

shift

Dafi Hazzan A. H  
A11.2024.15851    A11.4208

No.

Date 16 . 04 . 25

= Sorting =

Tugas Resume 21 Pro

Proses Pengurutan data acak menjadi teratur (ASC/DES)

### 1. bogo Sort

• konsep: Mengacak data hingga terurut, tapi tidak efisien

• contoh: array awal:  $[3, 2, 5, 1, 0, 4]$   
Random 1:  $[4, 5, 0, 3, 2, 1]$   
Random 2:  $[4, 1, 3, 2, 5, 0]$   
Random ke-n:  $[0, 1, 2, 3, 4, 5] \rightarrow$  terurut } belum urut

### 2. bubble Sort

• konsep: bandingkan dan tukar elemen berdekatan hingga tidak ada pertukaran

• contoh: array:  $[5, 1, 4, 2, 8]$

↳ tahap 1:  $5 > 1 \rightarrow$  tukar  $\rightarrow [1, 5, 4, 2, 8]$   
 $5 > 4 \rightarrow$  tukar  $\rightarrow [1, 4, 5, 2, 8]$   
 $5 > 2 \rightarrow$  tukar  $\rightarrow [1, 4, 2, 5, 8]$   
 $5 < 8 \rightarrow$  tetap

↳ tahap 2:  $1 < 4 \rightarrow$  tetap

$4 > 2 \rightarrow$  tukar  $\rightarrow [1, 2, 4, 5, 8] \rightarrow$  terurut

### 3. selection Sort

• konsep: cari nilai terkecil, lalu tempatkan di posisi awal

• contoh: array:  $[29, 10, 14, 37, 13]$

↳ tahap 1: cari yang terkecil (10)  $\rightarrow$  tukar dengan indeks 0  $\rightarrow [10, 29, 14, 37, 13]$

↳ tahap 2: cari yang terkecil (13) di sisa array  $\rightarrow$  tukar dengan indeks 1  
 $\rightarrow [10, 13, 14, 37, 29]$

↳ tahap 3: cari yang terkecil (14)  $\rightarrow$  sudah di posisi tepat

↳ tahap 4: cari yang terkecil (29)  $\rightarrow$  tukar dengan indeks 3  $\rightarrow$

$[10, 13, 14, 29, 37]$

#### 4. insertion sort

• konsep: sisipkan elemen ke posisi tepat dalam subset terurut

• contoh: array: [12, 11, 13, 5, 6]

- tahap 1: ambil 11  $\rightarrow$  bandingkan dengan 12  $\rightarrow$  sisipkan didepan  $\rightarrow$

[11, 12, 13, 5, 6]

- tahap 2: ambil 13  $\rightarrow$  sudah benar

- tahap 3: ambil 5  $\rightarrow$  geser 13, 12, 11  $\rightarrow$  [5, 11, 12, 13, 6]

- tahap 4: ambil 6  $\rightarrow$  geser 13, 12, 11  $\rightarrow$  [5, 6, 11, 12, 13]

#### 5. Merge Sort

• konsep: bagi, urutkan, gabung

• contoh: array: [38, 27, 43, 3]

- tahap 1: bagi menjadi [38, 27] dan [43, 3]

- tahap 2: urutkan tiap bagian  $\rightarrow$  [27, 38] dan [3, 43]

- tahap 3: gabung dengan membandingkan elemen  $\rightarrow$  [3, 27, 38, 43]

= Searching =

#### 1. linear search

• konsep: cek data satu per satu dari awal hingga akhir

• contoh: array: [12, 15, 17, 20, 50] cari 15

↳ langkah: bandingkan 12  $\neq$  15  $\rightarrow$  15  $=$  15  $\rightarrow$  ditemukan di indeks 1

#### 2. binary search

• konsep: bagi array terurut menjadi dua bagian, bandingkan dengan elemen tengah

• contoh: array: [12, 15, 17, 20, 50] cari 15

- tahap 1: batas bawah (L=0), batas atas (H=4).  $mid = (0+4)/2 = 2$   
 $\rightarrow$  nilai 17

- tahap 2: karena 15 < 17, cari di kiri (L=0, H=1)

- tahap 3: mid baru =  $(0+1)/2 = 0 \rightarrow$  nilai 12

- tahap 4: karena 15 > 12, cari di kanan (L=1, H=1).  $mid = 1 \rightarrow$  nilai 15  $\rightarrow$  ditemukan

= linked list =

Pengertian

- linked list adalah salah satu bentuk struktur data, berisi kumpulan data (node) yang tersusun secara sekuensial, saling terkait, dan dinamis.

• Sejarah

\* dikembangkan tahun 1955-1956 oleh Allen Newell, Cliff Shaw, dan Herbert Simon di RAND Corporation sebagai struktur data utama untuk bahasa Information Processing Language (IPL) yang dibuat untuk mengembangkan program artificial intelligence.

\* Victor Yngve di MIT (Massachusetts Institute of Technology) menggunakan linked list untuk natural language processing dan machine transitions.

\* Perbedaan linked list vs array

Aspek	Linked list	Array
Alokasi memori	dinamis (sesuai kebutuhan)	statik (ukuran tetap)
Akses data	sequensial (tidak langsung)	langsung (indeks)
Fleksibilitas	mudah ditambah / dihapus	kaku (ukuran tetap)
Penggunaan memori	lebih besar (simpan pointer)	lebih efisien

1. Single linked list

• merupakan struktur data linier dimana setiap node berisi dua bagian:

- Data: nilai/informasi yang disimpan pada node
- Next pointer: referensi/alamat ke node berikutnya

\* Single linked list non circular

• karakteristik:

- node terakhir memiliki pointer yang bernilai NULL, menandakan akhir dari list
- traversal dimulai dari node pertama (head) dan berhenti ketika pointer mencapai NULL

### • kelebihan :

- konsep sederhana dan mudah diimplementasikan
- penggunaan memori lebih sedikit karena hanya menyimpan satu pointer per node

### • kekurangan :

- proses traversal tidak dapat berulang tanpa memulai dari awal
- jika ingin mengakses akhir list, harus melakukan iterasi dari awal sehingga tidak bisa langsung mengakses node terakhir

## \* Single linked list circular

### • karakteristik :

- node terakhir tidak menunjuk ke NULL, melainkan menunjuk kembali ke node pertama (head), membentuk sirkular
- list ini tidak memiliki "akhir" secara eksplisit, sehingga traversal bisa dilakukan secara kontinu (selalu berputar)

### • kelebihan :

- sangat cocok untuk aplikasi yang membutuhkan pengulangan terus-menerus
- memudahkan dalam mengimplementasikan algoritma tertentu yang memerlukan siklus

### • kekurangan :

- perlu penanganan khusus agar tidak terjadi looping tak terbatas ketika melakukan traversal
- operasi seperti pencarian elemen / pembalikan (reverse) harus diimplementasikan dengan sangat hati-hati mengingat tidak ada acuan "akhir" (NULL)

No.

Date

## 2. Double linked list

- merupakan struktur data linier dimana setiap node berisi tiga bagian:
  - Data : nilai / informasi pada node
  - next pointer : pointer yang menunjuk ke node berikutnya
  - previous pointer : pointer yang menunjuk ke node sebelumnya

### \* Double linked list non circular

- karakteristik:
  - Node pertama memiliki pointer prev bernilai NULL
  - node terakhir memiliki pointer next bernilai NULL
  - memungkinkan traversal ke depan maupun ke belakang
- kelebihan:
  - mempermudah operasi insert dan delete, terutama ketika ingin mengakses node sebelumnya
  - traversal bolak balik sehingga cocok untuk aplikasi seperti browser history dan navigasi data
- kekurangan:
  - membutuhkan lebih banyak memori karena setiap node menyimpan dua pointer
  - operasi pembaruan pointer harus dilakukan dengan hati-hati (contoh saat penghapusan, perlu mengupdate kedua pointer dari node tetangga)

### \* Double linked list circular

- karakteristik:
  - Node terakhir menunjuk ke node pertama melalui pointer next dan node pertama menunjuk ke node terakhir melalui pointer prev
  - struktur ini membentuk "lingkaran ganda" sehingga tidak ada node yang memiliki nilai NULL

No.

Date

### • kelebihan:

- Mendukung traversal terus menerus ke kedua arah tanpa harus mulai ulang dari awal atau akhir
- Cocok untuk aplikasi seperti aplikasi video / musik yang harus bergerak mulus dari awal ke akhir atau sistem dengan jadwal sirkular

### • kekurangan:

- Implementasinya lebih kompleks dibandingkan dengan yang double linked list linear karena pemeliharaan referensi circular harus konsisten (contoh ketika insert / delete, harus memastikan link kembali mengarah pada sentinel / awal list)
- Penggunaan memori sama dengan double linked list non circular (dua pointer per node) dengan tambahan kompleksitas logika untuk circular linking.

### Ringkasan Perbandingan

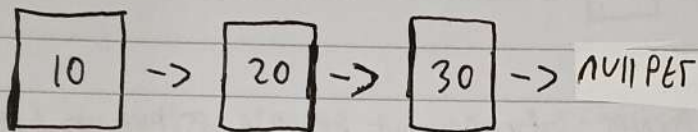
Tipe	arah traversal	akhir list	kemudahan inisialisasi
Single linked list non circular	satv arah (forward)	Null pada node terakhir	sederhana
single linked list circular	satv arah (forward)	node terakhir menunjuk ke head	Perlu logika tambahan
double linked list non circular	dua arah	Null pada head dan tail	lebih kompleks dari single
double linked list circular	dua arah	last $\leftrightarrow$ first (circular)	kompleks (referensi circular)

No.

Date

## ilustrasi sederhana

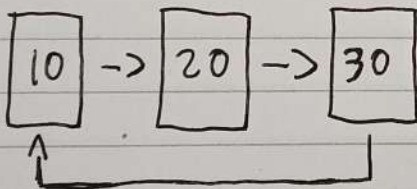
### 1. Single linked list non circular



Penjelasan :

- Setiap node hanya memiliki satu pointer (next) yang menunjuk ke node selanjutnya
- Node terakhir menunjuk ke null (tidak ada node berikutnya)

### 2. Single linked list circular



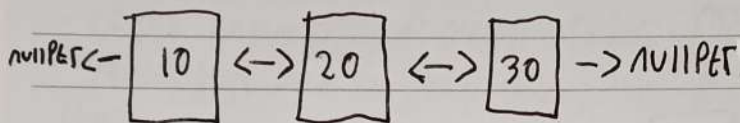
Penjelasan :

- Sama seperti single linked list pada umumnya, tetapi node terakhir (dengan nilai 30) menunjuk kembali ke node pertama (10), sehingga membentuk lingkaran
- traversal harus berhenti jika sudah kembali ke node awal agar tidak terjadi infinite loop

No.

Date

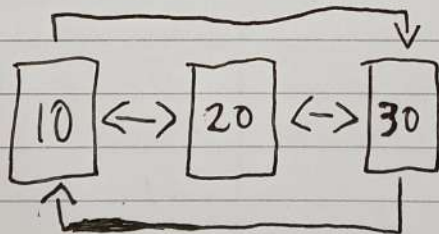
### 3. Double linked list non circular



Penjelasan:

- Setiap node memiliki dua pointer: satu menunjuk ke node berikutnya (next) dan satu menunjuk ke node sebelumnya (prev)
- Node pertama memiliki prev bernilai null dan node terakhir memiliki next bernilai null

### 4. Double linked list circular



Penjelasan:

- Setiap node memiliki dua pointer (next dan prev)
- Node terakhir menunjuk kembali ke node pertama melalui pointer next dan node pertama menunjuk ke node terakhir melalui pointer prev, membentuk double linked list circular tanpa "ujung" (NULL)