

Daf'i Huzan A.H

AI1.2024.15851 AI1.4208

Tugas resume 21PTO

=Sorting=
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] } belum urut
 - Random 2 : [4, 1, 3, 2, 5, 0]
 - Random ke-n : [0, 1, 2, 3, 4, 5] } terurut

2. bubble Sort

- konsep: bandingkan dan tukar elemen berdekatan hingga tidak ada pertukaran
- Contoh: array: [5, 1, 4, 2, 8]
 - Langkah 1: 5 > 1 -> tukar -> [1, 5, 4, 2, 8]
5 > 4 -> tukar -> [1, 4, 5, 2, 8]
5 > 2 -> tukar -> [1, 4, 2, 5, 8]
5 < 8 -> tetap
 - Langkah 2: 1 < 4 -> tetap
4 > 2 -> tukar -> [1, 2, 4, 5, 8] -> terurut

3. Selection Sort

- konsep: cari nilai terkecil, lalu tempatkan di posisi awal
- Contoh: array: [29, 10, 14, 37, 13]
 - Langkah 1: Cari yang terkecil (10) -> tukar dengan indeks 0 -> [10, 29, 14, 37, 13]
 - Langkah 2: Cari yang terkecil (13) di sisa array -> tukar dengan indeks 1 -> [10, 13, 14, 37, 29]
 - Langkah 3: Cari yang terkecil (14) -> Sudah di posisi tepat
 - Langkah 4: Cari yang terkecil (29) -> tukar dengan indeks 3 -> [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 di depan \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: $mid = (0+1)/2 = 0 \rightarrow$ 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 sekvensial, 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) | statis (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 menuju ke NULL, melainkan menuju kembali ke node pertama (head), membentuk siklus
- List ini tidak memiliki "akhir" secara eksplisit, sehingga traversal bisa dilakukan secara kontinu (selalu berputar)

- Kelebihan :

- Sangat cocok untuk aplikasi yang membutuhkan perulangan terus-menerus
- Memudahkan dalam menimplementasikan 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)

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 menambah node sebelumnya
 - traversal bolak-balik sehingga cocok untuk aplikasi seperti browser history dan navigasi web
- kekurangan:
 - membutuhkan lebih banyak memori karena setiap node menyimpan dua pointer
 - operasi pembalikan pointer harus dilakukan dengan hati-hati (contoh saat penghapusan, perlu mengupdate kedua pointer dari node tersebut)

* 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 sejak awal jika akhir
 - cocok untuk aplikasi seperti aplikasi video / musik yang harus bergerak mulus dari awal ke akhir atau sistem dengan jadwal siklus
- 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

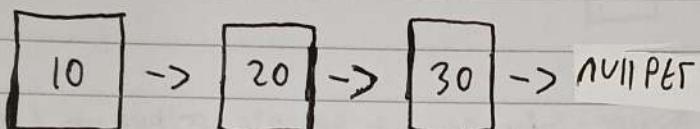
| Tipe | arah traversal | akhir list | kemudahan inisialisasi |
|------------------------------------|------------------------|--|----------------------------------|
| single linked list non circular | satu arah (forward) | null pada node terakhir | sederhana |
| single linked list circular | satu arah (forward) | node terakhir menunjuk ke head | perlu logika tambahan |
| double linked list non circular | dua arah dua cap | null pada head dan tail | lebih kompleks dari single |
| double linked list circular | dua arah dua cap | last \leftrightarrow first (circular) | kompleks (referensi Circular) |

No. _____

Date _____

Ilustrasi Sederhana

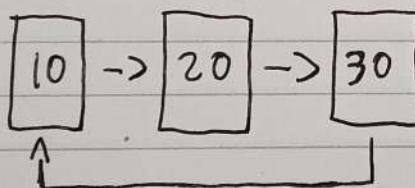
1. Single linked list non circular



Penjelasan:

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

2. Single linked list circular



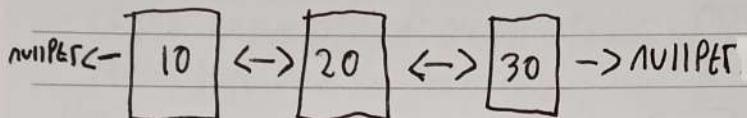
Penjelasan:

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

No. _____

Date _____

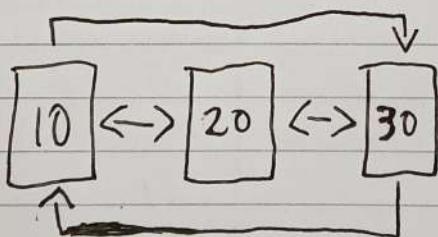
3. double linked list non circular



Penjelasan:

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

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)