Pencarian

- Data seringkali dibutuhkan dalam pembacaan kembali informasi (*retrieval information*).
- Pencarian/searching → cara untuk mendapat data yang diinginkan dengan cara menelusuri data-data tersebut.
- Pencarian elemen data pada Java terbagi atas:
  - *Linier / sequential searching*
    → melakukan pengujian setiap item.
    → digunakan pada data yang masih acak & berurut.
  - *Binary searching*
    → dengan membagi dua data: (N+1) div 2
    → digunakan pada data yang berurut

Pencarian data pada array

- Tempat pencarian data dapat berupa array dalam memori, bisa juga pada file pada external storage
- *Searching* terbagi atas:
  - *Internal searching* – algoritma pencarian yang dilakukan dalam memori komputer.
  - *External searching* – algoritma pencarian yang melibatkan external media dan menambahkan ke memori utama.

Pencarian sequential

Proses pencarian dilakukan dengan membandingkan setiap elemen larik satu per satu secara beruntun, mulai dari elemen pertama sampai elemen yang dicari ditemukan atau seluruh elemen sudah diperiksa serta dapat dilakukan dalam kondisi data apapun.

Algoritma Pencarian sequential (1/2)

52	87	93	28	76	21
1	2	3	4	5	6

Nilai yang dicari: 28
Elemen yang diperiksa : 52 87 93 28
Index : 4

Nilai yang dicari: 52
Elemen yang diperiksa : 52
Index : 1

Nilai yang dicari: 55
Elemen yang diperiksa : 52 87 93 28 76 21
Index : 0

Algoritma Pencarian sequential (2/2)

```
1.  begin
2.  index ← 1
3.  while index <= n
4.    if key(cari) = key(index)
5.      result ← index
6.      goto 11
7.    end if
8.    index ← index + 1
9.  end while
10. hasil ← not found
11. end
```


Pencarian biner

- Pencarian Biner (Binary Search) dilakukan untuk :
  - memperkecil jumlah operasi pembandingan yang harus dilakukan antara data yang dicari dengan data yang ada di dalam tabel, khususnya untuk jumlah data yang sangat besar ukurannya.
  - Prinsip dasarnya adalah melakukan proses pembagian ruang pencarian secara berulang-ulang sampai data ditemukan atau sampai ruang pencarian tidak dapat dibagi lagi (berarti ada kemungkinan data tidak ditemukan).
- Syarat utama untuk pencarian biner adalah data di dalam tabel harus sudah terurut.

Algoritma Pencarian biner (1/3)

Elemen yang dicari 18

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

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

Langkah 1:
Low = 1 dan High = 8
Elemen tengah Middle = (1+8) div 2 = 9 div 2 = 4

Langkah 2:
Array[4] = X ? (18 = 18) true X ditemukan dan pencarian dihentikan.

Algoritma Pencarian biner (2/3)

Elemen yang dicari 16

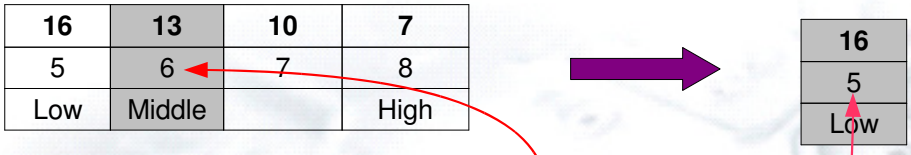
81	**76**	**21**	**18**	**16**	**13**	**10**	**7**
1	2	3	4	5	6	7	8
Low							High

Iterasi 1:

Langkah 1:
Low = 1 dan High = 8
Elemen tengah (Middle) = (1+8) div 2 = 9 div 2 = 4

Langkah 2:
Array[4] = X ? (18 = 16) FALSE, sehingga diputuskan pencarian di kiri atau dikanan.
Jika Array[4] > 16 ?, (18 > 16) TRUE, lakukan pencarian disebelah kanan dengan
Low = middle+1 (4+1 = 5) dan High = 8.

Algoritma Pencarian biner (3/3)



Iterasi 2:

Langkah 1:
Low = 5 dan High=8
Elemen tengah Middle = (5+8) div 2 = 13 div 2 = 6

Langkah 2:
Array[6] = X ? (13 = 16) FALSE untuk memutuskan pencarian di kiri atau di kanan.
Jika Larik[6] > 16 ?, (13 > 16) FALSE,
lakukan pencarian di sebelah KIRI --> low = 5 (TETAP) dan High = middle-1 = 5

Iterasi 3:

Langkah 1:
Low = 5 dan High=5
Elemen tengah Middle = (5+5) div 2 = 10 div 2 = 5

Langkah 2:
Array[5] = X ? (16 = 16) TRUE (X ditemukan , pencarian dihentikan)

Algoritma Pencarian biner (3/3)

```
1.   Mulai
2.   Awal ← 1
3.   Akhir ← n
4.   while Awal <= Akhir
5.     Tengah ← (Awal + Akhir) / 2
6.     if Kunci(Cari) = Kunci(Tengah)
7.       Hasil ← Tengah
8.       goto 15
9.     else if Kunci(Cari) > Kunci(Tengah)
10.      Awal ← Tengah + 1
11.    else Akhir ← Tengah - 1
12.    end if
13.  end while
14.  Hasil ← tidak ketemu
15.  Selesai
```

Pencarian dengan Sentinel

- Proses pencarian dengan Sentinel selalu menjamin bahwa elemen data yang dicari (x) pasti berhasil ditemukan.
- Untuk menyimpulkan apakah x ditemukan pada elemen sentinel atau bukan, yaitu dengan melihat nilai idx.
  - Jika idx = n+1, x ditemukan pada sentinel → x tidak terdapat di dalam array semula (sebelum penambahan sentinel).
  - jika idx < n +1, x ditemukan sebelum sentinel → x sudah ada di dalam larik L semula.

Ilustrasi Pencarian Sentinel

Elemen data dicari 35 & 28:

52	87	93	28	76	21	
1	2	3	4	5	6	

Untuk Sentinel

Di isi data sentinel sesuai dengan elemen yang dicari:

52	87	93	28	76	21	35
1	2	3	4	5	6	7

- 35 ditemukan pada elemen ke-n+1. Sentinel otomatis sudah ditambahkan ke dalam larik. Ukuran larik menjadi = 7.
- 28 ditemukan pada elemen ke-4. Sentinel batal menjadi elemen yang ditambahkan ke dalam larik. Ukuran larik tetap 6.

Algoritma Pencarian Sentinel (1/2)

```
1.   Mulai
2.   L[n + 1] ← x
3.       i ← 1
4.       while (L[i] ≠ x) do
5.           i ←  i + 1
6.       endwhile
7.       if idx = n + 1 then
8.           idx ← -1
9.       else
10.          idx ← i
11.      Endif
12.  Selesai
```


Selesai