ALGORITMA DAN PEMROGRAMAN

MATERI 5

- ALGORITMA PENCARIAN (*SEARCHING*)
- ALGORITMA PENCARIAN BERUNTUN (*SEQUENTIAL SEARCH*)
- ALGORITMA PENCARIAN BAGIDUA (*BINARY SEARCH*)

Oleh : Roni Sapto P., S. Kom.
E : ronisapto [at] gmail [dot] com
T : +62 821 16 75 93 57
W : mycampus.dezignwebster.com

ALGORITMA PENCARIAN (*SEARCHING*)

- **Proses Pencarian (*searching*)** adalah menemukan nilai (data) tertentu di dalam sekumpulan data yang bertipe sama.

- **Algoritma Pencarian Beruntun (*Sequential Search*)**
- **Algoritma Pencarian Bagidua (*Binary Search*)**

ALGORITMA PENCARIAN BERUNTUN
(SEQUENTIAL SEARCH)

- Merupakan algoritma pencarian yang paling sederhana.
- **Pencarian Lurus (*Linear Search*)** merupakan nama lain dari **Pencarian Beruntun**.
- Pencarian Beruntun adalah proses membandingkan setiap elemen larik satu per satu secara beruntun, mulai dari elemen pertama, sampai elemen yang dicari ditemukan, atau seluruh elemen sudah diperiksa.

ALGORITMA PENCARIAN BERUNTUN
LANGKAH KERJA

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

Langkah kerja :

1. Membandingkan data yang dicari, misal: *DC* = 18, dengan data pada larik (asumsi jumlah data pada larik = *n*), mulai dari indeks pertama (*i* = 1).
2. Bila pada indeks pertama bukan data yang dicari, lanjut pada indeks berikutnya, hingga data yang dicari ditemukan atau hingga indeks terakhir (*i* = *n*).

ALGORITMA PENCARIAN BERUNTUN

CONTOH ALGORITMA DALAM BENTUK PSEUDO-CODE

```
ALGORITMA Cari_beruntun
   {I.S. : data pada larik dan data yang dicari}
   {F.S. : mendapatkan indeks (idx) dari data yang
           dicari}
DEKLARASI :
   i, DC, idx : integer
   D : array [1..8] of integer

ALGORITMA :

Input (DC)
i ← 0
idx ← 0 {bila idx = 0 -> data belum ketemu}
WHILE (idx = 0) AND (i < 8) DO {asumsi jumlah data=8}
   i ← i + 1
   IF (D[i] = DC) THEN
        idx ← i
   ENDIF
ENDWHILE
```


ALGORITMA PENCARIAN BAGIDUA

(*BINARY SEARCH*)

- Merupakan pencarian pada **data terurut**.
- Merupakan algoritma pencarian pada data terurut yang **paling mangkus (*efficient*)**.
- Cara kerjanya adalah membagi dua area pencarian, lalu memeriksa di daerah mana data yang dicari berada. Pada daerah tersebut dibagi lagi menjadi dua area, lalu memeriksanya kembali. Begitu seterusnya hingga data ditemukan atau hingga area tidak dapat dibagi lagi.
- Karena adanya pemeriksaan area dimana data berada, maka **syaratnya adalah data terurut**.

ALGORITMA PENCARIAN BAGIDUA

LANGKAH KERJA

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

- Misal : diketahui sebuah larik *D* seperti gambar di atas, dengan jumlah data *n* = 8.
- Data yang akan dicari *DC* = 16.

Langkah kerja :

1. Tentukan batas kiri K_1 = 1 dan batas kanan K_2 = n
2. Periksa apakah data $D[K_1] = DC$. Bila ***TRUE*** maka pencarian berakhir (indeks DC ditemukan = K_1).
3. Bila langkah 2 bernillai ***FALSE***, maka periksa apakah data $D[K_2] = DC$. Bila ***TRUE*** maka pencarian berakhir (indeks DC ditemukan = K_2). Bila ***FALSE*** maka lanjut pada langkah 4.

ALGORITMA PENCARIAN BAGIDUA

LANGKAH KERJA (LANJUTAN)

4. Tentukan batas tengah $K_T = (K1 + K2)$ ***DIV 2***
5. Periksa apakah data $D[K_T] = DC$. Bila ***TRUE*** maka pencarian berakhir (indeks DC ditemukan = K_T).
6. Bila langkah 5 bernillai ***FALSE***, maka periksa apakah : $DC < D[K_T]$. (untuk menentukan data di daerah kiri atau kanan)
7. Ikuti langkah pada tabel sesuai kondisi.

		Data terurut :	
		Naik	Turun
$DC < D[K_T]$	TRUE	$K_2 = K_T$	$K_1 = K_T$
	FALSE	$K_1 = K_T$	$K_2 = K_T$

•Tabel ini khusus bila anda menggunakan persamaan *DC < D[KT]*

8. Ulangi mulai dari langkah 4 sampai data ditemukan ($D[K_T] = DC$) atau sampai $K_2 - K_1 = 1$.

ALGORITMA PENCARIAN BAGIDUA

CONTOH ALGORITMA 1 DALAM BENTUK PSEUDO-CODE

```
ALGORITMA Cari_bagidua_ascending
   {I.S. : data pada larik dan data yang dicari}
   {F.S. : mendapatkan indeks (idx) dari data yang
           dicari}
DEKLARASI :
   k1, k2, kt, DC : integer
   D : array [1..8] of integer
   ketemu : boolean

ALGORITMA :
INPUT (DC)
ketemu ← FALSE
k1 ← 1
k2 ← 8 {asumsi jumlah data = 8}
IF (D[k1] = DC) THEN
   ketemu ← TRUE
ELSE
   IF (D[k2] = DC) THEN
        ketemu ← TRUE
   ENDIF
ENDIF
```


ALGORITMA PENCARIAN BAGIDUA

CONTOH ALGORITMA 1 (LANJUTAN)

```
WHILE (NOT ketemu) AND (k2 - k1 > 1) DO
   kt = (k1 + k2) DIV 2
   IF (D[kt] = DC) THEN
      ketemu ← TRUE
   ELSE
      IF (DC < D[kt]) THEN
         k2 ← kt  {                              }
      ELSE        { asumsi data terurut naik  }
         k1 ← kt  {                              }
      ENDIF
   ENDIF
ENDWHILE
```

ALGORITMA PENCARIAN BAGIDUA

CONTOH ALGORITMA 2 DALAM BENTUK PSEUDO-CODE

```
ALGORITMA Cari_bagidua_descending
   {I.S. : data pada larik dan data yang dicari}
   {F.S. : mendapatkan indeks (idx) dari data yang
           dicari}
DEKLARASI :
   k1, k2, kt, DC : integer
   D : array [1..8] of integer
   ketemu : boolean

ALGORITMA :
INPUT (DC)
ketemu ← FALSE
k1 ← 1
k2 ← 8 {asumsi jumlah data = 8}
IF (D[k1] = DC) THEN
   ketemu ← TRUE
ELSE
   IF (D[k2] = DC) THEN
        ketemu ← TRUE
   ENDIF
ENDIF
```


ALGORITMA PENCARIAN BAGIDUA

CONTOH ALGORITMA 2 (LANJUTAN)

```
WHILE (NOT ketemu) AND (k2 - k1 > 1) DO
   kt = (k1 + k2) DIV 2
   IF (D[kt] = DC) THEN
      ketemu ← TRUE
   ELSE
      IF (DC < D[kt]) THEN
         k1 ← kt  {                          }
      ELSE        { asumsi data terurut turun }
         k2 ← kt  {                          }
      ENDIF
   ENDIF
ENDWHILE
```