nilai *fingerprints* yang telah diperoleh dari proses *hashing* dengan memilih nilai *fingerprints* yang memenuhi kriteria 0 mod p [9] Contoh tahapan Algoritma Manber dapat dilihat sebagai berikut:

Langkah awal dalam penerapan algoritma *manber* adalah membuang karakter-karakter dari isi dokumen yang tidak relevan misal tanda baca spasi dan simbol lain. Sebagai contoh:



Langkah kedua isi dokumen yang telah dilakukan pembersihan selanjutnya dilakkukan pembentukan rangkaian gram, dimana k =5.



Langkah ketiga dari rangkaian gram yang telah terbentuk dibentuk nilai *hash* dengan nilai ASCII tiap karakter. Perhitungan nilai hash menggunakan metode Rolling Hash dan menghasilkan nilai *hash* dari setiap gram sebagai berikut:

12281, 12658, 13536, 12532, 13161, 13579, 12895, 13275, 12706, 11988, 12498, 12580, 12334, 13532

Setelah mendapatkan nilai hash langkah selanjutnya adalah pemilihan nilai hash menurut ukuran 0 mod p, p=4 maka diperoleh nilai fingerprints:

13536, 13161, 13275, 11988, 12498

2.7 Rolling Hash

Rolling hash adalah suatu cara menyimpan dan mengambil target elemen tanpa *searching,* yaitu dengan cara menghitung lokasi target. Fungsi *hash* dengan basis disebut dengan *Rolling Hash.* Pada awalnya metode *Rolling Hash* digunakan pada Algoritma Rabin Karp dimana metode ini digunakan untuk membandingan nilai *hashing* dari semua k-gram kedalam sebuah string yang panjang [10]. Persamaannya adalah sebagai berikut :

$$H_{(c_1..c_{k+1})} = C_1 * b^{k-1} + C_2 * b^{k-2} + \cdots + c_{k-1} * b + c_k \qquad (2.1)$$

Keterangan:
H: nilai hash
c : nilai ascii karakter
b : basis (bilangan prima)
k : banyak karakter

Contoh Penggunaan:
Kata: “sepeda”

Diambil nilai ascii dari setiap karakter dari kata “sepeda” didapatkan nilai ascii sebagai berikut:

s=115, **e**=101, **p**=112, **e**=101, **d**=100, **a**=97

Dari rumus Rolling Hash pertama didapatkan hasil sebagai berikut:

$155*3^5 + 101*3^4 + 112*3^3 + 101*3^2 + 100*3 + 97 = 40456$

115*243 + 101* 81 + 112*27 + 101*9 + 100*3 + 97 = 40456

27945 + 8181 + 3024 + 909 + 300 + 97 = 40456

R*olling hash* memiliki dua rumus untuk menghitung nilai hash karakter berikutnya yaitu H(c2...ck+1) dapat dilakukan dengan cara:

$$H_{(C_2..c_{k+1})} = \left(H_{C_1..C_k)} - C_1 * b^{k-1}\right) * b + c_{(k+1)} \qquad (2.2)$$

Dengan begitu tidak perlu melakukan iterasi dari indeks pertama sampai terakhir untuk menghitung nilai *hash* untuk *gram* ke-2 sampai terakhir.

Contoh Penggunaan:

- Kata: “s**epedai**tubagus”
- Diambil nilai ascii dari setiap karakter dari kata “sepeda” dan dilakukan perhitungan menggunakan rumus *Rolling Hash* pertama didapatkan nilai hash 40456
- Karena setiap kata yang bergeser memiliki huruf yang pernah dihitung sebelumnya yaitu **“sepeda”** dan bergeser menjadi **“epedai”** digunakanlah rumus Rolling Hash kedua dan menghasilan nilai sebagai berikut:

  H=((40456-(115*243))*3+105) = 37638

2.8 Jaccard’s Similarity Coefficient

Untuk menghitung persentasi kemiripan (*similarity)* dari sebuah dokumen dapat dilakukan menggunakan *Jaccard’s Similarity Coefficient* [4] dengan rumus:

$$D(A,B) = \frac{|A \cap B|}{|A \cup B|} \times 100 \qquad (2.3)$$

Keterangan:

A : nilai *fingerprints* dokumen 1

B : nilai *fingerprints* dokumen 2

Dimana D(A,B) adalah nilai similaritas, $|A \cap B|$ adalah jumlah *fingerprint* yang sama dari dokumen 1 dan 2, $|A \cup B|$ dalah jumlah seluruh *fingerprint* dari dokumen 1 dan 2. Sebagai contoh a={1,2,4} dan b={1,2,4,7,8} maka $|A \cap B|$ = {1,2,4} dan $|A \cup B|$ = {1,2,4,7,8} sehingga, D(a,b) = 3/5 * 100 = 60

2.9 Teori Analisis Algoritma

Algoritma yang baik adalah algoritma yang mangkus. Kemangkusan algoritma diukur dari berapa jumlah waktu dan ruang(*space*) memori yang dibutuhkan untuk menjalankannya. Algoritma yang mangkus ialah algoritma yang meminimumkan kebutuhan waktu dan ruang **[11]** Kebutuhan waktu dan ruang suatu algoritma bergantung pada ukuran masukan yang secara khas adalah jumlah data yang diproses. Ukuran masukan itu disimbolkan dengan *n.* Misalkan bila mengurutkan 1000 buah data maka n=1000.

Analisis waktu sebuah algoritma bergantung pada mesin komputer yang digunakan. Perbedaan kecepatan sebuah komputer atau mesin compiler dalam memproses sebuah eksekusi program mengakibatkan perbedaan hasil waktu proses. Misalkan terdapat sebuah komputer yang mampu menjalankan program dengan masukan berukuran *n* dalam waktu $\mathbf{10^{-4} \times 2^{n}}$ detik. Maka dapat dihitung bahwa untuk,

a. n = 10, dibutuhkan waktu eksekusi kira-kira 1/10 detik,
b. n = 20, dibutuhkan waktu eksekusi kira-kira 2 menit,
c. n = 30, dibutuhkan waktu eksekusi lebih dari satu hari

Jika kita mengganti mesin baru yang dapat memproses 100 kali lebih cepat dari mesin pertama menjadi $\mathbf{10^{-6}}$, maka proses yang dapat dilakukan lebih banyak karena waktu proses semakin cepat menjadi menjadi $\mathbf{10^{-6} \times 2^{n}}$ detik.

2.9.1 Kompleksitas Waktu dan Ruang

Secara teoritis model abstrak pengukuran waktu atau ruang harus independen dari pertimbangan mesin compiler apapun. Model abstrak seperti itu dapat dipakai untuk membandingkan algoritma yang berbeda. Kompleksitas waktu diekspresikan sebagai jumlah tahapan komputasi yang dibutuhkan untuk menjalankan algoritma sebagai fungsi dari ukuran masukan *n* [11]. kompleksitas ruang diekspresikan sebagai jumlah memori yang digunakan oleh struktur data yang terdapat di dalam algoritma sebagai fungsi dari ukuran masukan *n*. dengan menggunakan besaran kompleksitas waktu/ruang algoritma, dapat ditentukan laju peningkatan waktu/ruang yang diperlukan algoritma dengan meningkatnya ukuran masukan n. Umumnya Algoritma mempunyai parameter utama yang paling menentukan waktu proses. Parameter ini berkaitan dengan jumlah data yang diproses algoritma tersebut. Parameter ini dapat berupa pangkat suatu persamaan *polinom* atau jumlah *record* yang akan diurutkan atau dicari. Kompleksitas algoritma menyatakan kelajuan perkembangan waktu proses dibandingkan dengan kelajuan perkembangan data yang diproses . Salah satu notasi kompleksitas algoritma adalah *Big-Oh*. Definisi dari *Big-O* Adalah fungsi yang lebih berkaitan dengan kelajuan proses daripada kelajuan pertambahan data yang dirumuskan dengan T(n) = O(f(n)). Jika ada konstan c dan no sedemikian rupa sehingga $T(n) \leq c.\ f(n)$ untuk $n \geq no$. Secara sederhana dikatakan bahwa O(f(n)) dapat dianggap sebagai nilai maksimum dari c.f(n).

2.9.1.1 Kompleksitas Waktu

Kompleksitas waktu diukur dengan menghitung banyaknya operasi yang dilakukan oleh algoritma [12]. Sebagai contoh terdapat suatu algoritma yang

memiliki proses pengisian nilai dan penjumlahan sesuai dengan pseudocode sebagai berikut:

a. Operasi Pengisian Nilai

Jumlah ← 0,	jumlah eksekusi 1 kali
k ← 1,	jumlah eksekusi 1 kali
jumlah ← jumlah + $\boldsymbol{a_k}$,	jumlah eksekusi n kali
k ← k+1,	jumlah eksekusi n kali
r ← jumlah/n	jumlah eksekusi 1 kali

Jumlah seluruh operasi pengisian nilai adalah:

t1 = 1 + 1 + n + n + 1 = 3 + 2n

b. Operasi Penjumlahan

Jumlah + $\boldsymbol{a_k}$,	jumlah eksekusi n kali
k + 1,	jumlah eksekusi n kali

jumlah seluruh operasi penjumlahan adalah:

t2 = n + n = 2n

Dari kedua proses pengisian nilai dan penjumlahan pada algoritma tersebut dapat dihitung jumlah kompleksitas waktu berdasarkan jumlah operasi yang telah dilakukan yaitu:

T(n) = t1 + t2 = 3 + 2n + 2n = 4n + 3, dimana T(n) merupakan kompleksitas waktu yang menghasilkan 4n + 3

2.9.2 Aturan Algoritma

Aturan algoritma digunakan untuk mengklasifikasikan sebuah algoritma termasuk pada jenis algoritma seperti apa keterhubungan algoritma tersebut pada T(n) [11].

a. Perulangan (*For Loop*)

Algoritma yang termasuk pada jenis ini memiliki waktu eksekusi maksimum sebanyak waktu eksekusi statement yang ada didalam loop dikalikan banyaknya iterasi. Contoh:

```
For(a=0;a<n;a++)
{
        m=a+b;
        n=c+d;
}
```

Waktu eksekusi yaitu 2 x n sehinggal T(n) = O(n). Algoritma yang memiliki 1 kali perulangan termasuk pada klasifikasi O(n).

b. Perulangan Bersarang (*Nested For Loop*)

Algoritma yang termasuk pada jenis ini yaitu algoritma yang memiliki perulangan bersarang. Waktu eksekusi total sebuah statement yaitu waktu eksekusi statement dikalikan hasil kali dari banyaknya iterasi semua loop yang ada didalamnya. Contoh:

```
for(int i=0;i<n;i++)
  for(int j=0;j<m;j++)
a[i,j]=0;
```

Dari pseudocode diatas a[i,j] akan dieksekusi sebanyak (m x n) kali sehingga T(n) termasuk dalam T(n) = O($\boldsymbol{n^2}$).

c. Statement Yang Berurutan (*Consecutive Statement*)

Untuk statement yang berurutan, waktu eksekusinya adalah jumlah dari masing-masing statement. Contoh:

```
for(int k=0;k<10;k++)
x[k]=0;
for(int i=0;i<n;i++)
for(int j=0;j<m;j++)
a[i,j]=0;
```

Jadi T(n)=T1(n)+T2(n)+T3(n) = O($\boldsymbol{n^3}$)

2.10 Pengertian ASCII

Kode Standar Amerika untuk Pertukaran Informasi atau ASCII (*American Standard Code for Information Interchange*) merupakan suatu standar internasional dalam kode huruf dan simbol seperti Hex dan Unicode tetapi ASCII lebih bersifat universal, contohnya 124 adalah untuk karakter "|". Ia selalu digunakan oleh komputer dan alat komunikasi lain untuk menunjukkan teks. Kode ASCII sebenarnya memiliki komposisi bilangan biner sebanyak 8 bit. Dimulai dari 0000 0000 hingga 1111 1111. Total kombinasi yang dihasilkan sebanyak 256, dimulai dari kode 0 hingga 255 dalam sistem bilangan Desimal

2.10.1 Tabel Karakter ASCII

Tabel berikut berisi karakter-karakter ASCII. Dalam sistem operasi Windows dan MS-DOS, pengguna dapat menggunakan karakter ASCII dengan menekan tombol Alt+[nomor nilai ANSI (desimal)]. Sebagai contoh, tekan kombinasi tombol **Alt+87** untuk karakter huruf latin "W" kapital. Tabel kode ascii [13] dapat dilihat pada gambar 2.3.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Gambar 2.3 Table ascii

2.11 Data Flow Diagram (DFD)

Data Flow Diagram (DFD) merupakan diagram yang mengunakan notasi-notasi atau simbol-simbol untuk mengambarkan sistem jaringan kerja antar fungsi-fungsi yang berhubungan satu sama lain dengan aliran dan penyimpanan data (Jogiyanto, 2005).

DFD sering digunakan untuk menggambarkan suatu sistem yang telah ada atau sistem baru yang akan dikembangkan secara logika tanpa mempertimbangkan lingkungan fisik dimana data tersebut mengalir atau dimana data tersebut akan disimpan. Salah satu keuntungan menggunakan diagram aliran data adalah

memudahkan pemakai (*user*) yang kurang menguasai bidang komputer untuk mengerti sistem yang akan dikerjakan.

DFD terdiri dari diagram konteks (*context diagram*) dan diagram rinci (*level diagram*). Diagram konteks adalah diagram yang terdiri dari suatu proses dan menggambarkan ruang lingkup suatu sistem. Diagram konteks merupakan level tertinggi dari DFD yang menggambarkan seluruh input ke sistem atau output dari sistem. Sistem dibatasi oleh *boundary* (dapat digambarkan dengan garis putus). Dalam diagram konteks biasanya hanya ada satu proses. Tidak boleh ada store dalam diagram konteks. Diagram rinci adalah diagram yang menguraikan proses apa yang ada dalam diagram level di atasnya.

Adapun yang digunakan dalam DFD adalah:

1. Entitas Eksternal (*External Entity*)

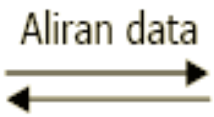
   Entitas Eksternal (*entity*) di lingkungan luar sistem yang dapat berupa orang, organisasi atau sistem lainnya yang berada di lingkungan luarnya yang akan memberikan input atau menerima output dari sistem. Suatu kesatuan luar dapat disimbolkan dengan suatu notasi persegi panjang.



Gambar 2.4 Entitas

2. Aliran data

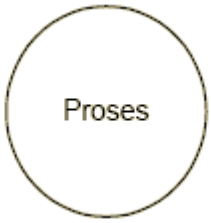
   Aliran data di DFD diberikan simbol suatu panah. Aliran data ini mengalir diantara proses (*process*), simpanan data (*data store*) dan kesatuan luar (*External ent*ity). Aliran data ini menunjukkan arus dari data yang dapat berupa masukan untuk sistem atau hasil dari proses sistem.



Gambar 2.5 Aliran Data

3. Proses

   Suatu proses adalah kegiatan atau kerja yang dilakukan oleh orang, mesin atau komputer dari hasil suatu aliran data yang masuk ke dalam proses untuk dihasilkan aliran data yang akan keluar dari proses. Suatu proses dapat disimbolkan dengan notasi lingkaran.



Gambar 2.6 Proses

4. Penyimpan Data (*Data Store*)

   Penyimpan data (*data store*) merupakan penyimpan data yang dapat berupa:

   a. Suatu file atau basis data di sistem komputer.
   b. Suatu arsip atau catatan manual.
   c. Suatu tabel acuan manual.
   d. Suatu agenda atau buku.

   Simpanan data di DFD dapat disimbolkan dengan sepasang garis horizontal paralel yang ujungnya tidak ditutup.



Gambar 2.7 Data Store