Fakultas Informatika
School of Computing
Telkom University

KUG1C3
Dasar Algoritma dan Pemrograman

Pencarian (Searching)

-RSM-

Pencarian pada Array/ Table Look Up

- Proses ini sering dilakukan terhadap sekumpulan data yang disimpan dalam tabel.
- Ada beberapa variasi pencarian. Suatu metoda yang dipakai dapat menentukan kecepatan pencarian.
- Untuk semua persoalan pencarian, dipakai kamus umum sebagai berikut :

```
KAMUS UMUM
    constant Nmax : integer = 100
    type TabInt : array [1..Nmax] of integer
    { jika diperlukan sebuah tabel, maka akan dibuat deklarasi
    sebagai berikut }
    T : TabInt { tabel integer }
    N : integer { indeks efektif, 1 ≤ N ≤ Nmax+1 }
```

Sequential Search pada Tabel Acak

- Diketahui sebuah tabel berisi harga integer TabInt [1..N], yang telah diisi.
- Tuliskanlah algoritma yang jika diberikan sebuah X bernilai integer akan mencari apakah harga X ada dalam TabInt secara berurutan mulai dari elemen pertama sampai ketemu atau sampai elemen terakhir.
- Algoritma akan menghasilkan harga indeks IX dimana X diketemukan pertama kalinya, IX diberi harga 0 jika pencarian tidak ketemu.
- Pencarian segera dihentikan begitu harga pertama diketemukan

Sequential Search pada Tabel Acak: Contoh 1

- N = 8, TabInt berisi : { 1, 3, 5, -8, 12, 90, 3, 5} ; X = 5
  Pemeriksaan dilakukan terhadap {1, 3, 5}
  Output : IX = 3

i	1	2	3	4	5	6	7	8	X = 5
TabInt	1	3	5	-8	12	90	3	5	

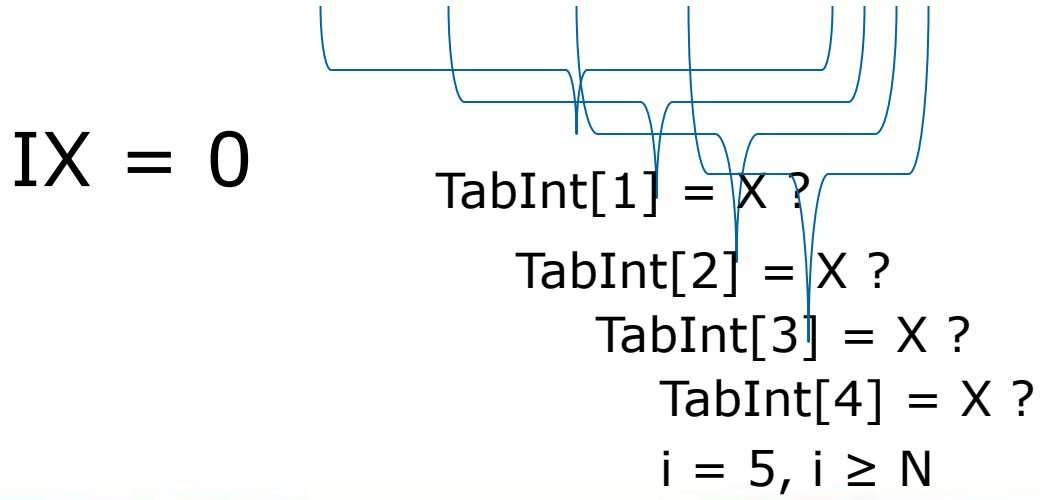
IX = 3

TabInt[1] = X ?
TabInt[2] = X ?
TabInt[3] = X ?

Sequential Search pada Tabel Acak: Contoh 2

- N = 4, TabInt berisi : { 11, 3, 5, 8} ; X = 100
  Pemeriksaan dilakukan terhadap {11, 3, 5, 8}
  Output : IX = 0

i	1	2	3	4	X = 100
TabInt	11	3	5	8	



Algoritma 1: Sequential Search tanpa Boolean

Procedure SEQSearchX1 (Input: TabInt: TabelInteger, Input N : integer,
Output: IX : integer, Input X : integer)
{ Mencari harga X dalam Tabel TabInt [1..N] secara berurutan mulai dari TabInt1}
{ Hasilnya adalah indeks IX dimana $TabInt_{ix} = X$ (i terkecil) }
{ IX = 0 jika tidak ketemu.}

```
Kamus   :
   i : integer  [1..Nmax]          {indeks untuk pencarian }
Algoritma :
   i ← 1
   while  (i < N) and (TabInt i   ≠  X) do
      i ← i + 1
   { i = N or TabInt i = X }
   depend on Tabint, i, X
      TabInt i = X :    IX  ← i
      TabInt i  ≠ X :   IX  ← 0
```

Algoritma 1

- Elemen tabel yang terakhir diperiksa secara khusus.
- Proses pada badan pengulangan hanya berisi "maju". Pencarian dihentikan karena ketemu, atau karena sudah tidak dapat "maju". Yang menentukan apakah pencarian ketemu atau tidak adalah kesamaan nilai elemen tabel ketika pencarian dihentikan.
- Pada versi ini tidak diperlukan nama boolean.

Algoritma 1a: Hasil Boolean

Procedure SEQSearchX1a (Input: TabInt: TabelInteger, Input N : integer,
Output: Found : boolean)
{ Mencari harga X dalam Tabel TabInt [1..N] secara berurutan mulai dari TabInt1}
{ Hasilnya adalah sebuah nilai boolean Found (true jika ketemu, false jika tidak. Pada versi ini informasi yang diperlukan hanaa ketemu atau tidak, namun kita tidak mendapatkan informasi DI MANA (indeks) nilai yang dicari diketemukan}

```
Kamus   :
   i : integer  [1..Nmax]          {indeks untuk pencarian }
   Found : boolean    {hasil pencarian, true jika ketemu, false jika tidak}
```

```
Algoritma :

   i ← 1

   while  (i < N) and (TabInt i   ≠  X) do

      i ← i + 1

   { i = N or TabInti = X }

   Found  ← (TabInti = X)
```

Algoritma 2: Sequential Search dengan Boolean

Procedure SEQSearchX2 (Input: TabInt: TabelInteger, N : integer, X : integer
Output: IX : integer, Found : boolean)
{ Mencari harga X dalam Tabel TabInt [1..N] secara berurutan mulai dari $TabInt_1$ }
{ Hasilnya adalah indeks IX dimana $TabInt_i$ =X (i terkecil) }
{ IX = 0 jika tidak ketemu dan sebuah boolean Found (true jika ketemu).}

```
Kamus   :
   i : integer  [1..Nmax+1]     {indeks untuk pencarian }
   Found : boolean    {Hasil pencarian }
Algoritma :
   Found   ←  false      { awal pencarian, belum ketemu ]
   i ←  1
   while  (i ≤ N) and (not  Found) do
      if (TabInt i = X) then
         Found ←   true
      else
         i ← i + 1
   { i > N or  Found }
   depend on Found
      Found  : IX  ←  i
      not Found:   IX  ←  0
```

Algoritma 2

- Pada versi ini, semua elemen tabel diperiksa dalam badan pengulangan dengan instruksi yang sama.
- Nilai tabel yang diperiksa indeksnya adalah dalam definisi tabel tersebut

Sequential Search pada Tabel Terurut

- Diketahui sebuah tabel bilangan integer TabInt [1..N], yang telah diisi, dan isinya terurut membesar (untuk setiap i ≤ [1..N-1], T[i] ≤ T[i+1]).
- Tuliskanlah algoritma, yang jika diberikan sebuah X bernilai integer akan mencari apakah harga X ada dalam TabInt secara berurutan mulai dari elemen pertama array sampai ketemu atau sampai elemen terakhir.
- Prosedur akan menghasilkan harga indeks IX dimana X diketemukan pertama kalinya, IX diberi harga 0 jika pencarian tidak ketemu.
- Pencarian segera dihentikan begitu harga pertama diketemukan.
- Contoh
  - N = 8, TabInt berisi : { 1, 3, 5, 8, 12, 90, 311, 500} ; X = 5
    Pemeriksaan dilakukan terhadap {1, 3, 5}
    Output : IX = 3
  - N = 7, TabInt berisi : { 11, 30, 50, 83, 99, 123, 456} ; X = 100
    Pemeriksaan dilakukan terhadap {11, 30, 50, 83, 99, 123}
    Output : IX = 0

Sequential Search pada Tabel Terurut: Algoritma

Program SEQSearchSorted (Input: TabInt: TabelInteger, N : integer, X : integer
Output: IX : integer, Found : boolean)

{ Mencari harga X dalam Tabel TabInt [1..N] secara berurutan mulai dari $TabInt_1$ }

{ Hasilnya adalah indeks IX dimana $TabInt_i = X$ (i terkecil) }

{ IX = 0 jika tidak ketemu. $TabInt_1 \leq X\ TabInt_2 \leq TabInt_3 \ldots \leq TabInt_i$ }

```
Kamus   :
   i : integer [1..Nmax] {indeks untuk pencarian }
Algoritma :
   i ← 1
   while  (i < N) and (TabInt i <  X) do
      i ← i + 1
   { i = N or TabInt I ≥ X }
   depend on Tabint, i, X
      TabInt i = X :   IX  ←  i
      TabInt i  ≠ X :   IX  ←  0       { TabInt i > X }
```

Sequential Search dengan Sentinel

- Dengan teknik sentinel, sengaja dipasang suatu elemen fiktif setelah elemen terakhir tabel, yang disebut SENTINEL. Elemen fiktif ini harganya adalah sama dengan elemen yang dicari.
- Jika nilai ditemukan, diperiksa lagi apakah ketemunya :
  - di antara elemen tabel yang sebenarnya
  - sesudah elemen terakhir (berarti tidak ketemu, karena elemen fiktif)
- Sentinel dapat diletakkan sebelum elemen pertama tabel atau sesudah elemen terakhir tabel tergantung kepada arah pencarian
- Jika pencarian dilakukan secara “maju”, maka sentinel diletakkan sesudah elemen terakhir tabel.
- Teknik sentinel sangat efisien, terutama jika pencarian dilakukan sebelum penyisipan sebuah elemen yang belum terdapat di dalam tabel.

Sequential Search dengan Sentinel: Kamus Umum

```
KAMUS UMUM
    constant  Nmax : integer  = 100
    type TabInt :  array  [1..Nmax+1] of integer
    N      : integer  {indeks efektif, maksimum tabel yang terdefinisi,
                       1 ≤ N ≤ Nmax+1}
{ jika diperlukan sebuah tabel, maka akan dibuat deklarasi sebagai berikut }
    T : TabInt { tabel integer }
    N : integer {indeks , 1 ≤ N ≤ Nmax+1}
```

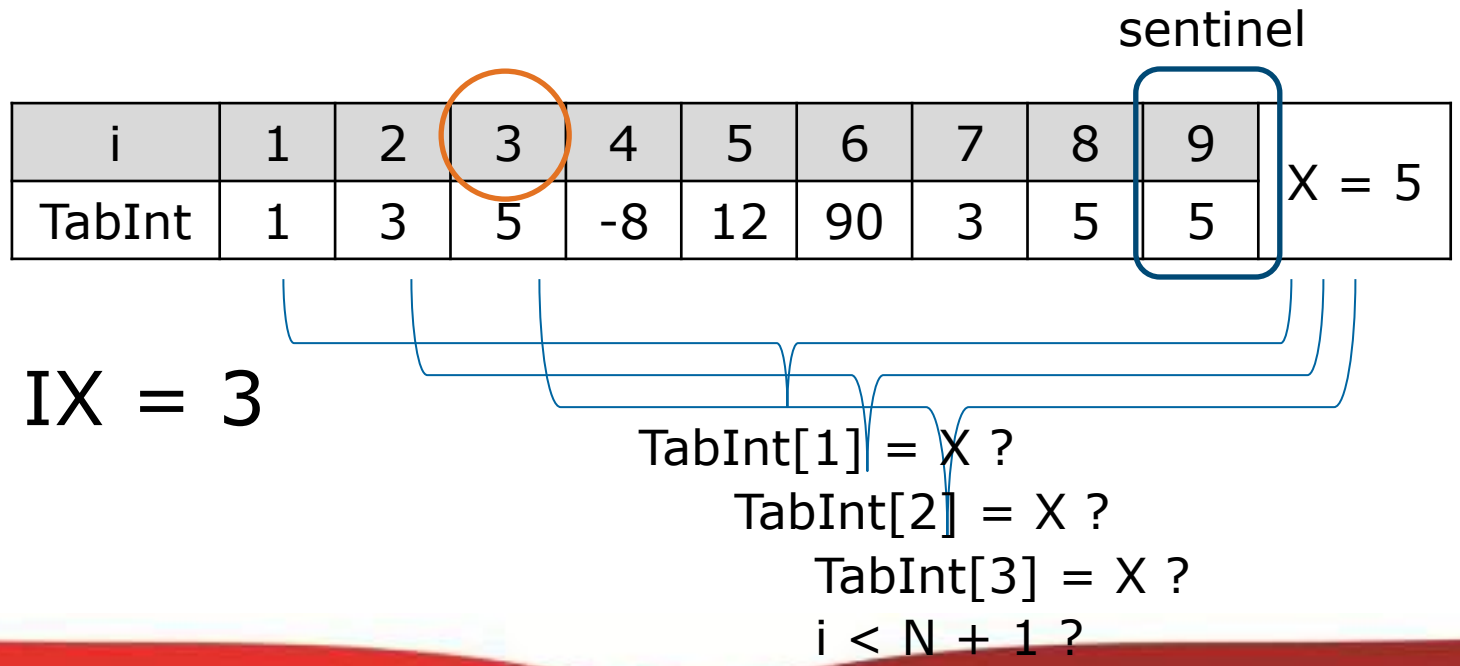
Sequential Search dengan Sentinel: Persoalan

- Diketahui sebuah tabel bilangan integer TabInt [1..N], yang telah diisi. Tuliskanlah sebuah algoritma, yang jika diberikan sebuah X bernilai integer akan mencari apakah harga X ada dalam TabInt secara berurutan.
- Prosedur akan menghasilkan harga indeks IX dimana X diketemukan pertama kalinya, IX diberi harga 0 jika pencarian tidak berhasil.
- Pencarian segera dihentikan ketika harga pertama diketemukan.

Sequential Search dengan Sentinel: Contoh 1

- N = 8, TabInt berisi : { 1, 3, 5, -8, 12, 90, 3, 5} ; X = 5

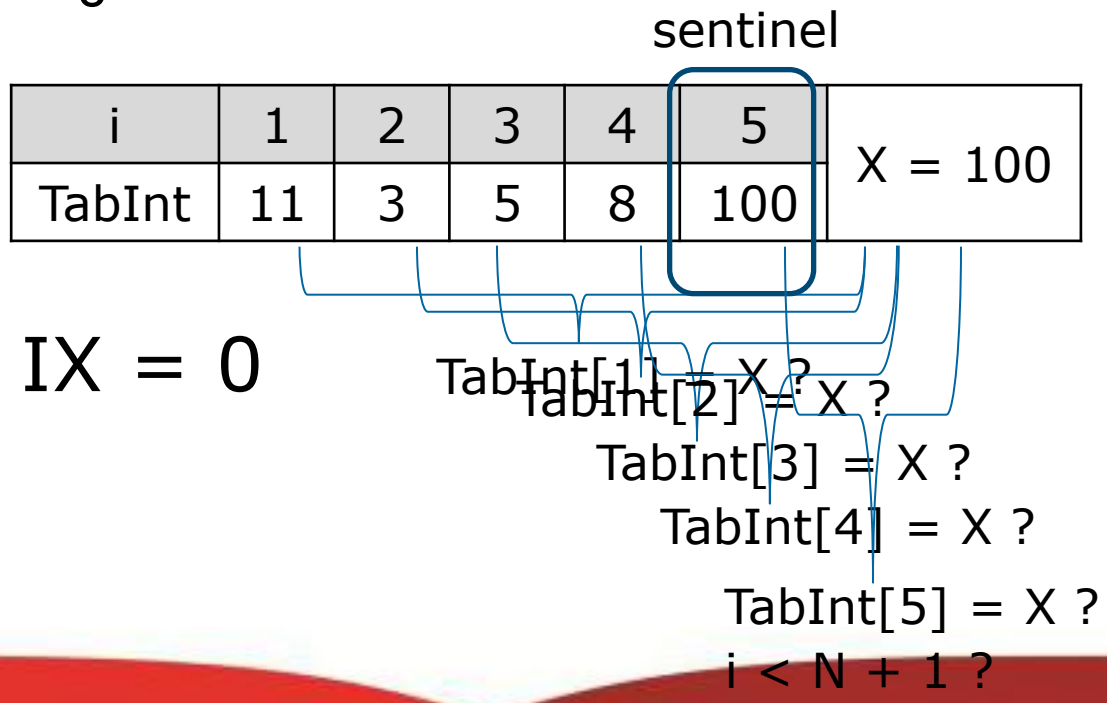
  TabInt dijadikan : { 1, 3, 5, -8, 12, 90, 3, 5, 5}
  Pemeriksaan dilakukan terhadap {1, 3, 5}
  Output : IX = 3



Sequential Search dengan Sentinel: Contoh 2

- N = 4, TabInt berisi : { 11, 3, 5, 8} ; X = 100

  TabInt dijadikan: { 11, 3, 5, 8, 100}
  Pemeriksaan dilakukan terhadap {11, 3, 5, 8}
  Output : IX = 0



Sequential Search dengan Sentinel: Algoritma

Program SEQSearchWithSentinel (Input: TabInt: TabelInteger, N : integer, X : integer
Output: IX : integer)
{ Mencari harga X dalam Tabel TabInt [1..N] secara berurutan mulai dari $TabInt_1$ }
{ Hasilnya adalah indeks IX dimana $TabInt_i$ =X (i terkecil) }
{ IX = 0 jika tidak ketemu. Sentinel diletakkan di $TabInt_{N+1}$ }

```
Kamus :
    i : integer [1..Nmax] {indeks untuk pencarian }
Algoritma :
    TabInt(N+1) ← X {pasang sentinel }
    i ← 1
    while (TabInt i ≠ X) do
    {tidak perlu test terhadap batas i, karena pasti berhenti}
        i ← i + 1
    { TabInt i = X; harus diperiksa apakah ketemunya di sentinel }
    depend on i, N
        i < N+1 : IX ← i {ketemu pada elemen tabel }
        i = N+1 : IX ← 0 {sentinel,berarti tidak ketemu }
```

Binary Search

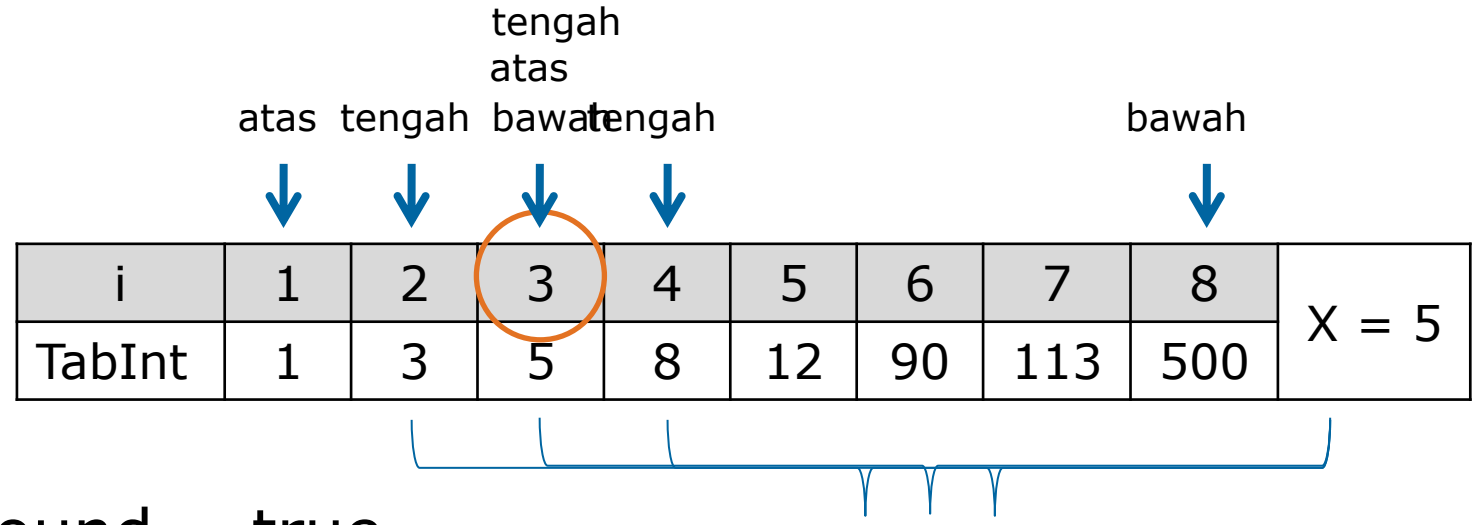
- Bekerja sebagai berikut : jika diberikan sebuah X bernilai integer akan mencari apakah harga X ada dalam TabInt secara dikhotomik, yaitu setiap saat pemeriksaan dilakukan dengan mereduksi elemen tabel, yaitu terhadap separuh dari elemen tabel:
  - bandingkan harga yang dicari dengan harga elemen tengah
  - jika sama, berarti ketemu
  - jika yang dicari lebih kecil, berarti pencarian dengan cara yang sama harus dilakukan terhadap elemen-elemen pada belahan atas.
  - jika harga yang dicari lebih besar, berarti pencarian dengan cara yang sama harus dilakukan terhadap elemen-elemen pada belahan bawah.
- Teknik ini hanya dapat diterapkan pada data yang terurut
- Pencarian dengan cara ini akan mengurangi waktu pencarian karena pembandingan harga direduksi secara logaritmik (dengan basis 2). Kecepatan pencarian sebanding dengan ln (N).

Binary Search: Persoalan

- Algoritma pencarian secara dikhotomik berikut akan menghasilkan sebuah harga boolean FOUND yang berharga true jika X ada dalam tabel, false jika tidak. Selain itu, harga indeks IX dimana X diketemukan pertama kalinya juga disimpan.
- Pencarian segera dihentikan begitu harga pertama diketemukan

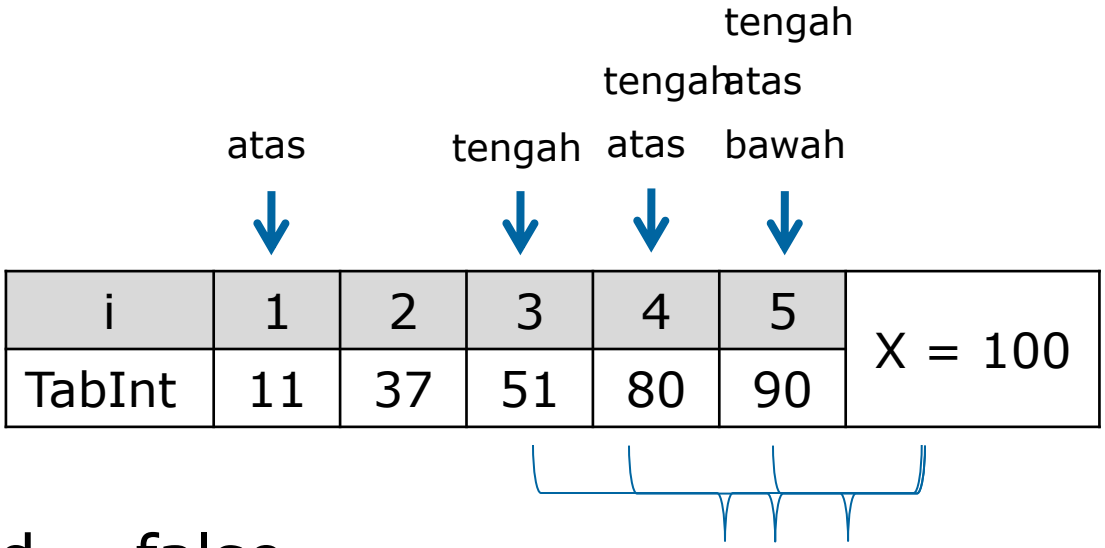
Binary Search: Contoh 1

- N = 8, TabInt berisi : {1, 3, 5, 8, 12, 90, 113, 500} ; X = 5
  Pemeriksaan dilakukan terhadap {8, 3, 5}
  Output : Found = true, IX = 3



Binary Search: Contoh 2

- N = 5, TabInt berisi : {11, 37, 51, 80, 90} ; X = 100
  Pemeriksaan dilakukan terhadap {51, 80, 90}
  Output : Found = false, IX = 0



Found = false
IX = 0

TabInt[tengah] = X ?
X > TabInt[tengah]
atas = tengah + 1

TabInt[tengah] = X ?
X < TabInt[tengah]
bawah = tengah - 1

atas = 6, atas ≥ bawah

Binary Search: Algoritma dengan Boolean

Procedure BinSearch1 (Input: TabInt: TabelInteger, N : integer, X : integer
Output: IX : integer, Found : boolean)
{ Mencari harga X dalam Tabel TabInt [1..N] secara dikhotomik }
{ Hasilnya adalahsebuah boolean Found, true jika ketemu.}
{Nilai elemen tabel terurut membesar: $TabInt_1 \leq X\ TabInt_2 \leq TabInt_3 \ ... \ \leq TabInt_N$ }

```
Kamus   :
    Found : boolean {true jika ketemu }
    Atas,Bawah, Tengah : integer
                              {indeks atas,bawah,tengah: batas pemeriksaan}
Algoritma :
    Atas ← 1;    Bawah ← N; {Batas atas dan bawah seluruh tabel}
    Found ← false  {Mula-mula belum ketemu }
    while  (Atas ≤ Bawah) and (not Found ) do
        Tengah ← (Atas+Bawah)  div 2
        depend on  TabInt, Tengah, X
            X= TabInt_Tengah   : Found  ←  true
            X < TabInt_Tengah  : Bawah ← Tengah  - 1
            X > TabInt_Tengah  :  Atas ←  Tengah + 1
    { Atas > Bawah  or  Found }
    { Harga Found menentukan hasil pencarian }
```

Binary Search: Algoritma dengan Boolean

- Semua pemeriksaan dilakukan dengan cara yang sama di dalam badan pengulangan.
- Algoritma tersebut berlaku untuk elemen tabel yang terurut membesar, dan harus dimodifikasi untuk elemen tabel yang terurut mengecil.

Binary Search: Algoritma tanpa Boolean

Procedure BinSearch1 (Input: TabInt: TabelInteger, N : integer, X : integer
Output: IX : integer, Found : boolean)
{ Mencari harga X dalam Tabel TabInt [1..N] secara dikhotomik }
{ Hasilnya adalah sebuah boolean Found, true jika ketemu.}
{Nilai elemen tabel terurut membesar: $TabInt_1 \leq X\ TabInt_2 \leq TabInt_3 \ ... \ \leq TabInt_N$ }

```
Kamus   :
    Atas,Bawah, Tengah : integer
                              {indeks atas,bawah,tengah: batas pemeriksaan}
Algoritma :
    Atas ← 1;    Bawah ← N; {Batas atas dan bawah seluruh tabel}
    Tengah ← (Atas+Bawah)  div 2
    Found ← false     {Mula-mula belum ketemu }
    while  (Atas < Bawah) and (X ≠ TabInt_Tengah) do
        depend on  TabInt, Tengah, X
            X < TabInt_Tengah  : Bawah ← Tengah  - 1
            X > TabInt_Tengah  :  Atas ←  Tengah + 1
        Tengah ← (Atas+Bawah)  div 2
    { Atas ≥ Bawah  or  (X = TabInt_Tengah) }
    Found ← (X = TabInt_Tengah)
```

Binary Search: Algoritma tanpa Boolean

- Pemeriksaan elemen terakhir dilakukan secara khusus di luar badan pengulangan.
- Perhatikan bagaimana indeks Tengah didefinisikan.

Referensi

- Liem, Inggriani. 2007. Diktat Kuliah Dasar Pemrograman (Bagian Pemrograman Prosedural). Bandung: Institut Teknologi Bandung

Fakultas Informatika
School of Computing
Telkom University

THANK YOU