Fakultas Informatika
School of Computing
Telkom University

CCH1A4 / Dasar Algoritma & Pemrogramanan

Yuliant Sibaroni M.T, Abdurahman Baizal M.Kom

KK Modeling and Computational Experiment

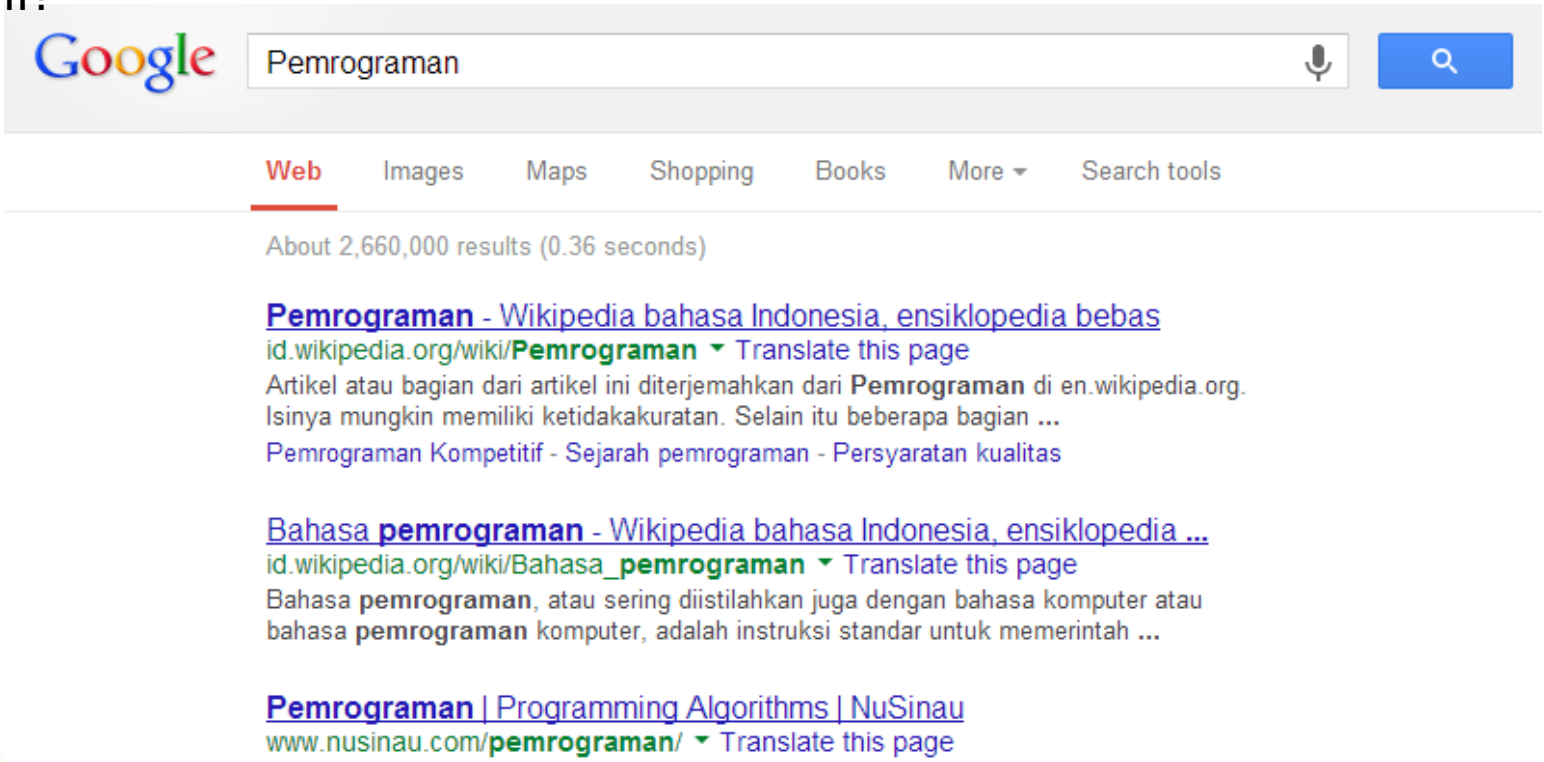
Pencarian dalam Tabel

- Pendahuluan
- Pencarian Sekuensial
- Pencarian Sekuensial Data Terurut
- Pencarian Biner

Pendahuluan

Pencarian pada Mesin Pencari

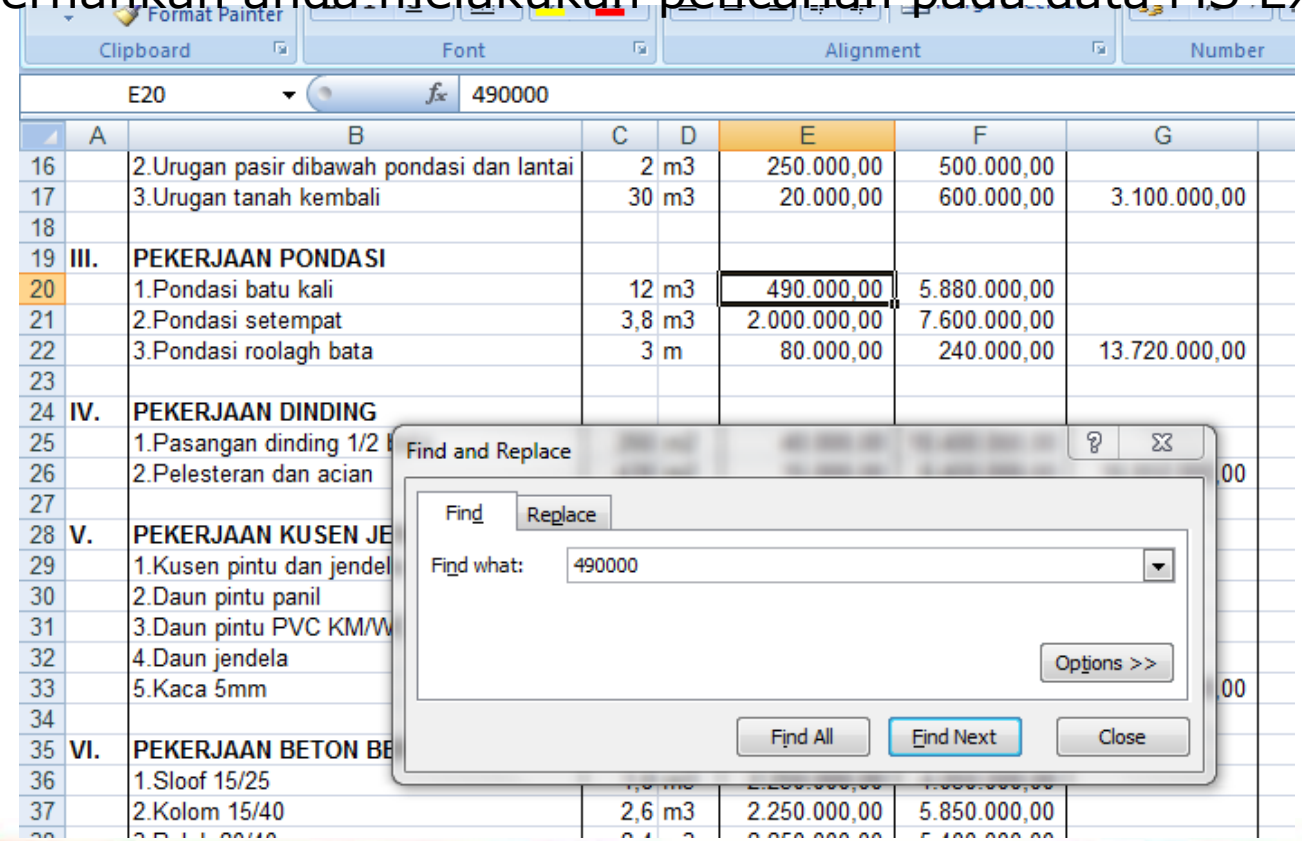
Pernahkah anda melakukan pencarian dengan mesin pencari seperti ini?



Pendahuluan

Pencarian pada Ms Excel

Pernahkah anda melakukan pencarian pada data MS Excel?



	A	B	C	D	E	F	G
16		2.Urugan pasir dibawah pondasi dan lantai	2	m3	250.000,00	500.000,00	
17		3.Urugan tanah kembali	30	m3	20.000,00	600.000,00	3.100.000,00
18							
19	III.	PEKERJAAN PONDASI					
20		1.Pondasi batu kali	12	m3	490.000,00	5.880.000,00	
21		2.Pondasi setempat	3,8	m3	2.000.000,00	7.600.000,00	
22		3.Pondasi roolagh bata	3	m	80.000,00	240.000,00	13.720.000,00
23							
24	IV.	PEKERJAAN DINDING					
25		1.Pasangan dinding 1/2					
26		2.Pelesteran dan acian					00
27							
28	V.	PEKERJAAN KUSEN JE					
29		1.Kusen pintu dan jendel					
30		2.Daun pintu panil					
31		3.Daun pintu PVC KM/W					
32		4.Daun jendela					
33		5.Kaca 5mm					00
34							
35	VI.	PEKERJAAN BETON BE					
36		1.Sloof 15/25					
37		2.Kolom 15/40	2,6	m3	2.250.000,00	5.850.000,00	

Pendahuluan

Ruang Lingkup

- Algoritma pencarian atau searching banyak dilakukan oleh kita sehari-hari, dan contoh penerapannya sangat banyak
- Algoritma pencarian atau searching pada dasarnya adalah pencarian nilai pada suatu himpunan data
- Kita akan mempelajari dasar dari algoritma pencarian
  - Pencarian secara sekuensial (beruntun): mencari nilai dari sekumpulan data dimulai dari elemen pertama hingga elemen terakhir data
  - Pencarian biner: pencarian dilakukan dengan membagi tabel menjadi 2 bagian

Pencarian Sekuensial

Definisi

Pencarian secara sekuensial (beruntun): Pencarian nilai dari sekumpulan data (tabel) dimulai dari elemen pertama hingga elemen terakhir data

Untuk semua persoalan pencarian, dipakai kamus umum sebagai berikut :

Kamus

```
Type Tabel : array[1..100] of integer
N : integer {maksimum tabel yang terdefinisi, 1<=N<=Nmax}
```

Pencarian Sekuensial

Contoh 10.1

Permasalahan

- Diketahui sebuah Tabel DataInt [1..N] berisi nilai integer, yang telah diisi
- Buatlah algoritma untuk mencari apakah harga X ada dalam DataInt
- Algoritma akan menghasilkan nilai indeks posisi dimana nilai X diketemukan pertama kalinya, dan jika pencarian tidak ketemu, mencetak "nilai yang dicari tidak ada".
- Pencarian segera dihentikan begitu harga pertama ditemukan
- Pencarian dihentikan karena ada kecocokan antara data[i] dengan X, atau karena sudah sampai pada data terakhir.
- Pada versi ini tidak diperlukan nama boolean

Pencarian Sekuensial

Contoh 10.1

```
Procedure SeqSearch1 (input DataInt : Tabel, X : integer,
                      Output pos : integer)
{Mencari nilai X pada tabel DataInt, versi tanpa Boolean
IS : harga X dan elemen tabel DataInt[1..N]sudah terdefinisi
FS : pos berisi indeks tabel, posisi X ditemukan, jika tdk
ketemu, pos bernilai -1 }
Kamus
  i : integer
Algoritma
  pos ← -1
  i ← 1
  while ((i < N) and (DataInt[i] <> X)) do
      i ← i + 1
  {(i > N) or (DataInt[i] = X)}
  if (DataInt[i] = X) then  {X ditemukan}
     pos ← i
```

Pencarian Sekuensial

Tabel Tracing

Misal, DataInt = {20, 12, 32, 5, 40} dan X = 51, berarti N=5

i	i< N	DataInt[i]<>X	(i<N) and (DataInt[i] <> X)	pos
				-1
1	true	true	true	-1
2	true	true	true	-1
3	true	true	true	-1
4	true	true	true	-1
5	false	true	false	-1

Pencarian Sekuensial

Contoh 10.2

```
Procedure SeqSearch2 (input DataInt : Tabel, X : integer,
                      Output pos : integer)
{Mencari nilai X pada tabel DataInt, versi dengan Boolean
IS : harga X dan elemen tabel DataInt[1..N]sudah terdefinisi
FS : pos berisi indeks tabel, posisi X ditemukan, jika tdk
ketemu, pos bernilai -1 }
Kamus
  i : integer; Ketemu: boolean
Algoritma
  pos ← -1 ; Ketemu ← false; i ← 1
  while ((i <= N) and (Not Ketemu) do
    if DataInt[i]=X then
         Ketemu ← true
         pos ← i
    else  i ← i + 1
  {(i = N) or Ketemu}
  if (Ketemu) then  {X ditemukan}
     pos ← i
```

Pencarian Sekuensial

Tabel Tracing

Misal, DataInt = {20, 12, 32, 5, 40} dan X = 32

i	Ketemu	i<=N	(i<N) and not ketemu	DataInt[i]=X	pos
	False				-1
1	False	True	True	False	-1
2	False	True	True	False	-1
3	False	True	True	True	-1
3	True	True	False		3

- Algoritma ini menggunakan variabel boolean untuk menentukan berhenti tidaknya proses pencarian
- Dalam praktek, teknik ini lebih sering digunakan, karena variabel boolean "ketemu" terasa lebih natural untuk menandakan apakah nilai sudah ditemukan

Pencarian Sekuensial

Contoh 10.3

```
Procedure Irisan (input A,B : Tabel, Output C: Tabel)
{IS : Terdefinisi tabel A dan B yang semua elemennya tidak ada
yang sama
FS : Elemen2 dari tabel C adalah irisan dari tabel A dan B }
Kamus Lokal
   i, j,JumIrisan: integer
Algoritma
  JumIrisan ← 0
  for i ← 1 to m do  {m = jumlah elemen A}
     temp ← A[i]
     for  j ← 1 to n do  {n = jumlah elemen B}
         if temp= B[j] then
            JumIrisan ← JumIrisan + 1
            C[JumIrisan] ← temp
  If JumIrisan = 0 then
     output('tidak ada irisan')
```

Pencarian Sekuensial

Contoh 10.4

- Diketahui sebuah tabel yang menyimpan data kota-kota yang menjadi sasaran pemasaran suatu produk. Diasumsikan semua nama kota yang ada dalam tabel adalah unik.
- Buatlah program untuk mencari, apakah sebuah kota tertentu ada dalam sasaran pemasaran produk.
- Karena data kota-kota bertipe string, maka didefinisikan sebuah tabel TabelString yang khusus untuk menyimpan elemen-elemen bertipe string

Kamus Global

```
Type TabelString : array[1..100] of string
N : integer {maksimum tabel yang terdefinisi,   1<=N<=100}
```

Pencarian Sekuensial

Contoh 10.4

```
Procedure SeqString (input DataKota:TabelString, Z:string, Output ada:boolean)
{IS : harga Z dan elemen tabel DataKota[1..N]sudah terdefinisi
 FS : ada bernilai true jika kota yang dicari (Z) ditemukan di DataKota,
false jika tidak ditemukan}
{versi dengan menggunakan variabel boolean}
Kamus Lokal
 i      : integer
 ketemu : boolean
Algoritma
 i ← 1
    ketemu ← false
    while ((i ≤ N) and (not ketemu) do
         if DataKota[i] = Z then
              ketemu ← true
         else
              i ← i + 1
    {(i > N) or (ketemu)}
    Ada ← ketemu
```

Pencarian Sekuensial Untuk Tabel Terurut

Contoh 10.5

- Terdefinisi sebuah tabel bilangan integer **TabInt**[1..N] yang isinya teurut membesar
- Buatlah algoritma untuk mencari nilai X dalam TabInt
- Prosedur akan menghasilkan nilai pos, dengan pos adalah indeks dimana X diketemukan pertama kalinya. Kalau tidak ketemu, pos diberi nilai -1
- Pencarian segera dihentikan begitu harga pertama diketemukan

Ilustrasi

N = 8, TabInt berisi : {2, 4, 6, 12, 14, 20, 32, 40},
X = 14
Pemeriksaan dilakukan terhadap {2, 4, 6, 12, 14}
Output : pos = 5
X = 5
Pemeriksaan dilakukan terhadap {2, 4, 6}
Output : pos = -1

Pencarian Sekuensial Untuk Tabel Terurut

Contoh 10.5

Detail dari prosedur SeqSearchUrut

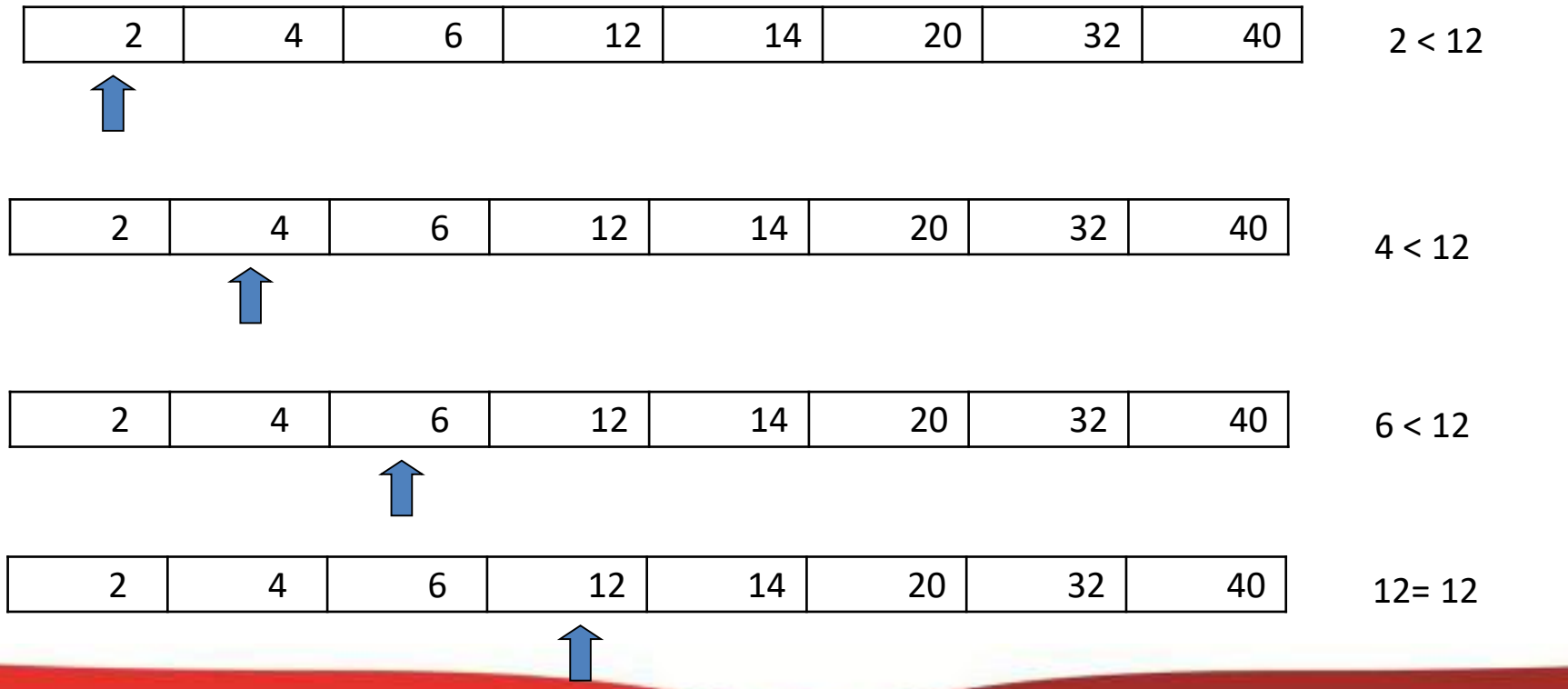
```
Procedure SeqSearchUrut (input Data:TabelInt, X:integer,Output pos : integer)
{Mencari apakah nilai X ada pada tabel DataInt yang terurut membesar
IS : DataInt[1..N]sudah terdefinisi, dengan elemen terurut membesar
FS : pos berisi indeks tabel, posisi X ditemukan}
Kamus Lokal
 i       : integer
Algoritma
 pos ← -1
 i ← 1
 while ((i < N) and (Data[i] < X)) do
   i ← i + 1
 {(I = N) or (Data[i] >= X)}
 if (Data[i] = X) then  {X ditemukan}
   pos ← i
```

Pencarian Sekuensial Untuk Tabel Terurut

Contoh 10.5

Ilustrasi

Data = {2, 4, 6, 12, 14, 20, 32, 40} dan X = 12



Pencarian Sekuensial Untuk Tabel Terurut

Contoh 10.6

- Diketahui sebuah tabel yang menyimpan data kota-kota yang menjadi sasaran pemasaran suatu produk. Diasumsikan semua nama kota yang ada dalam tabel adalah unik.
- Semua elemen tabel disusun secara terurut menaik (Ascending), misal :
  - {bandung, bogor, jakarta, jombang, klaten, magelang}
- Buatlah program untuk mencari, apakah sebuah kota tertentu ada dalam sasaran pemasaran produk.

Kamus Global

```
Type TabelString : array[1..100] of string
N : integer {maksimum tabel yang terdefinisi,   1<=N<=100}
```

Pencarian Sekuensial Untuk Tabel Terurut

Contoh 10.6

Detail prosedur StringSearch1

```
Procedure StringSearch1(input DataKota:TabelString,Z: string,Output ada:boolean)
{Mencari apakah nilai Z ada pada tabel DataKota yang terurut membesar
IS : DataKota[1..N]sudah terdefinisi, dengan elemen string terurut membesar
FS : variabel ada bernilai true jika kota Z yang dicari ada dalam tabel, false
jika tidak ada}
Kamus
   i : integer
Algoritma
    i ← 1
    while ((i < N) and (DataKota[i] < Z)) do
         i ← i + 1
    {(I = N) or (DataKota[i] >= Z)}
    if (DataKota[i] = Z) then
       ada ← true
    else
       ada ← false
```

Pencarian Sekuensial Untuk Tabel Terurut

Contoh 10.6

Penjelasan

- Dalam algoritma ini dapat dilihat, bahwa tidak ada perbedaan dalam pencarian untuk data integer serta data string
- Dalam penjelasan tipe string pada pertemuan minggu ke-2, sebenarnya tidak terlalu pas jika 2 buah string diperbandingkan
- Tetapi, umumnya bahasa pemrograman mempunyai kemampuan untuk membandingkan 2 nilai string, misal
  - 'aku' < 'kamu'
  - 'mereka' > 'dia'

Pencarian Sekuensial Untuk Tabel Terurut

Contoh 10.7

Penyisipan Data

- Buatlah sebuah prosedur untuk menyisipkan sebuah nilai X dalam sebuah **tabelInt** (**Data**) yang terurut membesar.
- Pada akhir penyisipan, **Data** tetap dalam keadaan terurut membesar.
- Pada awal proses, harus dilakukan searching untuk menentukan posisi penyisipan, kemudian dilakukan pergeseran elemen-elemen tertentu untuk menjaga keterurutan elemen

Pencarian Sekuensial Untuk Tabel Terurut

Contoh 10.7

Ilustrasi

12	14	16	28	30	35	40	52				

Disisipkan X = 32

Posisi sisip

12	14	16	28	30	35	35	40	52			

Dilakukan pergeseran untuk elemen setelah posisi sisip
Pergeseran : Data[6..8] dicopy ke Data[7..9]
X disisipkan pada posisi sisip, **hasilnya :**

12	14	16	28	30	32	35	40	52			

Pencarian Sekuensial Untuk Tabel Terurut

Contoh 10.7

Detail prosedur sisipkan

```
Procedure Sisipkan(input/Output Data : TabelInt Input X : integer)
{Menyisipkan nilai X pada tabelInt Data yang terurut membesar
IS : Data [1..N]sudah terdefinisi, dengan elemen terurut membesar
FS : Nilai X disipkan pada Data, dan Elemen2 Data tetap terurut membesar}
Kamus
   i, PosSisip, j : integer
Algoritma
 i ← 1
 while ((i =< N) and (Data[i] < X)) do {Cari posisi sisip}
    i ← i + 1
 {(i > N) or (Data [i] >= X)}
 PosSisip ← i
 if PosSisip <= N then
   for j← PosSisip to N do      {pergeseran elemen data}
       Data[j+1] ← Data[j]
 Data[PosSisip] ← X        {penyisipan elemen X}
```

Pencarian Biner

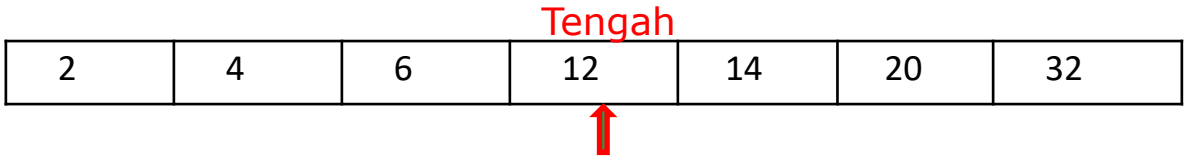
Alur Pencarian

- Pencarian biner merupakan teknik pencarian data dalam **tabel yang sudah terurut** berdasarkan algoritma berikut : setiap saat pemeriksaan dilakukan dengan mereduksi elemen tabel, yaitu menjadi 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 bagian bawah.
  - jika harga yang dicari lebih besar, berarti pencarian dengan cara yang sama harus dilakukan terhadap elemen-elemen pada bagian atas

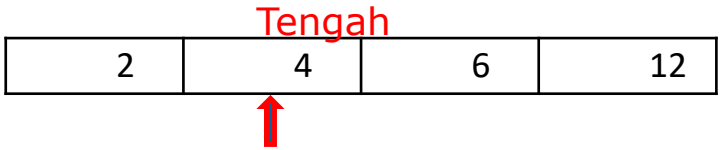
Pencarian Biner

Ilustrasi

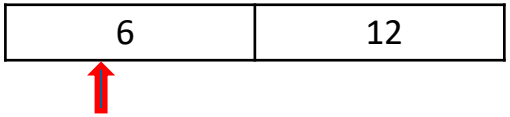
Akan dicari apakah 6 ada dalam tabel ?



1. Karena 6 < 12 maka 4 dicari pada bagian bawah



2. Karena 6 > 4 maka 6 dicari pada bagian atas



3. Split data tidak dilakukan lagi, karena 6=6, maka ketemu=true (selesai)

Pencarian Biner

Contoh 10.8

```
Procedure BinSearch(input Data:TabelInt,N,X:integer  output pos:integer)
IS : terdifinisi DataInt yang terurut membesar,
FS : dikeluarkan pos yang menandakan indeks posisi X ditemukan, pos bernilai
-1 jika tidak ketemu}
Kamus Lokal
 i,bawah,atas,tengah : integer
Algoritma
 bawah ← 1; atas ← N
 ketemu ← false;  pos← -1
 while (bawah<=atas) and not(ketemu) do
    tengah←(bawah+atas)div 2
    Depend on (Data [tengah],X)
       Data [tengah]=X : ketemu ← true
       Data [tengah]<X : bawah ← tengah+1
       Data [tengah]>X : atas ← tengah-1
 If ketemu then
    Pos ← tengah
```

Pencarian Biner

Tabel Tracing

Misal Data = {2, 4, 6, 12, 14, 20, 32, 40} dan X = 14

bawah	atas	ketemu	(bawah<=atas) and not(ketemu)	tengah	DataInt[Tengah]	pos
1	8	false	true			-1
5	8	false	true	4	12	-1
5	5	true	false	6	20	-1
						5

Pencarian Biner

Resume

Untuk algoritma pencarian, pencarian biner secara umum lebih efisien daripada pencarian sekuensial dengan asumsi data telah diurutkan

Contoh kasus
Data = {2, 4, 6, 12, 14, 20, 32, 40}
dan X = 14
Pada pencarian sekuensial, pembandingan dilakukan 5 kali, karena pencarian diurutkan dari depan

Pada pencarian biner, pembandingan dilakukan 3 kali

Dalam kasus terburuk, misal X = 40, pencarian sekuensial melakukan pembandingan 8 kali, sementara itu pencarian biner hanya melakukan 4 kali pembandingan

Soal Latihan

Soal 1

Diketahui sebuah tabel dengan elemen sebagai berikut sebagai berikut :
50 48 12 9 8 5 20 25
Jika tabel tersebut diurutkan secara ascending dan kemudiaan dilakukan searching sebuah nilai X= 26 dengan linier search, maka pembandingan dilakukan sebanyak :

A. 9 kali
B. 8 kali
C. 7 kali
D. 6 kali
E. 5 kali

Soal Latihan

Soal 2

Diketahui tabel sebagai berikut :
4 7 12 20 25 30 40 50
Jika dilakukan searching sebuah nilai X=7 dengan binary search, maka pembandingan dilakukan sebanyak :

A. 2 kali
B. 3 kali
C. 4 kali
D. 5 kali
E. 6 kali

Soal Latihan

Soal 3

Jika diketahui, sebuah tabel di berisi n buah data, jumlah maksimum pembandingan yang dilakukan untuk pencarian sekuensial adalah sebanyak ..... kali

A. n-1 kali
B. n kali
C. n/2 kali
D. n-2 kali
E. Tidak tentu

Soal 4

Diketahui sebuah tabel dengan elemen :
4 8 12 16 20 24 28 32

Jika dilakukan pencaria biner untuk sebuah nilai x, jumlah pembandingan terbanyak dilakukan pada pencarian nilai x = ..., sebanyak ..... kali

Jawab : 32 4 kali

Soal Latihan

Soal 4

Dibawah ini manakah pernyataan yang benar dan yang salah?

A. Secara umum, pencarian sekuensial untuk data terurut lebih efisien daripada untuk data yang tidak terurut B/S
B. Pencarian sekuensial pada data terurut ascending lebih efisien dari pada data terurut descending B/S
C. Secara umum, pencarian biner lebih efisien daripada pencarian sekuensial B/S

Soal Latihan

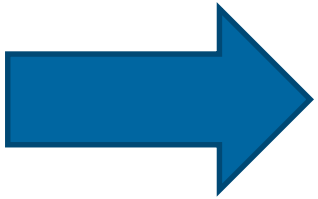
Soal 5

```
Procedure cari(input Tab:array[1..100] of integer,
                     n:integer,
                     X:integer)
{IS : terdefinisi TabMhs yang tidak kosong
 FS : mencetak nama mahasiswa berdasar nimcari}
```

```
Kamus
  i: integer
```

```
Algoritma
  i←1
  while (i<n) and (X<>Tab[i]) do
     if X=Tab[i] then
         output('ketemu di data ke', i )
     else
         output('belum ketemu')
  i←i+1
  endwhile
  output('proses selesai')
```

Lanjut Ke Halaman Berikutnya



SOAL LATIHAN

Jika Tabel Tab berisi elemen :

12 10 45 50 60

dan X = 45, maka hasil dari prosedur di atas akan tercetak :

A. belum ketemu
   belum ketemu
   proses selesai
B. belum ketemu
   belum ketemu
   ketemu di data ke 3
   proses selesai
C. belum ketemu
   belum ketemu
   ketemu di data ke 4
   proses selesai
D. belum ketemu
   ketemu di data ke 3
   proses selesai
E. Tidak ada jawaban yang memenuhi

REFERENSI

- Inggriani Liem, Diktat Kuliah IF223 Algoritma Dan Pemrograman, Jurusan Teknik Informatika Bandung, 1999

Fakultas Informatika
School of Computing
Telkom University

THANK YOU