

yang besar. Algoritma yang lebih cepat daripada algoritma pencarian beruntun adalah algoritma pencarian bagidua, yang diberikan di bawah ini.

## 1.4 Pencarian Bagidua

Pencarian bagidua atau pencarian biner (*binary search*) adalah metode pencarian yang diterapkan pada sekumpulan data yang sudah terurut (terurut menaik atau terurut menurun). Metode ini digunakan untuk kebutuhan pencarian dengan waktu yang cepat. Data yang terurut adalah syarat untuk penerapan algoritma pencarian bagidua. Memang, syarat ini pula yang merupakan kelemahan algoritma pencarian bagidua, berbeda pada pencarian beruntun yang dapat diterapkan baik pada data terurut maupun pada data yang tidak terurut. Namun, sebenarnya data yang sudah terurut banyak ditemukan di dalam kehidupan sehari-hari. Data nomor telepon di buku telepon misalnya, sudah terurut berdasarkan nama/instansi pelanggan telepon dari A sampai Z. Data karyawan diurut berdasarkan nomor induknya dari nomor kecil ke nomor besar. Data mahasiswa diurut berdasarkan NIM (Nomor Induk Mahasiswa), kata-kata (*entry*) di dalam kamus bahasa Inggris/Indonesia telah diurut dari A sampai Z, dan sebagainya.

Salah satu keuntungan data yang terurut adalah memudahkan pencarian, yang dalam hal ini pencarian bagidua. Sebenarnya, dalam kehidupan sehari-hari kita sering menerapkan pencarian bagidua. Untuk mencari arti kata tertentu di dalam kamus (misalnya kamus Bahasa Inggris), kita tidak membuka kamus itu dari halaman awal sampai halaman akhir satu per satu, namun kita mencarinya dengan cara membelah atau membagi dua buku itu. Jika kata yang dicari tidak terletak di halaman pertengahan itu, kita mencari lagi di belahan bagian kiri atau belahan bagian kanan dengan cara membagi dua belahan yang dimaksud. Begitu seterusnya sampai kata yang dicari ditemukan. Hal ini hanya bisa dilakukan jika kata-kata di dalam kamus sudah terurut.

Prinsip pencarian dengan membagi data atas dua bagian mengilhami algoritma pencarian bagidua. Data yang disimpan di dalam larik harus sudah terurut. Untuk memudahkan pembahasan, misalkan elemen larik sudah terurut menurun. Selama proses pencarian, kita memerlukan dua buah indeks larik, yaitu indeks terkecil dan indeks terbesar. Kita menyebut indeks terkecil sebagai indeks ujung kiri larik dan indeks terbesar sebagai indeks ujung kanan larik. Istilah "kiri" dan "kanan" dinyatakan dengan membayangkan elemen larik terentang horizontal (lihat Gambar 1.1(a)).

Misalkan indeks kiri adalah  $I_a$  dan indeks kanan adalah  $I_b$ . Pada mulanya,  $I_a = 1$  dan  $I_b = N$ .

**Langkah 1:** Bagi dua elemen larik pada elemen tengah. Elemen tengah adalah elemen dengan indeks  $k = (I_a + I_b) \div 2$   
(Elemen tengah,  $L[k]$ , membagi larik menjadi dua bagian, yaitu bagian kiri  $L[I_a..k-1]$  dan bagian kanan  $L[k+1..I_b]$ )

## Bab 1 : Pencarian

**Langkah 2:** Periksa apakah  $L[k] = X$ . Jika  $L[k] = X$ , pencarian dihentikan sebab  $X$  ditemukan. Tetapi, jika  $L[k] \neq X$ , harus ditentukan apakah pencarian akan dilakukan di bagian kiri atau di bagian kanan. Jika  $L[k] < X$ , maka pencarian dilakukan di bagian kiri. Sebaliknya, jika  $L[k] > X$ , pencarian dilakukan pada bagian kanan.

**Langkah 3:** Ulangi Langkah 1 sampai  $X$  ditemukan atau  $Ia > Ib$  (yaitu, ukuran larik sudah nol).

Contoh ilustrasi pencarian bagidua:

Misalkan diberikan larik  $L$  dengan delapan buah elemen yang sudah terurut menurun seperti di bawah ini:

81	76	21	18	16	13	10	7
$Ia=1$	2	3	4	5	6	7	$8=Ib$

(i) Misalkan elemen yang dicari adalah  $X = 18$ .

Langkah 1:

$Ia = 1$  dan  $Ib = 8$

Indeks elemen tengah  $k = (1 + 8) \text{ DIV } 2 = 4$  (diarsir).

81	76	21	18	16	13	10	7
1	2	3	4	5	6	7	8
kiri				kanan			

Langkah 2:

$L[4] = 18$ ? Ya! ( $X$  ditemukan, pencarian dihentikan)

(ii) Misalkan elemen yang dicari adalah  $X = 16$ .

Langkah 1:

$Ia = 1$  dan  $Ib = 8$

Indeks elemen tengah  $k = (1 + 8) \text{ DIV } 2 = 4$  (diarsir)

81	76	21	18	16	13	10	7
1	2	3	4	5	6	7	8
kiri				kanan			

Langkah 2:

$L[4] = 16$ ? Tidak!

Harus diputuskan apakah pencarian akan dilakukan di bagian kiri atau bagian kanan dengan pemeriksaan sbb:

$L[4] > 16$ ? Ya! Lakukan pencarian pada larik bagian kanan dengan  
 $Ia = k + 1 = 5$  dan  $Ib = 8$  (tetap)



## Bab 1 : Pencarian

16	13	10	7
la=5	6	7	8=Ib

Langkah 1':

la = 5 dan Ib = 8

Indeks elemen tengah k =  $(5 + 8) \text{ DIV } 2 = 6$  (diarsir)

16	13	10	7
5	6	7	8

kiri'                  kanan'

Langkah 2':

L[6] = 16? Tidak!

Harus diputuskan apakah pencarian akan dilakukan di bagian kiri atau di bagian kanan dengan pemeriksaan sbb:

L[6] > 16? Tidak! Lakukan pencarian pada larik bagian kiri dengan

Ia = 5 (tetap) dan Ib = k - 1 = 5

16
5

Langkah 1'':

Ia = 5 dan Ib = 5

Indeks elemen tengah k =  $(5 + 5) \text{ DIV } 2 = 5$  (diarsir)

16
5

Langkah 2'':

L[5] = 16? Ya! (X ditemukan, pencarian dihentikan)

(iii) Misalkan elemen yang dicari adalah X = 100

Langkah 1:

la = 1 dan Ib = 8

Indeks elemen tengah k =  $(1 + 8) \text{ DIV } 2 = 4$  (diarsir)

## Bab 1 : Pencarian

81	76	21	18	16	13	10	7
1	2	3	4	5	6	7	8

kiri                      kanan

Langkah 2:

$L[4] = 100$ ? Tidak!

Harus diputuskan apakah pencarian akan dilakukan di bagian kiri atau bagian kanan dengan pemeriksaan sbb:

$L[4] > 100$ ? Tidak! Lakukan pencarian pada larik bagian kiri dengan

$Ia = 1$  (tetap) dan  $Ib = k - 1 = 3$

81	76	21
$Ia=1$	2	$3=Ib$

Langkah 1':

$Ia = 1$  dan  $Ib = 3$

Indeks elemen tengah  $k = (1 + 3) \text{ DIV } 2 = 2$  (diarsir)

81	76	21
1	2	3

kiri'              kanan'

Langkah 2':

$L[2] = 100$ ? Tidak!

Harus diputuskan apakah pencarian akan dilakukan di bagian kiri atau bagian kanan dengan pemeriksaan sbb:

$L[2] > 100$ ? Tidak! Lakukan pencarian pada larik bagian kiri dengan

$Ia = 1$  dan  $Ib = k - 1 = 1$

81
1

Langkah 1":

$Ia = 1$  dan  $Ib = 1$

Indeks elemen tengah  $k = (1 + 1) \text{ DIV } 2 = 1$  (diarsir)

81
1



## Bab 1 : Pencarian

Langkah 2":

$L[1] = 100$ ? Tidak!

Harus diputuskan apakah pencarian akan dilakukan di bagian kiri atau di bagian kanan dengan pemeriksaan sbb:

$L[1] > 100$ ? Tidak! Lakukan pencarian pada larik bagian kiri dengan  
 $Ia = 1$  dan  $Ib = k - 1 = 0$

Karena  $Ia > Ib$ , maka tidak ada lagi bagian larik yang tersisa. Dengan demikian,  $X$  tidak ditemukan di dalam larik. Pencarian dihentikan.

```
procedure Bagidual(input L : Larik100, input N : integer, input X:integer,
                  input/output IX : integer)
    Mencari X di dalam larik L[1..N] yang sudah terurut menurun dengan metode
    pencarian bagidua. Keluaran prosedur ini adalah indeks IX yang L[IX]= X. IX
    bernilai 0 jika X tidak ditemukan.
    Awal : Larik L[1..N] sudah berisi data yang sudah terurut menurun, dan X
    adalah harga yang akan dicari
    Akhir: IX berisi indeks larik tempat X ditemukan, IX = 0 jika X tidak
    ditemukan }
```

VARIES LOKAL

$Ia, Ib$  : integer

$k$  : integer

ketemu : boolean

{ indeks elemen tengah}

{flag untuk menentukan ketemu atau tidak}

ALGORITMA

$Ia \leftarrow 1$

$Ib \leftarrow N$

ketemu  $\leftarrow$  false

while (not ketemu) and ( $Ia \leq Ib$ ) do

$k \leftarrow (Ia + Ib) \text{ div } 2$  { bagidua larik L pada posisi k }

if ( $L[k] = X$ ) then

ketemu  $\leftarrow$  true

else {  $L[k] \neq X$  }

if ( $L[k] > X$ ) then

{ akan lakukan pencarian pada larik bagian kanan, set indeks  
ujung kiri larik yang baru }

$Ia \leftarrow k + 1$

else

{ akan lakukan pencarian pada larik bagian kiri, set indeks  
ujung kanan larik yang baru }

$Ib \leftarrow k - 1$

endif

endif

endwhile

ketemu = true or  $Ia > Ib$

if (ketemu) then { X ditemukan }

$IX \leftarrow k$

else { X tidak ditemukan di dalam larik }

$IX \leftarrow 0$

endif