LAPORAN PRAKTIKUM

ALGORITMA DAN STRUKTUR DATA

Cambria, 20 bold

IKA SULASTRI NINGSIH

240306030

github.com/IKASULASTRININGSIH

Program Studi Teknologi Informasi

Fakultas Dakwah dan Ilmu Komunikasi

Universitas Islam Negeri Mataram

2024

Pertemuan	10
Topik	Linked List
Repository	
Tanggal	2 Desember 2024

A. Tujuan

Tujuan praktikum Linked List

Tujuan utama dari Linked List adalah menyediakan cara yang fleksibel dan dinamis untuk menyimpan data dalam struktur terhubung, terutama dalam situasi di mana ukuran data atau jumlah elemen tidak diketahui sebelumnya atau dapat berubah selama waktu eksekusi. Berikut adalah beberapa tujuan spesifik penggunaan Linked List:

1. Manajemen Dinamis Memori
2. Linked List memungkinkan alokasi memori secara dinamis, sehingga dapat menangani perubahan ukuran data secara fleksibel tanpa perlu mengalokasikan atau mengosongkan blok besar memori secara manual.
3. Memori dialokasikan hanya saat diperlukan (sesuai kebutuhan).
4. . Penyisipan dan Penghapusan yang Efisien
5. Operasi penyisipan (insertion) dan penghapusan (deletion) elemen pada Linked List lebih cepat dibandingkan array, terutama pada awal atau tengah struktur data, karena tidak memerlukan penggeseran elemen lain.
6. 3. Mengatasi Keterbatasan Array
7. Tidak seperti array, yang memiliki ukuran tetap, Linked List tidak memerlukan ukuran awal yang ditentukan.
8. Cocok untuk situasi di mana jumlah elemen dapat berubah-ubah.
9. 4. Mendukung Struktur Data Kompleks
10. Linked List digunakan sebagai dasar untuk membangun struktur data lain seperti stack, queue, deque, dan bahkan graph atau tree.
11. 5. Efisiensi dalam Beberapa Kasus
12. Ketika elemen sering ditambahkan atau dihapus, Linked List lebih efisien dibandingkan array, karena array membutuhkan alokasi ulang memori jika ukurannya bertambah melebihi kapasitas awal.

13. 6. Menyimpan Data yang Tidak Berkesinambungan
14. Linked List dapat digunakan untuk menyimpan data yang tidak bersebelahan secara fisik di memori. Hal ini memanfaatkan potensi memori yang tersedia lebih baik dibandingkan array.
15. Jenis-jenis Linked List:
16. Singly Linked List: Setiap node terhubung ke node berikutnya.
17. Doubly Linked List: Setiap node memiliki dua pointer, ke node berikutnya dan sebelumnya.
18. Circular Linked List: Node terakhir terhubung kembali ke node pertama.
19. Mengenali simbol-simbol standar dalam flowchart

B. Requirement

1.	Sistem Operasi yang digunakan	:	Windows 10 vers
2.	Browser	:	Google Chrome Version 129.0.6668.100 (Official Build) (x86_64)
3.	Tools yang digunakan	:	Visual Studio Code

C. Dasar Teori

Dasar Teori Linked List (Dari Video [9])

Linked List adalah salah satu struktur data dasar yang sering digunakan dalam ilmu komputer dan pemrograman. Teori dasarnya mencakup konsep berikut:

1. Definisi Linked List

Linked List adalah struktur data linear yang terdiri dari node-node yang saling terhubung melalui pointer. Tidak seperti array, Linked List tidak memerlukan lokasi memori yang berurutan untuk menyimpan data.

Node dalam Linked List

Setiap node dalam Linked List memiliki dua bagian:

Data: Bagian yang menyimpan informasi atau nilai.

Pointer (Next): Bagian yang menunjuk ke node berikutnya dalam Linked List.

2. Jenis-jenis Linked List

Singly Linked List:

Setiap node hanya memiliki satu pointer yang menunjuk ke node berikutnya.
Traversal hanya dapat dilakukan maju dari node pertama (head) ke node terakhir (tail).

Doubly Linked List:

Setiap node memiliki dua pointer: satu menunjuk ke node sebelumnya, dan satu lagi menunjuk ke node berikutnya.
Traversal dapat dilakukan maju atau mundur.

Circular Linked List:

Node terakhir menunjuk kembali ke node pertama, membentuk lingkaran.
Bisa berupa singly atau doubly circular Linked List.

3. Keunggulan Linked List

Ukuran Dinamis: Linked List tidak memiliki batas ukuran tetap karena memori dialokasikan secara dinamis.

Kemudahan Penambahan dan Penghapusan:
Elemen baru dapat ditambahkan di mana saja tanpa perlu menggeser elemen lain, seperti pada array.

Efisiensi Memori (Untuk Data Tertentu): Tidak perlu memesan ruang untuk data yang tidak digunakan.

4. Kekurangan Linked List

Akses Lambat:
Tidak mendukung akses langsung ke elemen tertentu (harus melalui traversal).

Overhead Memori:
Membutuhkan ruang tambahan untuk menyimpan pointer di setiap node.

Implementasi Kompleks:
Operasi seperti traversal atau penghapusan memerlukan lebih banyak kode dibanding array.

5. Operasi Dasar pada Linked List

Insertion:

Menambahkan elemen baru di awal, tengah, atau akhir Linked List.

Deletion:

Menghapus elemen dari awal, tengah, atau akhir Linked List.

Traversal:

Menelusuri semua elemen untuk membaca atau mencari data.

Search:

Mencari elemen dengan nilai tertentu.

6. Perbandingan dengan Array

Array:

Lokasi memori berurutan.

Ukuran tetap.

Akses cepat (akses langsung dengan indeks).

Linked List:

Lokasi memori tidak harus berurutan.

Ukuran dinamis.

Akses lambat (melalui traversal).

6. Implementasi Linked List

Biasanya ditulis menggunakan bahasa pemrograman seperti Python atau C++.

Membuat kelas Node untuk mendefinisikan struktur data.

Membuat kelas LinkedList untuk mengelola node dan operasinya.

7. Contoh Kode Python untuk Singly Linked List

```
class Node:
   def _init_(self, data):
      self.data = data
      self.next = None

class LinkedList:
   def _init_(self):
      self.head = None
```

```
    def insert_at_end(self, data):
        new_node = Node(data)
        if self.head is None:
            self.head = new_node
            return
        temp = self.head
        while temp.next:
            temp = temp.next
        temp.next = new_node

    def print_list(self):
        temp = self.head
        while temp:
            print(temp.data, end=" -> ")
            temp = temp.next
        print("None")

# Contoh penggunaan
ll = LinkedList()
ll.insert_at_end(10)
ll.insert_at_end(20)
ll.insert_at_end(30)
ll.print_list()
```

8. Aplikasi Linked List

Mengimplementasikan struktur data seperti stack, queue, dan graph.

Dig

Berikut adalah poin-poin utama

1. Pengantar Linked List

Linked List adalah struktur data dinamis yang terdiri dari node.

Setiap node memiliki dua elemen:

Data: Menyimpan informasi.

Pointer: Menunjuk ke node berikutnya dalam list.

2. Perbedaan Linked List dan Array

Array: Data disimpan di lokasi memori yang bersebelahan, sehingga sulit menambah/menghapus elemen secara dinamis.

Linked List: Data dapat disimpan di lokasi memori acak karena setiap node memiliki pointer ke node berikutnya, membuat penambahan/penghapusan lebih mudah.

3. Operasi Dasar

Traversal: Menelusuri node dari awal hingga akhir.

Insertion:

Menambahkan node baru di awal (head).

Menambahkan di tengah (setelah node tertentu).

Menambahkan di akhir (tail).

Deletion:

Menghapus node pertama, terakhir, atau node tertentu.

Search: Mencari elemen berdasarkan nilai.

4. Implementasi dalam Python

Membuat kelas Node untuk mendefinisikan node (data + pointer).

Membuat kelas LinkedList untuk mengatur head dan operasi-operasi.

Implementasi fungsi seperti insert_at_beginning, insert_at_end, dan print_list.

5. Contoh Kode Python Singkat

Menunjukkan langkah-langkah membuat linked list dan menambahkan node:

```
class Node:
   def _init_(self, data):
      self.data = data
      self.next = None
```

```
class LinkedList:
    def _init_(self):
        self.head = None

    def insert_at_end(self, data):
        new_node = Node(data)
        if not self.head:
            self.head = new_node
        else:
            temp = self.head
            while temp.next:
                temp = temp.next
            temp.next = new_node

    def print_list(self):
        temp = self.head
        while temp:
            print(temp.data, end=" -> ")
            temp = temp.next
        print("None")
```

6. Keuntungan dan Kekurangan Linked List

Keuntungan:

Mudah menambahkan/menghapus elemen.

Tidak memerlukan alokasi memori kontigu.

Kekurangan:

Akses elemen lambat karena traversal dilakukan secara berurutan.

Menggunakan lebih banyak memori karena pointer.

7. Aplikasi Linked List

Mengimplementasikan struktur data seperti stack, queue, dan hash table.

Digunakan dalam aplikasi seperti undo operation dan representasi graf.

Untuk detail lebih lengkap, Anda dapat menonton video ini di Linked List - Data Structures & Algorithms Tutorials in Pyth

Linked list

Tipe Data Linked List (Dari Video [9])

Linked List adalah struktur data yang terdiri dari serangkaian elemen yang saling terhubung secara berurutan. Setiap elemen dalam Linked List disebut node. Masing-masing node terdiri dari dua bagian:

Data: Bagian ini menyimpan nilai atau informasi.

Pointer (Next): Bagian ini menyimpan referensi ke node berikutnya dalam Linked List.

Terdapat beberapa tipe Linked List:

Singly Linked List: Setiap node hanya memiliki satu pointer yang menunjuk ke node berikutnya.

Doubly Linked List: Setiap node memiliki dua pointer, satu menunjuk ke node sebelumnya dan satu lagi ke node berikutnya.

Circular Linked List: Node terakhir dalam Linked List menunjuk kembali ke node pertama, membentuk lingkaran.

Perbandingan Linked List dengan Array (Dari Video [9])

Array

Penyimpanan Data: Array menyimpan elemen secara berurutan di lokasi memori yang bersebelahan.

Akses Elemen: Elemen dapat diakses langsung menggunakan indeks (akses lebih cepat).

Ukuran Tetap: Ukuran array biasanya ditentukan saat deklarasi dan tidak dapat diubah.

Efisiensi Penambahan/Penghapusan:

Menambahkan elemen di tengah array atau menghapus elemen memerlukan pergeseran elemen lain, sehingga lebih lambat.

Memori: Memori digunakan secara efisien tanpa pointer tambahan.

Linked List

Penyimpanan Data: Elemen tidak perlu disimpan di lokasi memori yang bersebelahan. Setiap elemen menunjuk ke lokasi berikutnya melalui pointer.

Akses Elemen: Elemen harus diakses secara berurutan dari node pertama (lebih lambat dibanding array).

Ukuran Dinamis: Linked List dapat tumbuh atau menyusut sesuai kebutuhan karena memori dialokasikan secara dinamis.

Efisiensi Penambahan/Penghapusan:

Penambahan dan penghapusan elemen lebih mudah dan cepat, terutama di awal atau akhir Linked List.

Memori: Membutuhkan memori tambahan untuk menyimpan pointer.

Kelebihan Linked List Dibandingkan Array

Penambahan dan penghapusan elemen lebih efisien, terutama di awal atau tengah struktur.

Ukuran dapat diubah secara dinamis sesuai kebutuhan.

Kekurangan Linked List Dibandingkan Array

Akses elemen lebih lambat karena harus dilakukan secara berurutan.

Membutuhkan lebih banyak memori karena ada pointer di setiap node.

D. Implementasi

```python
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None
        self.prev = None

class DoublyLinkedList:
    def __init__(self):
        self.head = None
        self.tail = None

    def add_front(self, data):
        new_node = Node(data)
        if self.head is None:
            self.head = new_node
            self.tail = new_node
        else:
            new_node.next = self.head
            self.head.prev = new_node
            self.head = new_node

    def add_back(self, data):
        new_node = Node(data)
```

E. Daftar Pustaka

1. Codebasics. Linked List - Data Structures & Algorithms Tutorials in Python. Diakses dari YouTube.

2. Narasimha Karumanchi, Data Structures and Algorithms Made Easy. CareerMonk Publications, 2022.

3. Thomas H. Cormen et al., Introduction to Algorithms. MIT Press, 3rd Edition, 2009.

4. GeeksforGeeks. Data Structures: Linked List. Diakses dari GeeksforGeeks.

Wikipedia Contributors. Linked List. Diakses dari Wikipedia.