ALGORITMA SEARCHING

Oleh : Agus Priyanto, M.Kom

INSTITUT TEKNOLOGI TELKOM
Smart, Trustworthy, And Teamwork

Tujuan Pembelajaran

- Setelah mengikuti kuliah ini, mahasiswa dapat memahami berbagai jenis algoritma searching

- Setelah mengikuti kuliah ini, mahasiswa dapat mengimplementasikannya algoritma searching dalam kasus nyata

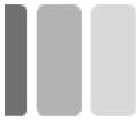
Introduction

- Pencarian (Searching) merupakan tindakan untuk mendapatkan suatu data dalam kumpulan data

- Searching adalah proses mendapatkan (retrieve) informasi berdasarkan kunci tertentu dari sejumlah informasi yang telah disimpan

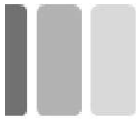
- **Contoh** implementasi pencarian dalam kehidupan sehari-hari :
  a. Menemukan nomor telepon seseorang pada buku telepon
  b. Mencari istilah dalam kamus

- **Contoh** implementasi pencarian pada aplikasi komputer :
  a. Untuk mendapatkan data dari seseorang mahasiswa (NIM, Nama, IPK)
  b. Mencari informasi suatu kata dalam kamus digital



- Untuk keperluan mencari data terdapat beragam **Algoritma Pencarian** (Search Algorithma)

- Yang dimaksud Algoritma Pencarian adalah **"algoritma yang menerima sebuah argumen a dan mencoba untuk menemukan sebuah rekaman yang memilki kunci a”**



- **Contoh**
  a. Siapa nama mahasiswa dengan NIM 13102056 ? single match
  b. Siapa saja yang mendapat nilai algoritma ≥ 85 ? multiple matches

- Pencarian yang dilakukan terhadap data, dapat dikelompokan sebagai berikut :
  a. **Pencarian Internal**
     Pencarian yang dilakukan terhadap data yang berada dalam memori komputer

  b. **Pencarian Eksternal**
     Pencarian yang dilakukan terhadap data yang berada dalam penyimpanan eksternal (hardisk)

Pencarian Sekuensial (Linier)

- **Pencarian Sekuensial (Linier)** merupakan model pencarian yang paling sederhana yang dilakukan terhadap kumpulan data

- Secara konsep, dapat dijelaskan sebagai berikut :
  a. Terdapat sebuah larik yang berisi n buah data (L[0], L[1], … , L[n-1])
  b. k adalah data yang dicari → L[i]=k

c. i adalah bilangan indeks terkecil , yang memenuhi kondisi $0 \leq k \leq n-1$

d. k adalah data yang dicari → L[i]=k

Contoh :

L ← [10, 9, 4, 6, 4, 3, 2, 5]

Dimanakan posisi 4 yang pertama ?
Dalam hal ini k adalah 4 dan k ditemukan pada posisi dengan indeks berupa 2

❑Algoritma

- Berikut ini merupakan implementasi algoritma pencarian secara sekuensial. Subrutin akan menghasilkan nilai balik berupa :

a. -1 jika data yang dicari tidak ditemukan dan

b. Bilangan antara 0 sampai dengan n-1 (dengan n adalah jumlah elemen larik) jika data yang dicari ditemukan

Pseudocode

```
SUBRUTIN cari (L, n, k)
    JIKA n ≤ 0 MAKA
            posisi ← - 1
    SEBALIKNYA
            ketemu ← SALAH
            i ← 0
            ULANG SELAMA (i<n-1) DAN (TIDAK ketemu)
               JIKA K = L[i] MAKA
                        posisi ← i
                        ketemu ← BENAR
               SEBALIKNYA
                        i ← i + 1
               AKHIR-JIKA
            AKHIR ULANG
            JIKA TIDAK ketemu MAKA
                        posisi ← -1
            AKHIR-JIKA
         AKHIR-JIKA
    NILAI-BALIK posisi
AKHIR-SUBRUTIN
```

L = larik
n = jml elemen larik
k = data yang dicari

```
int carilinier (int data [], int n, int k)
{
   int posisi, i, ketemu;
   if (n <= 0)
      posisi = -1;
   else
   {
      ketemu = 0;
      i = 1;
      while ((i < n -1) && ! ketemu)
         if (data [i] == k)
      {
         posisi = i;
         ketemu = 1;
      }
      else
         i++;
   if (!ketemu)
      posisi = -1;
   }
      return posisi;
}
```

Source code

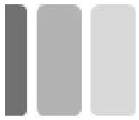
```
int main()
{
   int data [8] = {6, 7, 8, 5, 7, 8, 1, 9};
   int dicari = 5;

   cout << "Posisi " <<dicari<< " dalam larik
     data : "
       << carilinier(data, 8, dicari)<< "\n";
   return 0;
}
```

Pencarian Terhadap Data Terurut (Binary)

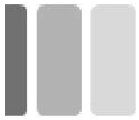
- Apabila kumpulan data sudah dalam keadaan terurut, pencarian data dengan menggunakan pencarian sekuensial akan memakan waktu yang lama jika jumlah data dalam kumpulan data tersebut sangat banyak.

- Untuk mengatasi hal tersebut terdapat algoritma yang dirancang agar pencarian lebih efesien yaitu pencarian biner (*Binary Search*)



- Pencarian biner dilakukan dengan membagi larik menjadi dua bagian dengan jumlah yang sama atau berbeda 1 jika jumlah data semula ganjil

- Data yang dicari kemudian dibandingkan dengan data terakhir pada bagian pertama



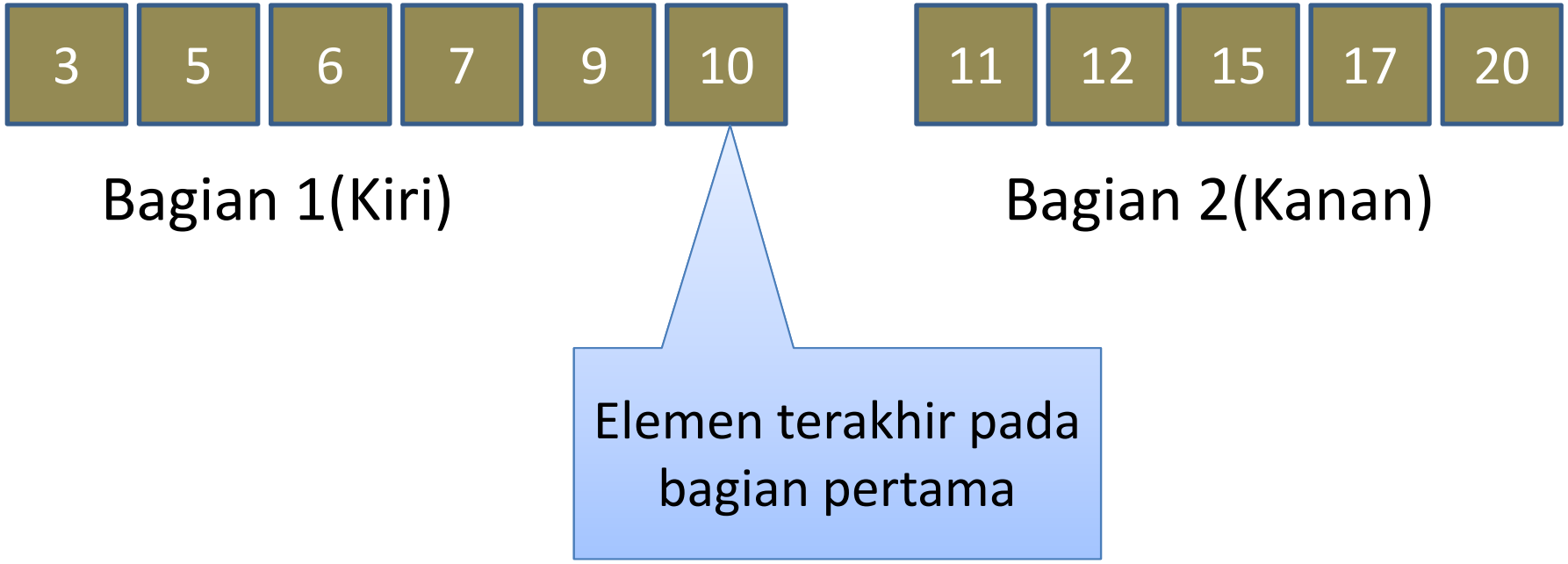
- Dalam hal ini akan terjadi 3 kemungkinan yang terjadi :
  a. Data yang dicari sama dengan elemen terakhir pada bagian pertama dalam larik. Jika kondisi ini terpenuhi, data yang dicari berarti ditemukan.
  b. Data yang dicari bernilai kurang dari nilai elemen terakhir pada bagian pertama dalam larik. Pada keadaan ini, pencarian diteruskan pada bagian pertama.
  c. Data yang dicari bernilai lebih dari nilai elemen terakhir pada bagian pertama dalam larik. Pada keadaan ini, pencarian diteruskan pada bagian kedua.

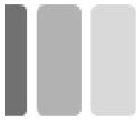
❑Contoh

Perhatikan gambar larik yang telah terurut dibawah ini :

Carilah : 12 ???

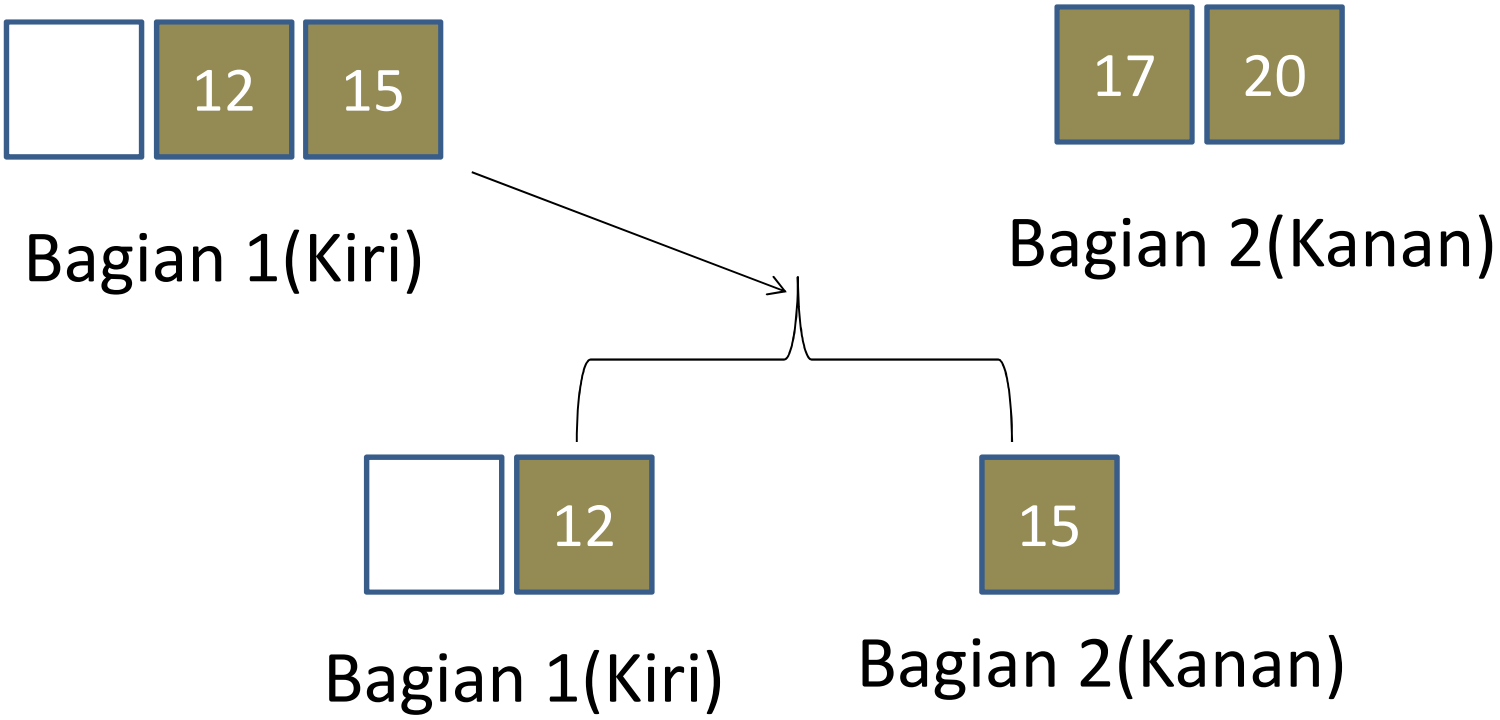
Mula-mula larik tersebut dipecah menjadi dua bagian seperti berikut

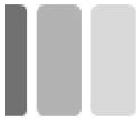




- Selanjutnya bilangan yang dicari (yaitu 12) dibandingkan dengan elemen terakhir pada bagian pertama pada larik (yaitu angka 10).

- Mengingat yang dicari lebih besar dari 10 maka pencarian diteruskan pada bagian kedua (bagian kanan)


12
15
17
20
Bagian 1(Kiri)
Bagian 2(Kanan)
12
15
Bagian 1(Kiri)
Bagian 2(Kanan)



- Langkah terakhir, angka yang dicari dibandingkan dengan elemen terakhir pada bagian pertama (yaitu angka 12)

- Melihat nilai yang dicari dan nilai elemen terakhir pada bagian pertama sama maka berarti pencarian telah ditemukan

Pseudocode

```
SUBRUTIN cari_biner (L, n, k)
    ada ← SALAH
    bawah ← 0
    atas ← n – 1
    ULANG SELAMA atas ≥ bawah
        tengah ← (atas + bawah) / 2  // Pembagi Bulat
        JIKA k >L[tengah] MAKA  // Data dicari dikanan
        bawah ← tengah + 1
        SEBALIKNYA
        JIKA k >L[tengah] MAKA // Data dicari dikiri
        atas = tengah – 1
        SEBALIKNYA
        ada ← BENAR
        posisi ← tengah
        bawah ← atas + 1  // Supaya perulangan berakhir
          AKHIR-JIKA
          AKHIR-JIKA
    AKHIR-ULANG
    JIKA TIDAK ada MAKA
        posisi ← - 1
    AKHIR-JIKA
    NILAI-BALIK posisi
AKHIR-SUBRUTIN
```

L = larik
n = jml elemen larik
k = data yang dicari

```
int caribiner (int data [], int n, int k)
{
   int ada, atas, bawah, tengah, posisi;
   ada = 0;
   bawah = 0;
   atas = n - 1;
   while (atas >= bawah)
   {
      tengah = (atas + bawah)/2 ;
      if (k > data [tengah])
         bawah = tengah + 1;
      else
      {
         ada = 1;
         posisi = tengah;
         bawah = atas + 1;
      }
   }
   if (!ada)
      posisi = -1;
      return posisi;
}
```

Source code

```
int main()
{
   int data [] = {1, 2, 4, 4, 5, 7, 8, 10, 13, 14,
     15};
   int dicari = 13;
   cout << "Posisi " <<dicari<< " dalam larik
     data : "
      << caribiner(data, 11, dicari)<< "\n";
   return 0;
}
```

Terimakasih