BAB II

TINJAUAN PUSTAKA

2.1 E-commerce

Electronic Commerce (*e-commer*ce) adalah proses pembelian, penjualan atau pertukaran produk, jasa dan informasi melalui jaringan komputer. *E-commerce* merupakan bagian dari *e-business*, dimana cakupan *e-business* lebih luas, tidak hanya sekedar perniagaan tetapi mencakup juga pengolaborasian mitra bisnis, pelayanan nasabah, lowongan pekerjaan, dan sebagainya. *E-commerce* juga memerlukan teknologi basis data atau pangkalan data (*database*), e-surat atau surat elektronik (*e-mail*), dan bentuk teknologi non komputer yang lain seperti halnya sistem pengiriman barang, dan alat pembayaran untuk *e-commerce* ini (Siregar, 2010).

Dalam mengimplementasikan *e-commerce* tersedia suatu integrasi rantai nilai dari infrastrukturnya, yang terdiri dari tiga lapis. Pertama, insfrastruktur sistem distribusi (*flow of good*); kedua, insfrastruktur pembayaran (*flow of money*); dan ketiga, infrastruktur sistem informasi (*flow of information*). Agar dapat terintegrasinya sistem rantai suplai dari supplier, ke pabrik, ke gudang, distribusi, jasa transportasi, hingga ke pelanggan maka diperlukan integrasi *enterprise system* untuk menciptakan *supply chain visibility*. Ada tiga faktor yang perlu dicermati jika ingin membangun toko e-commerce yaitu: *variability*, *visibility*, dan *velocity* (Sukmajati, 2009).

Dengan *e-commerce*, perusahaan dapat menekan biaya yang harus dikeluarkan untuk keperluan informasi (iklan), selain itu dengan *e-commerce*

transaksi bisa berlangsung dengan cepat dan biaya lebih efisien, sehingga dapat meningkatkan kemampuan perusahaan dalam bersaing.

E-commerce akan mengubah semua kegiatan *marketing* dan juga sekaligus memangkas biaya-biaya operasional untuk kegiatan *trading* (perdagangan). Proses yang ada dalam *e-commerce* adalah sebagai berikut : (Irmawati, 2011)

a. Presentasi elektronis (pembuatan *website*) untuk produk dan layanan.
b. Pemesanan secara langsung dan tersedianya tagihan.
c. Otomatisasi akun pelanggan secara aman (baik nomor rekening maupun nomor Kartu Kredit).
d. Pembayaran yang dilakukan secara langsung (*online*) dan penanganan transaksi.

Situs *e-commerce* adalah situs komersial yang siap menerima *order* dan dapat diakses dari berbagai wilayah di dunia dengan wilayah waktu yang bermacam-macam. Oleh karena itu, pemilik situs harus mempertimbangkan beban biaya dan wilayah jangkauan distribusi produk mereka sampai tingkat nasional. (Irmawati, 2011)

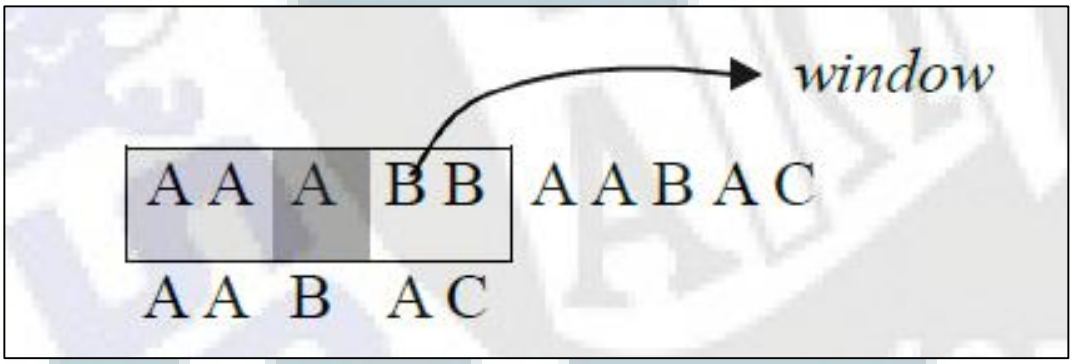
2.2 String Searching

String adalah urutan dari karakter, yang mana karakter ini dapat terdiri dari beberapa *alphabet*. Misalnya adalah *string* biner yang terdiri dari dua *alphabet*, yaitu 0 dan 1. Contoh yang lain adalah *string* ASCII yang terdiri dari 256 *alphabet*. (Utomo, Harjo, & Handoko, 2008)

String searching pada dasarnya adalah mencari pola P yang memiliki panjang m dalam suatu teks T yang memiliki panjang n. Baik pola maupun teks

dinyatakan dalam *array*, pola dinyatakan dengan P[0 .. m-1] dan teks dinyatakan dengan T[0 .. n-1]. (Utomo, Harjo, & Handoko, 2008)

Kecocokan terjadi apabila karakter pada teks dan pola yang dibandingkan adalah sama, sedangkan ketidakcocokan adalah sebaliknya.



Gambar 2.1 Perbandingan *String* (Utomo, Harjo, & Handoko, 2008)

*Abu - abu muda menunjukkan kecocokan

*Abu - abu tua menunjukkan ketidakcocokan

Window adalah kotak pada teks yang memiliki ukuran sama dengan panjang pola yang berfungsi membantu pencarian pola dalam teks. Pertama kali, *window* diletakkan di posisi paling kiri dari teks yang kemudian dilakukan perbandingan karakter-karakter di *window* dengan di pola. Pergeseran *window* ke kanan akan dilakukan jika terjadi dua sebab. Yang pertama adalah jika terjadi kecocokan dari semua karakter di pola atau berarti pola ditemukan di teks. Pergeseran karena sebab yang pertama ini dilakukan untuk mencari penemuan pola yang berikutnya. Sebab yang kedua adalah jika terjadi ketidakcocokan. Mekanisme ini diulang terus menerus sampai batas kanan dari *window* melebihi batas kanan dari teks (Utomo, Harjo, & Handoko, 2008).

Algoritma-algoritma *string matching* mempunyai tiga komponen, yaitu: *pattern,* teks dan alfabet dengan asumsi sebagai berikut:

1. *Pattern*, yaitu deretan karakter yang dicocokkan dengan teks, dinyatakan dengan x[0..m-1], panjang *pattern* dinyatakan dengan m.
2. Teks, yaitu tempat pencocokan *pattern* dilakukan, dinyatakan dengan y[0..n-1], panjang teks dinyatakan dengan n.

Alfabet, yang berisi semua simbol yang digunakan oleh bahasa pada teks dan *pattern*, dinyatakan dengan notasi, dengan ukuran dinyatakan sebagai ASIZE (Diana Effendi, 2009).

2.3 Algoritma Knuth Morris Pratt

Algoritma Knuth Morris Pratt (KMP) adalah salah satu algoritma yang digunakan untuk melakukan proses pencocokan *string*. Algoritma ini merupakan jenis *Exact String Matching Algorithm* yang merupakan pencocokan *string* secara tepat dengan susunan karakter dalam *string* yang dicocokkan memiliki jumlah maupun urutan karakter dalam *string* yang sama. Algoritma ini membandingkan satu persatu antara karakter di teks dan di pola dari kiri ke kanan. Tujuan algoritma ini memanfaatkan karakter-karakter pola yang sudah diketahui ada di dalam teks sampai terjadinya ketidakcocokan untuk melakukan pergeseran (Rusli, 2014).

Algoritma KMP melakukan proses awal (*preprocessing*) terhadap *pattern* P dengan menghitung fungsi pinggiran. Fungsi ini mengindikasikan pergeseran P terbesar yang mungkin dengan menggunakan perbandingan yang dibentuk sebelum pencarian *string*. (Sulun, 2007)

Fungsi pinggiran dihitung hanya berdasarkan kepada karakter-karakter dalam *pattern*, tidak menyertakan karakter-karakter dalam teks (*string* target).

Fungsi pinggiran *b(j)* didefinisikan sebagai ukuran awalan terpanjang dari *pattern P* yang merupakan akhiran dari *P[1..j]*. Untuk lebih jelasnya, berikut ini diberikan sebuah contoh untuk menghitung fungsi pinggiran dari sebuah *pattern P* = xlnxls. Sebagai catatan, penulis menggunakan nilai 0 (nol) sebagai indeks awal *string* pada permasalahan ini. (Sulun, 2007)

Awalan dari *P* adalah

□, x, xl, xln, xlnx, xlnxl

Akhiran dari *P* adalah

□, s, ls, xls, nxls, lnxls

Keterangan : □ = *string* kosong

Nilai fungsi pinggiran *b(j)* untuk setiap karakter dalam *P* adalah

Tabel 1.1 Tabel Fungsi pinggiran (Sulun, 2007)

j	0	1	2	3	4	5
P[j]	X	l	n	x	l	s
b(j)	0	0	0	1	2	0

Dengan larik penampung fungsi pinggiran, yaitu b : *array of integer*

Algoritma menghitung fungsi pinggiran itu sendiri adalah sebagai berikut.

```
procedure HitungPinggiran(input m :
integer, P : string)
{ Menghitung nilai pinggiran b[1..m]
untuk pattern P[1..m] }

Kamus :
  k, q : integer

Algoritma :
  b[] = array[1..m] of integer
  b[0] = 0
  q = 1
  k = 0

  for q ← 1 to m-1 do
```

Gambar 2.2 Algoritma Fungsi Pinggiran (Sulun, 2007)

```
  while ((k > 0) and (P[q] ≠ P[k]))
do
      k ← b[k]
    endwhile

    if P[q] = P[k] then
      k ← k+1
    endif

    b[q] ← k
  endfor
```

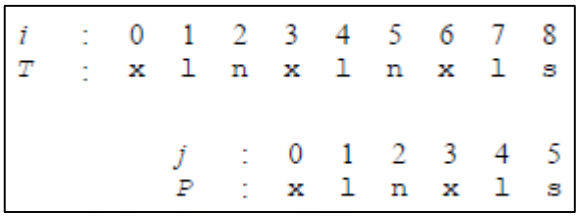
Gambar 2.2 Algoritma Fungsi Pinggiran (Lanjutan) (Sulun, 2007)

Sekarang ditinjau percobaan mencocokkan *pattern P* tadi dengan teks *T* = xlnxlnxls.

i	:	0	1	2	3	4	5	6	7	8
T	:	x	l	n	x	l	n	x	l	s
j	:	0	1	2	3	4	5			
P	:	x	l	n	x	l	s			

Gambar 2.3 Contoh Pencocokan *String* (Sulun, 2007)

Langkah pertama yaitu bandingkan ujung kiri *pattern* P dengan ujung kiri teks T. Karakter-karakter pada posisi 0 sampai dengan 4 sama (*match*), tetapi pada posisi $i = j = 5$ terjadi ketidakcocokan, n pada teks *T* dengan s pada *pattern P*. Karena terjadi ketidakcocokan, dilakukan pergeseran *pattern P* dengan jumlah pergeseran sesuai dengan nilai pinggiran dari awalan *pattern P* yang *match.* Pada kasus ini, awalan yang *match* adalah xlnxl dengan panjang $l = 5$. Nilai pinggiran yang terpanjang untuk *string P[0..4]* adalah *b(4)* = 2. Jumlah pergeseran adalah $l - b = 5 - 2 = 3$. Jadi, *pattern P* digeser sejauh tiga karakter ke kanan dan perbandingan selanjutnya dilakukan mulai pada posisi $j = l - b - 1 = 5 - 2 - 1 = 2$.



Gambar 2.4 Contoh Pergeseran *String* (Sulun, 2007)

Algoritma KMP selengkapnya adalah sebagai berikut.

```
function KMPSearch(input m : integer,
n : integer, P : string, T string) →
boolean
{ Mengembalikan true jika pattern P
ditemukan dalam teks T }

Kamus :
  i, j : integer
  ketemu : boolean
```

```
procedure HitungPinggiran(input m :
integer, P : string)
{ Menghitung nilai pinggiran b[1..m]
untuk pattern P[1..m] }

Algoritma :
  HitungPinggiran(m, P)
  i ← 0
  j ← 0
  ketemu = false

  while ((j > 0) and (P[j] ≠ T[i]))
do
    j ← b[j-1]
  endwhile

  if P[j] = T[i] then
    j ← j+1
  endif

  if j = m then
    ketemu ← true
  else
    i ← i+1
  endif

  return ketemu
```

Gambar 2.5 Algoritma Knuth Morris Pratt (Sulun, 2007)

Dengan demikian, untuk menghitung fungsi pinggiran, algoritma KMP membutuhkan waktu *O(m)* dan untuk proses pencarian *string* membutuhkan waktu *O(n)*. Jadi, kompleksitas waktu algoritma ini sebesar *O(m+n)* (Sulun, 2007).

Dalam penelitian ini, algoritma KMP digunakan untuk melakukan pencocokan *string* saat pelanggan mencari data barang yang ada dalam *database* dengan meng-*input* nama produk yang dicari.