JURUSAN
TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG

Searching

Tim Ajar Algoritma dan Struktur Data

2021

Searching

- Pada suatu data seringkali dibutuhkan pembacaan kembali informasi (*information retrieval*) dengan cara **searching**.
- Searching adalah **proses pencarian data** yang ada pada suatu deret data dengan cara menelusuri data-data tersebut.
- Tahapan paling penting pada searching: memeriksa jika data yang dicari sama dengan data yang ada pada deret data.
- Algoritma pencarian merupakan algoritma yang menerima suatu kata kunci sebagai kriteria pencarian, dan dengan langkah-langkah tertentu akan mencari data yang sesuai dengan kata kunci tersebut.

Searching

Hasil atau keluaran dari proses pencarian dapat berupa:

- Pesan
  - Ditemukan / Ada
  - Tidak ditemukan / Tidak ada
- Index array
  - `index = 13`
  - `i = 7`
  - `idx = -1` (jika data yang dicari tidak ditemukan)
- Nilai Boolean
  - TRUE
  - FALSE

Searching

- Macam algoritma pencarian :
  - Sequantial Search
  - Binary Search

Sequential Search

- Sequential Search atau disebut juga Linear Search adalah teknik pencarian data dimana data dicari secara **urut dari depan ke belakang** atau dari awal sampai akhir
- Proses pencarian dilakukan dengan **membandingkan** elemen array **satu per satu** secara beruntun mulai dari elemen pertama sampai elemen yang dicari sudah ditemukan atau sampai semua elemen sudah diperiksa

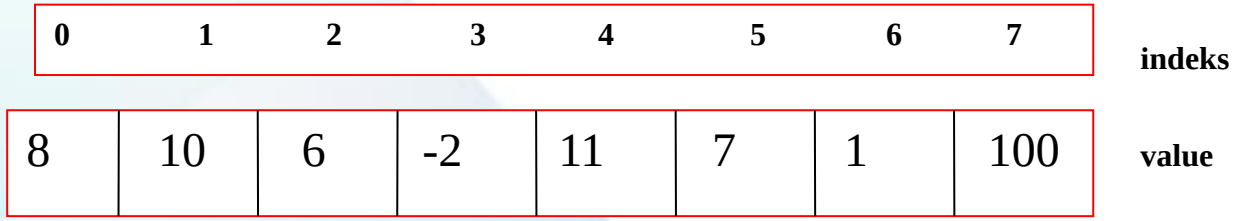
Algoritma Sequential Search

Secara umum, algoritma Sequential Search dijabarkan sebagai berikut:

1. Input **x** (data yang dicari)
2. Bandingkan x dengan data **ke-i sampai n** (n → jumlah elemen array)
3. Jika ada data yang sama dengan x maka cetak pesan “ditemukan”
4. Jika tidak ada data yang sama dengan x cetak pesan “tidak ditemukan”

Ilustrasi Sequential Search

- Misalnya terdapat array satu dimensi sebagai berikut:



- Kemudian program akan meminta data yang akan dicari, misalnya **6 (x = 6)**.
- Iterasi :

```
6 = 8 (tidak!)
6 = 10 (tidak!)
6 = 6 (Ya!) => output : "Ada" pada index ke-2
```

- Jika sampai data terakhir tidak ditemukan data yang sama maka output : " data yang dicari tidak ada".

Kelebihan

- Kumpulan data tidak harus dalam keadaan terurut
- Jika data yang dicari terletak di posisi depan, maka data akan ditemukan dengan cepat
- Penyisipan dan penghapusan elemen pada kumpulan data tidak mempengaruhi proses pencarian karena data tidak perlu diurutkan. Pada algoritma pencarian lainnya, data harus disusun kembali setelah adanya penyisipan atau penghapusan elemen
- Merupakan algoritma pencarian yang sangat sederhana, hemat sumber daya dan memori

Kekurangan Sequential Search

- Jika data yang dicari terletak di posisi belakang atau paling akhir, maka proses pencarian akan membutuhkan waktu yang lama
- Beban komputer akan semakin bertambah jika jumlah data dalam array sangat banyak, sehingga tidak cocok untuk data berukuran besar

Best & Worst Case Sequential Search

- **Best case** : jika data yang dicari terletak di depan sehingga waktu yang dibutuhkan minimal.
- **Worst case** : jika data yang dicari terletak di akhir sehingga waktu yang dibutuhkan maksimal.
- Contoh :

DATA = 5 6 9 2 8 1 7 4

bestcase ketika x = 5

worstcase ketika x = 4

*x = key/data yang dicari

Binary Search

- Teknik pencarian = data dibagi menjadi dua bagian untuk setiap kali proses pencarian.
- **Data awal** harus dalam kondisi **terurut.** Sehingga harus dilakukan proses sorting terlebih dahulu untuk data awal.
- Mencari posisi tengah :

Posisi tengah = (posisi awal + posisi akhir) / 2

Algoritma Binary Search

1. Data diambil dari posisi awal 1 dan posisi akhir N
2. Kemudian cari posisi data tengah dengan rumus: **(posisi awal + posisi akhir) / 2**
3. Kemudian data yang dicari dibandingkan dengan data yang di tengah, apakah sama atau lebih kecil, atau lebih besar?
4. Jika data sama, berarti ketemu.
5. Jika lebih besar, maka ulangi langkah 2 dengan posisi awal adalah **posisi tengah + 1**
6. Jika lebih kecil, maka ulangi langkah 2 dengan posisi akhir adalah **posisi tengah – 1**

Ilustrasi 1 Binary Search

Contoh Data:

Misalnya data yang dicari **23 (X = 23)**

Iterasi 1

m(tengah) = (0+8)/2 =4

Index	0	1	2	3	4	5	6	7	8
Data	3	9	11	12	15	17	23	31	35
posisi	i				m				j

Apakah 15 = 23? tidak

apakah 23 > 15 ? Ya , maka : awal (l)= tengah + 1

Ilustrasi 1 Binary Search

Iterasi 2

m(tengah) = (5+8)/2=6

Index	0	1	2	3	4	5	6	7	8
Data	3	9	11	12	15	17	23	31	35
posisi						i	m		j

Apakah

X = m / 23=23 (sama dengan data tengah).

Output = "Data ditemukan"

Ilustrasi 2 **Binary Search**

- Misalkan diberikan array arr dengan delapan buah elemen yang sudah terurut menurun seperti di bawah ini

arr	**81**	**76**	**21**	**18**	**16**	**13**	**10**	**7**
index	0	1	2	3	4	5	6	7

- **Misalkan elemen yang dicari adalah x = 16.**

Ilustrasi 2 **Binary Search**

- **Langkah 1:**
- i = 0 dan j = 8 Indeks elemen tengah **m = (0+ 7) div 2 = 3** (elemen dalam kolom abu-abu)

arr	**81**	**76**	**21**	**18**	**16**	**13**	**10**	**7**
index	0	1	2	3	4	5	6	7
		kiri		m		kanan		

Ilustrasi 2 **Binary Search**

Langkah 2:

- Pembandingan: arr[3] = 16 Tidak! Harus diputuskan apakah pencarian akan dilakukan di bagian kiri atau di bagian kanan dengan pemeriksaan sebagai berikut:
- Pembandingan: arr[3] > 16? →18>16? Ya! Lakukan pencarian pada array bagian kanan dengan **i = k + 1 = 4** dan **j = 7 (tetap)**

16	**13**	**10**	**7**
4	5	6	7
		kanan	

Ilustrasi 2 **Binary Search**

- Kembali ke langkah 1 untuk mencari nilai tengah
- i = 4 dan j = 7 Indeks elemen tengah **k = (4 + 7) div 2 = 5** (elemen yang diberi warna abu-abu)

16	**13**	**10**	**7**
4	5	6	7
kiri		kanan	

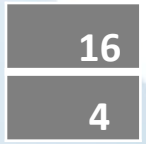
Ilustrasi 2 **Binary Search**

- Kembali melakukan langkah 2. Melakukan perbandingan
- Pembandingan: arr[5] = 16 →13=16? Tidak! Harus diputuskan apakah pencarian akan dilakukan di bagian kiri atau di bagian kanan dengan pemeriksaan sebagai berikut:
- Pembandingan: arr[5] > 16? 13>16? Tidak! Lakukan pencarian pada array bagian kiri dengan i = 4 (tetap) dan j = k - 1 = 4

16
4

Ilustrasi 2 **Binary Search**

- Kembali ke langkah 1 untuk mencari nilai tengah
- i = 4 dan j = 4 Indeks elemen tengah **k = (4 + 4) div 2 = 4 (**elemen yang berwarna abu)



- Kembali melakukan langkah 2. Melakukan perbandingan
- arr[4] = 16? Ya! (x ditemukan, proses pencarian selesai)

Best & Worst Case **Binary Search**

- **Best case** : jika data yang dicari terletak di posisi tengah.
- **Worst case** : jika data yang dicari tidak ditemukan.
- Contoh :

DATA = 5 6 9 2 8 1 7 4 3

bestcase ketika x = 5 (T(n)=1)

worstcase ketika x = 25 (T(n) = 5 atau n/2)

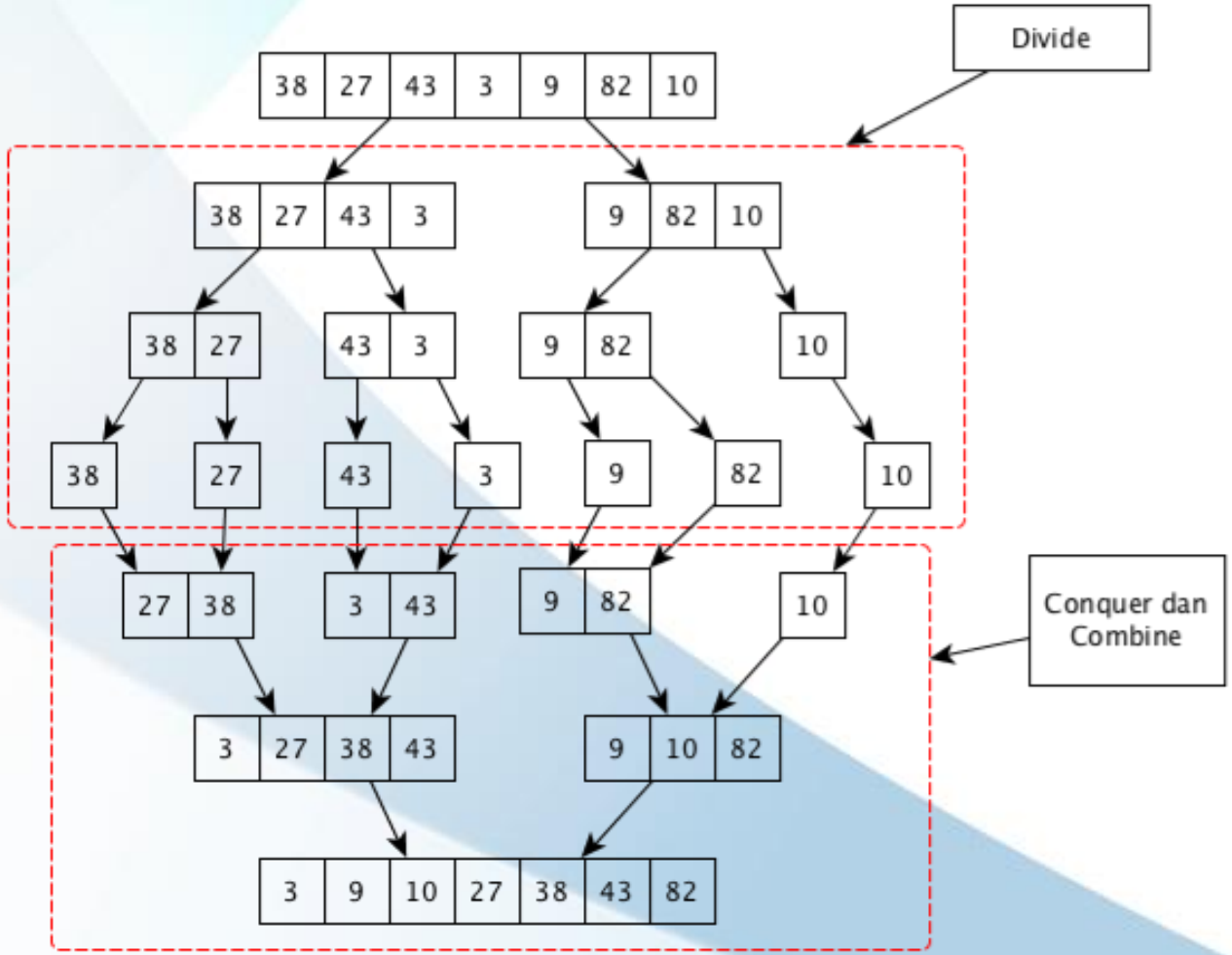
***x = key/data yang dicari**

Pengayaan Divide n Conquer (Merge Sort)

- Pengurutan dengan metode ini sering juga disebut dengan metode Divide and Conquer.
- Metode ini terdiri dari 3 tahapan.
  1. Divide membagi permasalahan atau koleksi data ke dalam bagian-bagian yang lebih kecil.
  2. Conquer mengurutkan dari bagian yang paling kecil.
  3. Dan tahapan yang terakhir yaitu Combine, mengkombinasikan atau menggabungkan solusi dari bagian yang paling kecil sehingga menjadi solusi utama.

Ilustrasi Merge Sort (Ascending) – 1

JURUSAN
TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG



Latihan

1. Buatlah flowchart dari algoritma binary search!
2. Buatlah flowchart dari algoritma sequential search!
3. Jika terdapat array {20,35,14,7,67,89,23,46}
   - Data yang ingin dicari (x= 35)
   - Gambarkan proses penyelesaian kasus pencarian dengan sequential search
   - Gambarkan proses penyelesaian kasus pencarian dengan binary seach (urutkan dahulu array nya dengan algoritma sorting)