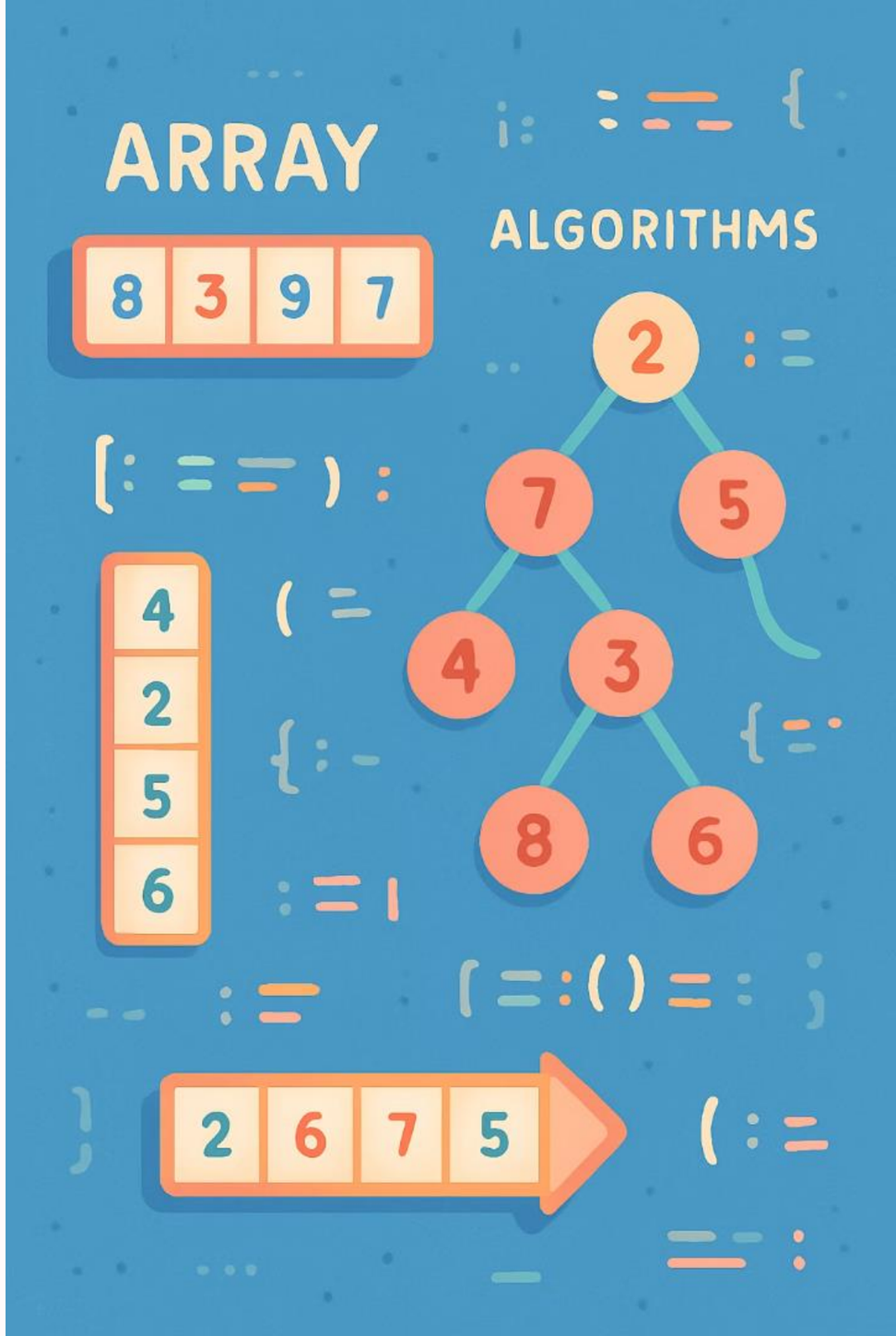
Algoritma Array: Konsep, Operasi, dan Implementasi

Silaturahmi Widaputri, S.T.P., M.T.P.



Tujuan Pembelajaran

Memahami Konsep Dasar

Mempelajari struktur dan karakteristik array sebagai struktur data.

Menguasai Operasi Dasar

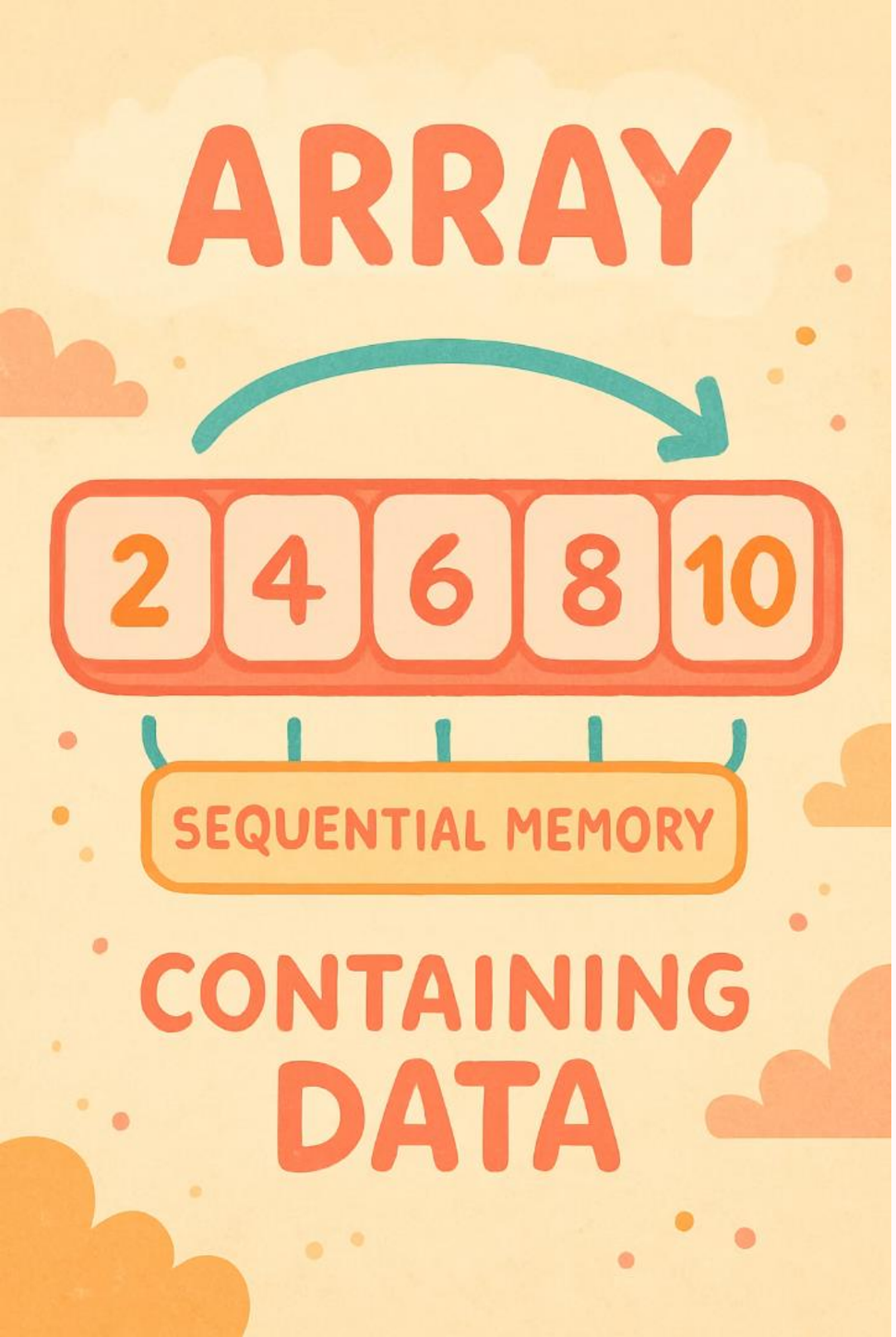
Mendalami cara mengakses, memanipulasi, dan mengelola data dalam array.

Mempelajari Algoritma Umum

Mengenal algoritma pencarian, pengurutan, dan manipulasi array.

Aplikasi Praktis

Menerapkan array untuk menyelesaikan masalah pemrograman nyata.



Pendahuluan: Apa itu Array?

Struktur Data Linier

Array menyimpan elemen-elemen dengan tipe data yang sama dalam lokasi memori berurutan.

Ukuran Tetap atau Dinamis

Array statis memiliki ukuran tetap. Array dinamis dapat diperluas.

Indeks Berbasis Nol

Pada kebanyakan bahasa pemrograman, indeks array dimulai dari 0.

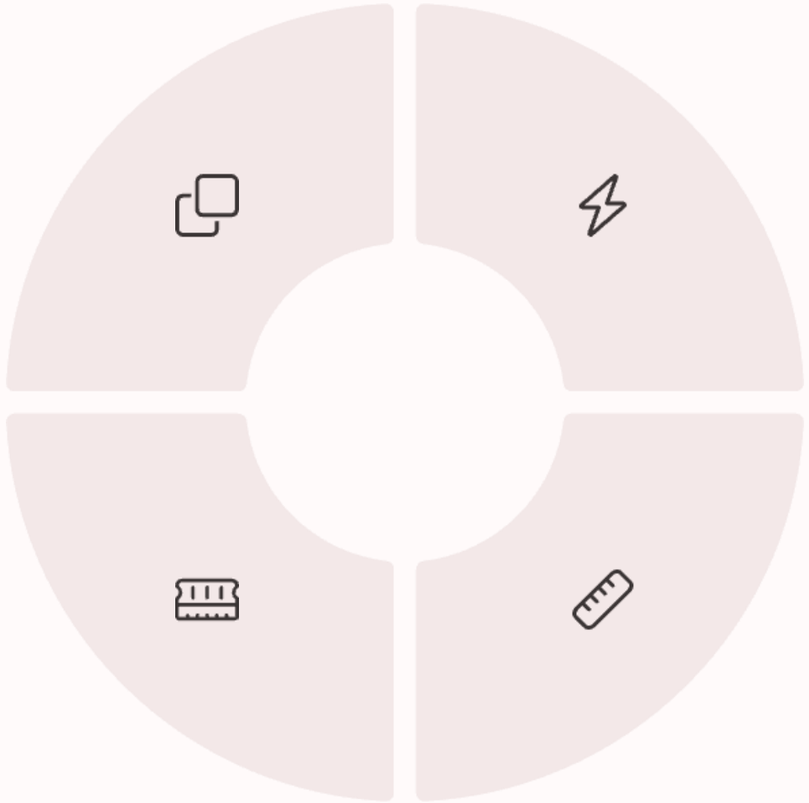
Penyimpanan Berurutan

Semua elemen disimpan secara berurutan dalam memori komputer.

Karakteristik Array

Homogen

Semua elemen dalam array memiliki tipe data yang sama.



Akses Langsung

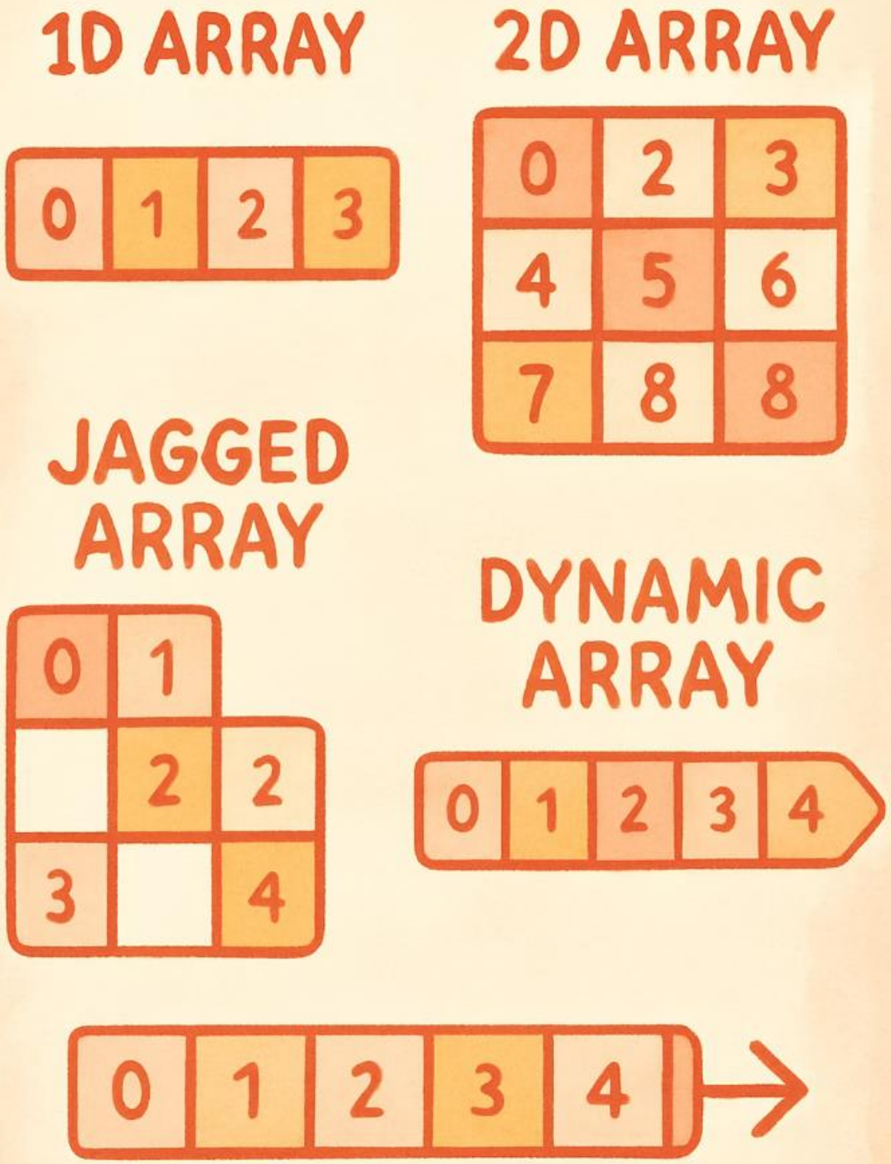
Mengakses elemen menggunakan indeks dengan kompleksitas O(1).

Efisiensi Memori

Penggunaan memori yang teratur dan efisien.

Ukuran Tetap

Pada array statis, ukuran dideklarasikan saat pembuatan.



Jenis-jenis Array

Array Satu Dimensi

Juga dikenal sebagai vektor. Menyimpan elemen dalam bentuk deret linier.

Array Multi-dimensi

Menyimpan elemen dalam bentuk tabel atau matriks dengan baris dan kolom.

Array Jagged

Array dari array dengan panjang yang berbeda-beda.

Array Dinamis

Ukuran dapat berubah saat runtime, seperti Vector atau ArrayList.

Representasi Array dalam Memori

Alamat Memori Berurutan

Elemen array disimpan pada alamat memori yang berurutan.

Formula Alamat

baseAddress + (index × sizeOfDataType)

Penyimpanan Berbeda Dimensi

Array 1D disimpan linear. Array 2D dengan pola tertentu.

Row-Major vs Column-Major

Dua cara menyimpan array multi-dimensi dalam memori.

Operasi Dasar: Inisialisasi Array

Deklarasi Array

Menentukan tipe data dan nama untuk array.

Contoh: **int[] numbers;** (Java), **int numbers[];** (C++)

Alokasi Memori

Menentukan ukuran array dan mengalokasikan memori.

Contoh: **numbers = new int[5];** (Java)

Inisialisasi Nilai

Memberikan nilai awal pada elemen array.

Contoh: **numbers = [1, 2, 3, 4, 5]** (Python)

Operasi Dasar: Akses Element

Akses Dengan Indeks

Elemen array diakses melalui indeks yang dimulai dari 0.

Contoh: `nilai = angka[3];` // mengakses elemen ke-4

Kompleksitas Waktu

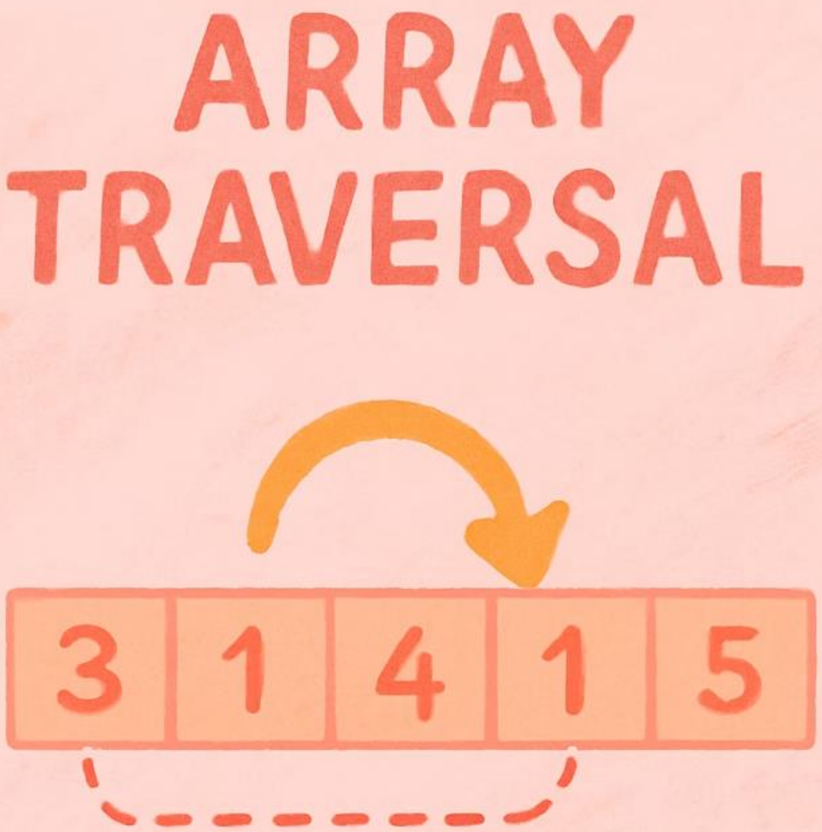
Akses elemen array memiliki kompleksitas O(1).

Waktu akses tidak tergantung ukuran array.

Penanganan Error

Indeks tidak valid dapat menyebabkan `IndexOutOfBoundsException`.

Validasi indeks perlu dilakukan untuk keamanan.



Operasi Dasar: Traversal

Iterasi Seluruh Elemen

Mengunjungi setiap elemen array secara berurutan untuk operasi tertentu.

Metode Traversal

- Loop For tradisional dengan indeks
- Loop While dengan kondisi
- For-each (enhanced for loop)

Kompleksitas

Traversal memiliki kompleksitas waktu O(n), dengan n adalah jumlah elemen.

Operasi Dasar: Pencarian (Linear Search)

Periksa Elemen Satu Per Satu

Memeriksa setiap elemen array secara berurutan.

Bandingkan Dengan Target

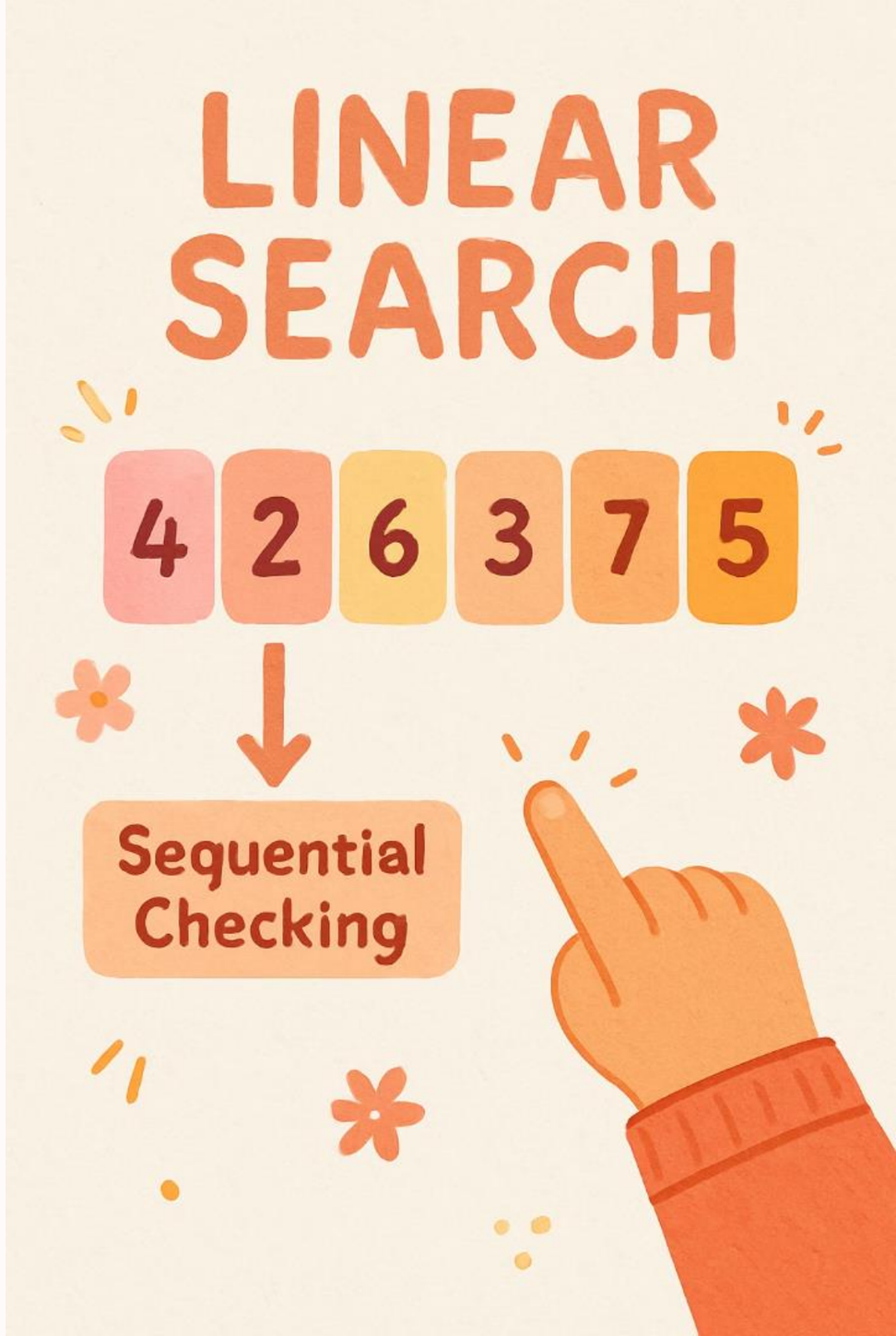
Membandingkan setiap elemen dengan nilai yang dicari.

Temukan Atau Lewati

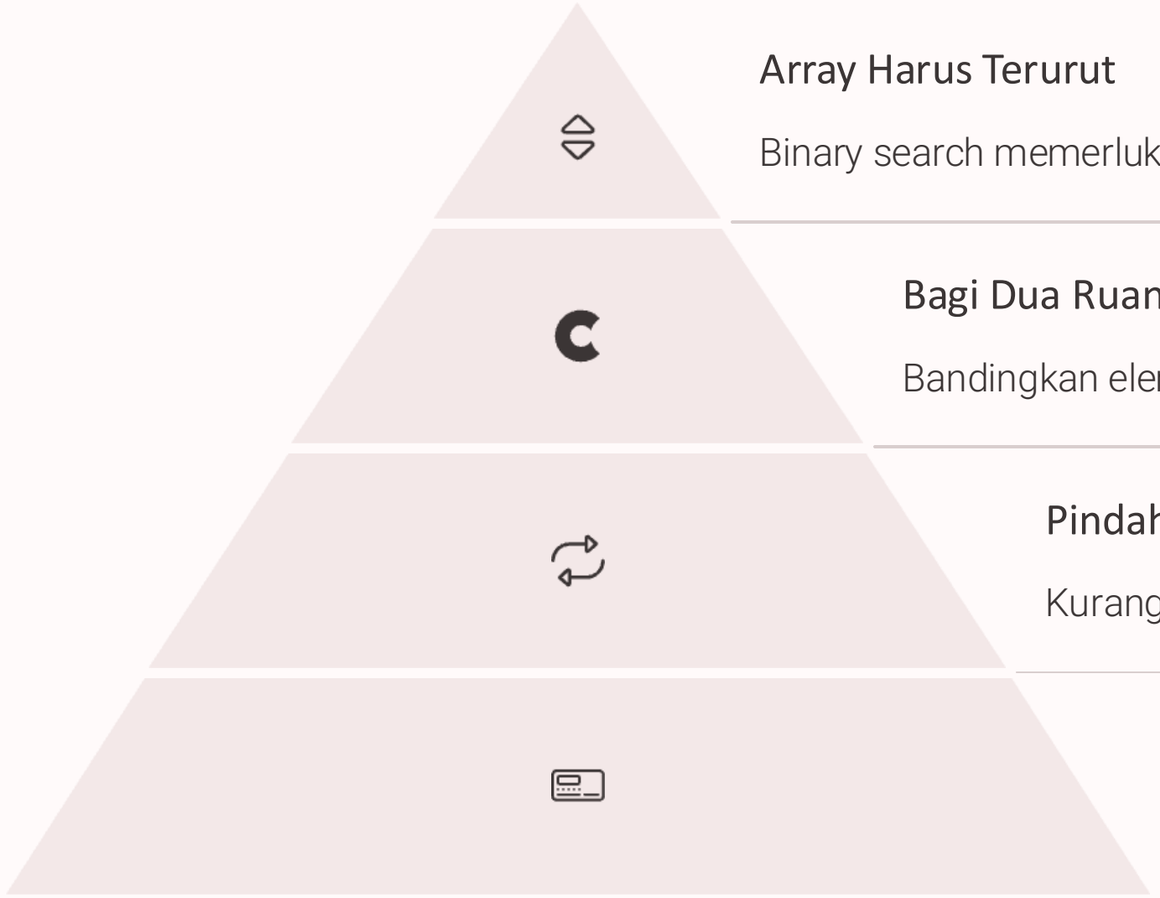
Jika cocok, pencarian selesai. Jika tidak, lanjut ke elemen berikutnya.

Kompleksitas O(n)

Dalam kasus terburuk, harus memeriksa seluruh array.



Operasi Dasar: Pencarian (Binary Search)



Array Harus Terurut

Binary search memerlukan array yang sudah diurutkan.

Bagi Dua Ruang Pencarian

Bandingkan elemen tengah dengan target.

Pindahkan Pencarian

Kurangi ruang pencarian ke setengah setiap iterasi.

Kompleksitas O(log n)

Jauh lebih efisien daripada linear search untuk array besar.

Operasi Dasar: Penyisipan (Insertion)

Temukan Posisi

Tentukan posisi dimana elemen baru akan disisipkan.

Geser Elemen

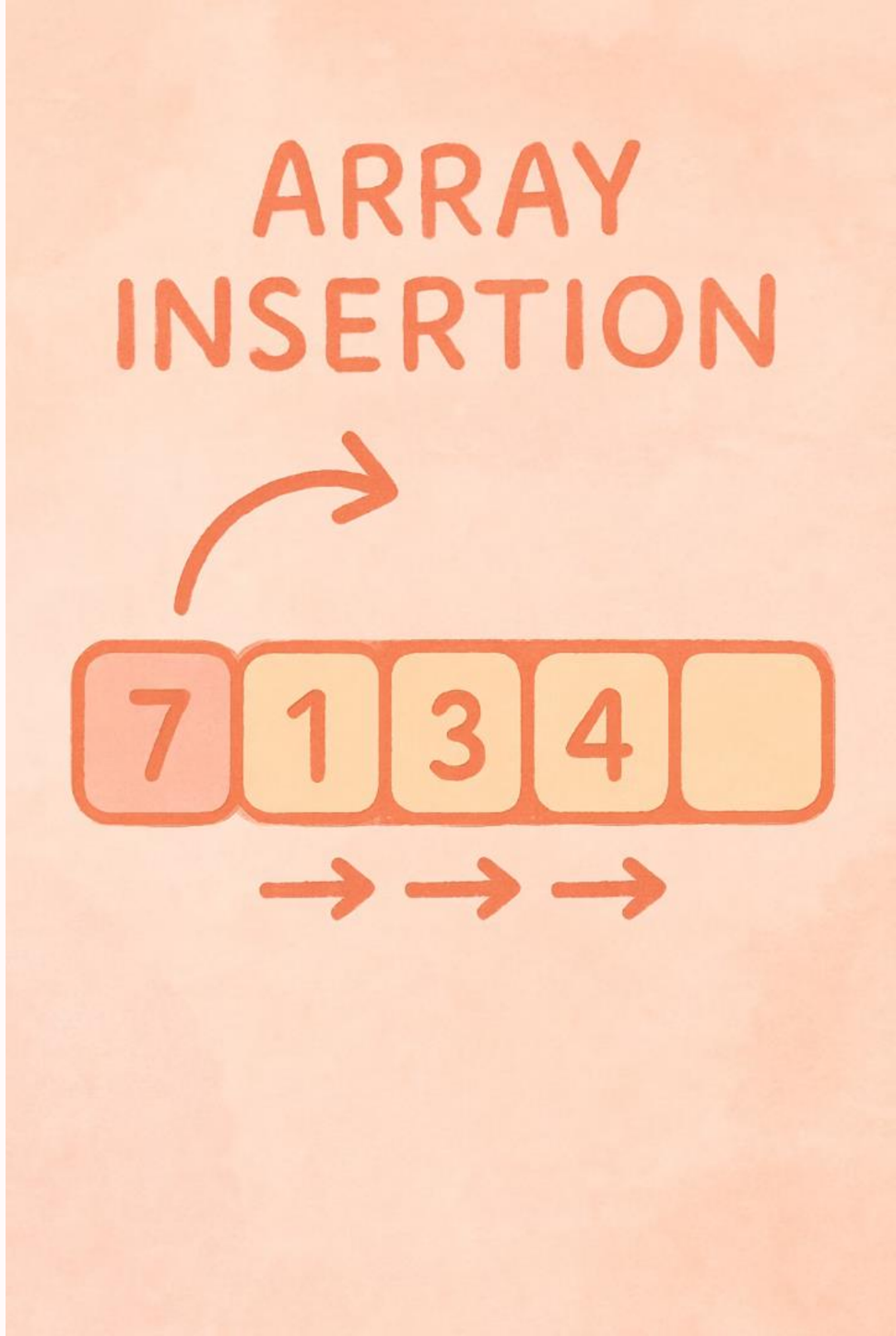
Pindahkan semua elemen setelah posisi tersebut satu langkah ke kanan.

Sisipkan Elemen

Tempatkan elemen baru pada posisi yang sudah dikosongkan.

Analisis Kompleksitas

Membutuhkan waktu O(n) pada kasus terburuk untuk pergeseran elemen.



Operasi Dasar: Penghapusan (Deletion)

Temukan Elemen

Identifikasi posisi elemen yang akan dihapus.

Hapus Elemen

Kosongkan posisi elemen dengan menimpa nilai.

Tutup Celah

Geser semua elemen setelahnya satu langkah ke kiri.

Kurangi Ukuran

Pada array dinamis, ukuran array dikurangi satu.

Algoritma Sorting: Bubble Sort

Bandingkan Elemen Berdekatan Berdekatan

Periksa setiap pasangan elemen yang berdekatan.

Tukar Jika Tidak Terurut

Tukar posisi jika elemen tidak dalam urutan yang benar.

Verifikasi Hasil

Array terurut dengan elemen terbesar di posisi akhir.

Ulangi Proses

Lakukan n-1 kali untuk n elemen array.

Algoritma Sorting: Selection Sort

$O(n^2)$

Kompleksitas Waktu

Selection sort memiliki kompleksitas waktu kuadratik.

n-1

Jumlah Perbandingan

Membutuhkan n-1 putaran untuk mengurutkan n elemen.

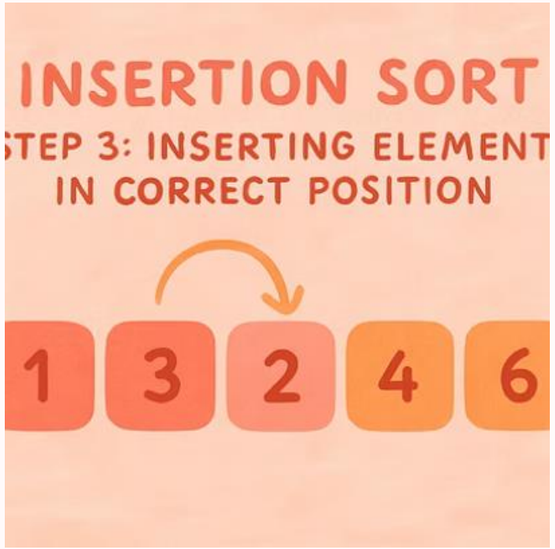
O(1)

Kebutuhan Memori

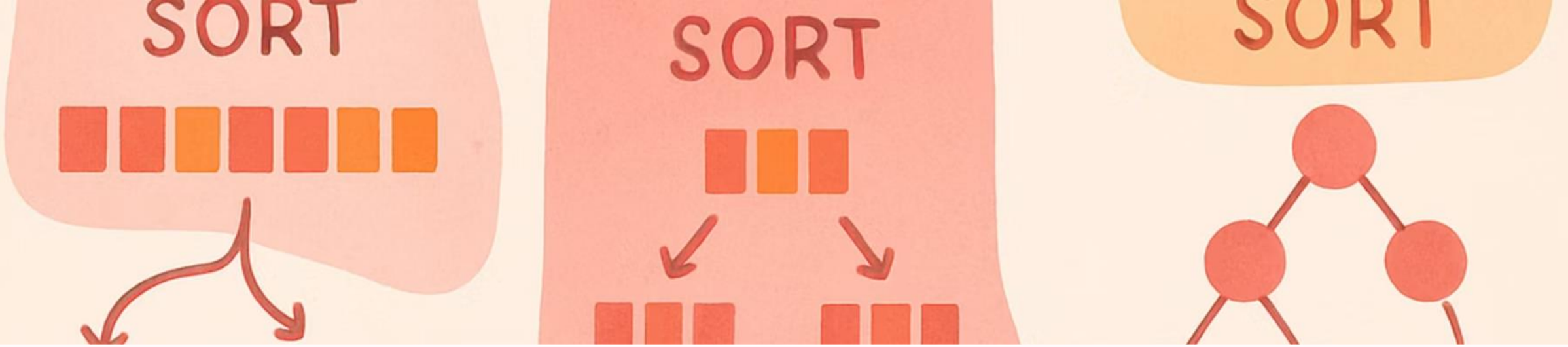
Hanya membutuhkan satu variabel tambahan untuk pertukaran.

Selection sort bekerja dengan mencari elemen minimum dari bagian array yang belum terurut, kemudian menempatkannya di posisi awal dari bagian yang belum terurut tersebut.

Algoritma Sorting: Insertion Sort



Insertion sort bekerja seperti mengurutkan kartu di tangan. Elemen diambil satu per satu dan disisipkan ke posisi yang tepat di bagian array yang sudah terurut.



Algoritma Sorting Lanjutan

Algoritma	Kompleksitas Rata-rata	Kompleksitas Terburuk	Ruang Memori
Quick Sort	O(n log n)	O(n^2)	O(log n)
Merge Sort	O(n log n)	O(n log n)	O(n)
Heap Sort	O(n log n)	O(n log n)	O(1)

Algoritma Array: Prefix Sum

Konsep Dasar

Prefix sum adalah array baru dimana setiap elemen merupakan jumlah dari semua elemen sebelumnya di array asli.

Prefix[i] = A[0] + A[1] + ... + A[i]

Implementasi

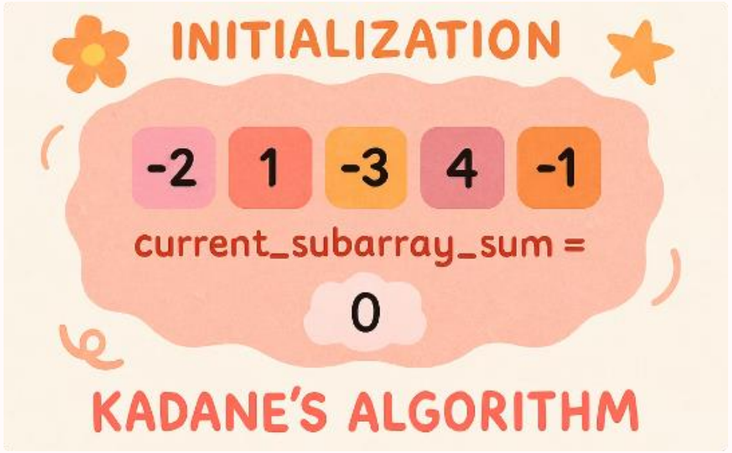
```
for(i = 1; i < n; i++)
  prefix[i] = prefix[i-1] + arr[i];
```

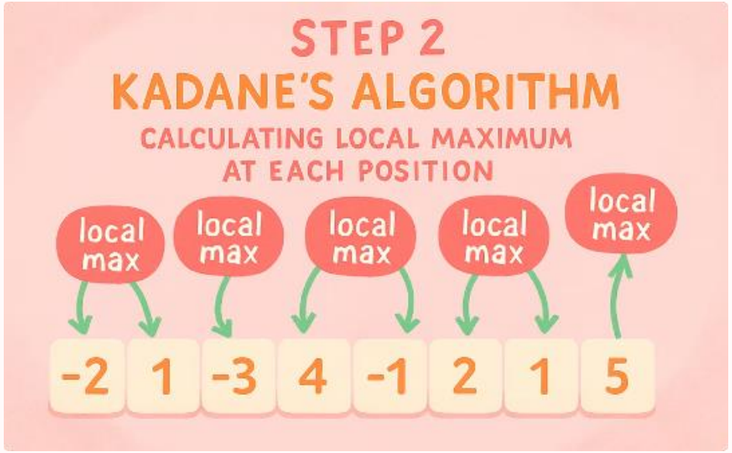
Kompleksitas waktu: O(n)

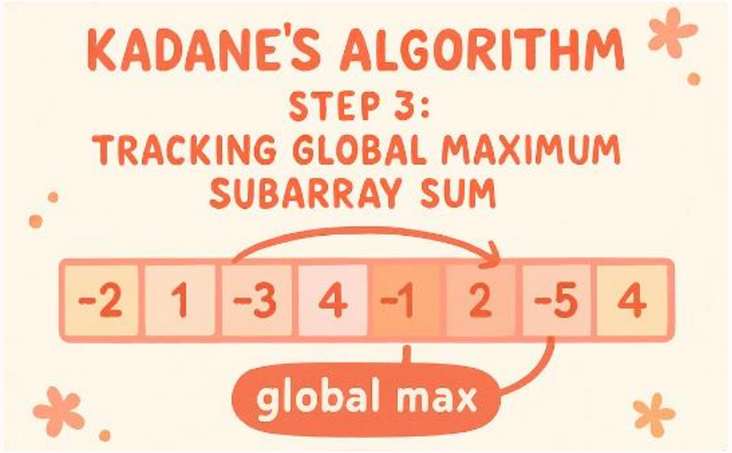
Aplikasi

- Menghitung jumlah rentang (range sum) dengan O(1)
- Mendeteksi subarray dengan jumlah tertentu
- Optimasi query rentang berulang

Algoritma Array: Kadane's Algorithm







Inisialisasi

Mulai dengan nilai awal current_max dan global_max sama dengan elemen pertama array.

Kalkulasi Lokal

Pada setiap posisi, pilih nilai maksimum antara elemen saat ini dan jumlah dengan elemen sebelumnya.

Update Maksimum Global

Perbarui nilai maksimum global jika nilai lokal saat ini lebih besar.

Algoritma Array: Two Pointers

Konsep Dasar

Menggunakan dua pointer yang bergerak dalam array untuk mencari solusi efisien.

Aplikasi Umum

Mencari pasangan elemen dengan jumlah tertentu dalam array terurut.

Kompleksitas

Mengurangi kompleksitas dari $O(n^2)$ menjadi $O(n)$ dengan menghindari nested loops.

Implementasi

Pointer bisa bergerak dari dua ujung atau dari posisi yang sama.

Algoritma Array: Sliding Window

Inisialisasi Jendela

Tentukan ukuran jendela (tetap atau variabel) dan mulai dari elemen pertama.

Proses Elemen dalam Jendela

Lakukan perhitungan yang diperlukan pada elemen-elemen dalam jendela.

Geser Jendela

Pindahkan jendela ke depan dengan menghapus elemen paling kiri dan menambahkan yang baru.

Perbarui Hasil

Simpan hasil terbaik (maksimum/minimum) selama pergeseran jendela.

Struktur Data Berbasis Array

Dynamic Array

Array yang dapat menyesuaikan ukuran secara otomatis. Contoh: ArrayList (Java), Vector (C++).

Stack

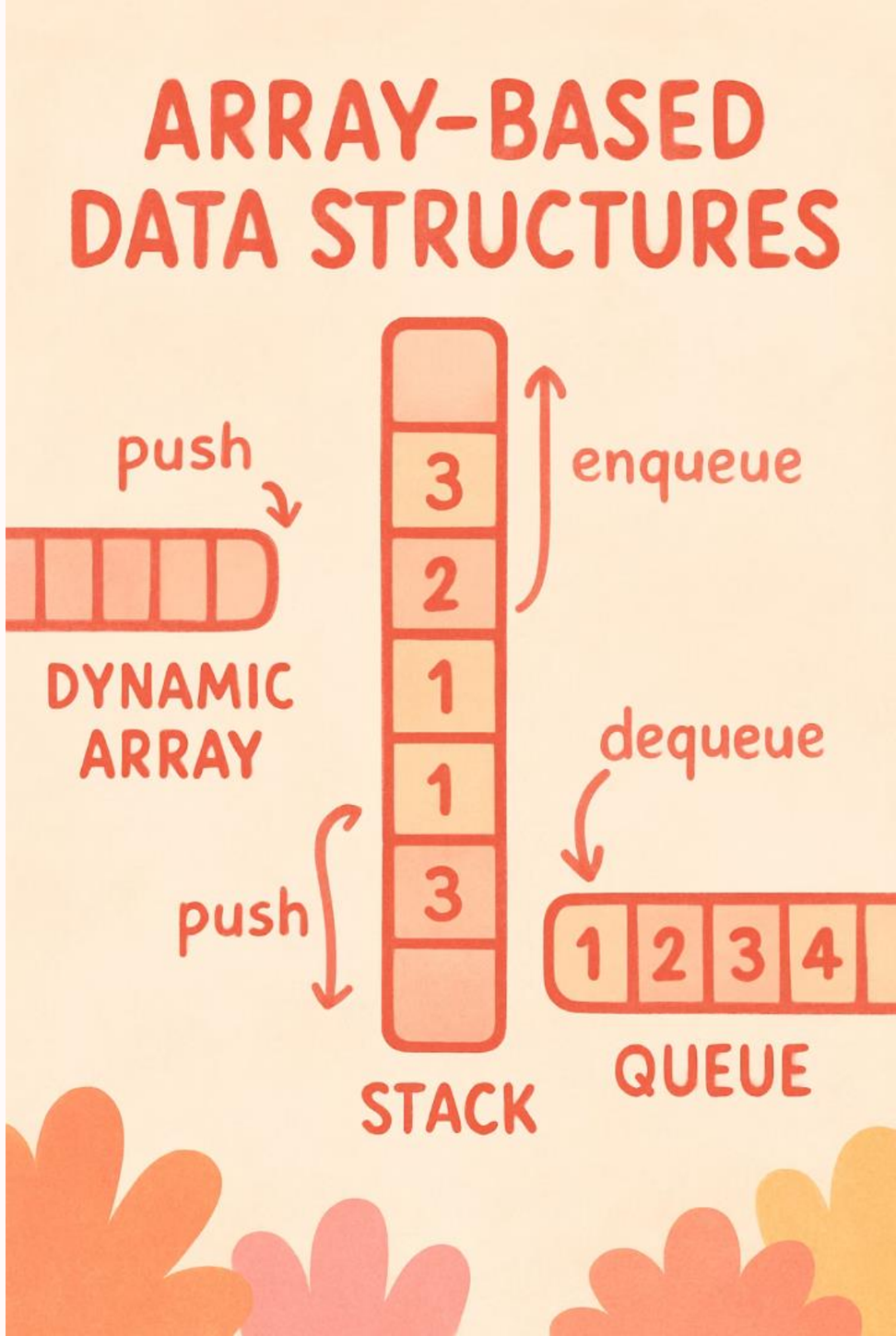
Struktur LIFO (Last In First Out) yang dapat diimplementasikan dengan array.

Queue

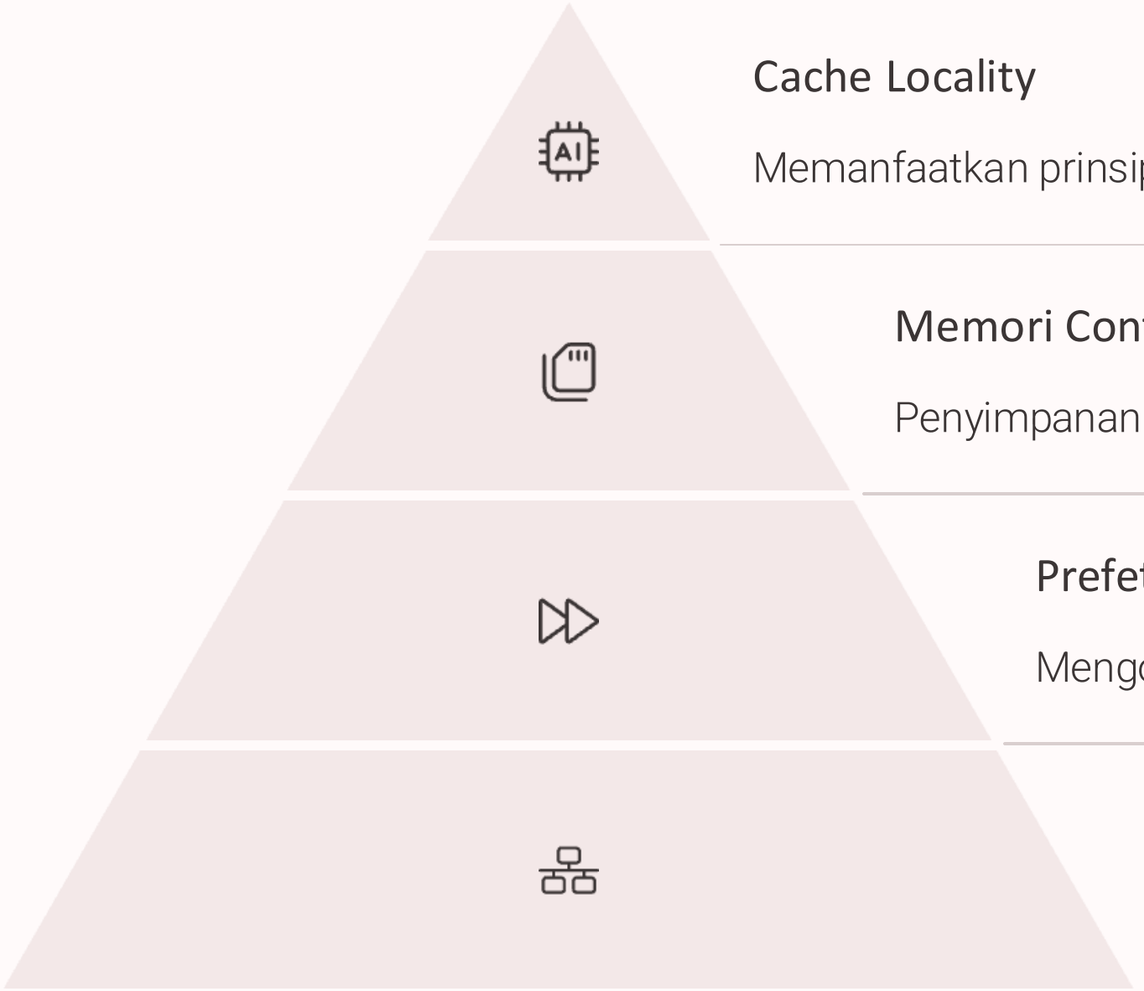
Struktur FIFO (First In First Out) dengan operasi enqueue dan dequeue.

Matrix

Representasi array 2D untuk operasi matematika dan grafis.



Optimasi Operasi Array



Cache Locality

Memanfaatkan prinsip lokalitas untuk meningkatkan performa.

Memori Contiguous

Penyimpanan berurutan meningkatkan efisiensi akses data.

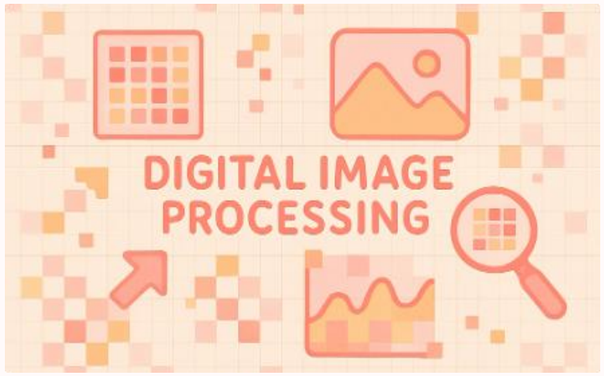
Prefetching

Mengoptimasi pengambilan data dengan membaca di muka.

Paralelisasi

Memproses beberapa elemen array secara bersamaan.

Aplikasi Array dalam Kehidupan Nyata

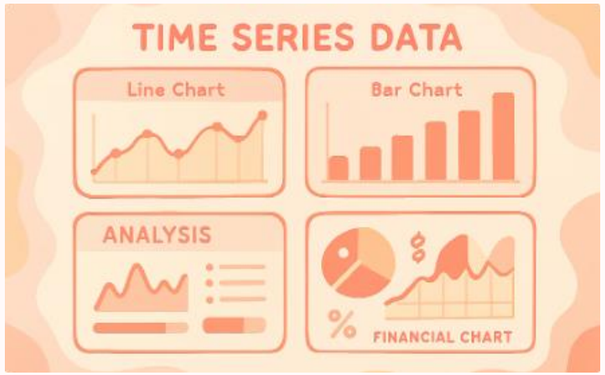


Pengolahan Gambar

Gambar digital direpresentasikan sebagai array 2D dari nilai piksel.

Spreadsheet

Tabel data dalam aplikasi seperti Excel diimplementasikan dengan array.



Analisis Data

Data deret waktu dan pengukuran sensor disimpan dan diproses sebagai array.

Algoritma Array dalam Competitive Competitive Programming

Problem Solving Techniques

- Prefix Sum untuk query rentang
- Kadane untuk maximum subarray
- Two Pointers untuk optimasi pencarian
- Sliding Window untuk rentang tetap

Optimasi Waktu dan Memori

- Menghindari alokasi berulang
- Menggunakan in-place algorithms
- Memanfaatkan struktur data khusus

Tips Penyelesaian

- Analisis kompleksitas sebelum implementasi
- Pertimbangkan kasus batas (edge cases)
- Uji dengan berbagai data input

Batasan dan Tantangan Array

Ukuran Tetap

Array statis memiliki ukuran yang tetap setelah deklarasi. Perubahan ukuran memerlukan realokasi.

Overhead Memori

Array dinamis menggunakan ruang lebih banyak untuk mengakomodasi pertumbuhan data.

Kompleksitas Operasi

Operasi insert/delete membutuhkan waktu O(n) karena pergeseran elemen.

Struktur Data Alternatif

Linked List, Hash Table, dan Tree bisa menjadi pilihan lebih baik untuk kasus tertentu.

Tren dan Perkembangan Terkini

1

Parallel Array Processing

Teknologi multi-core dan distributed computing untuk pemrosesan array skala besar.

GPU Acceleration

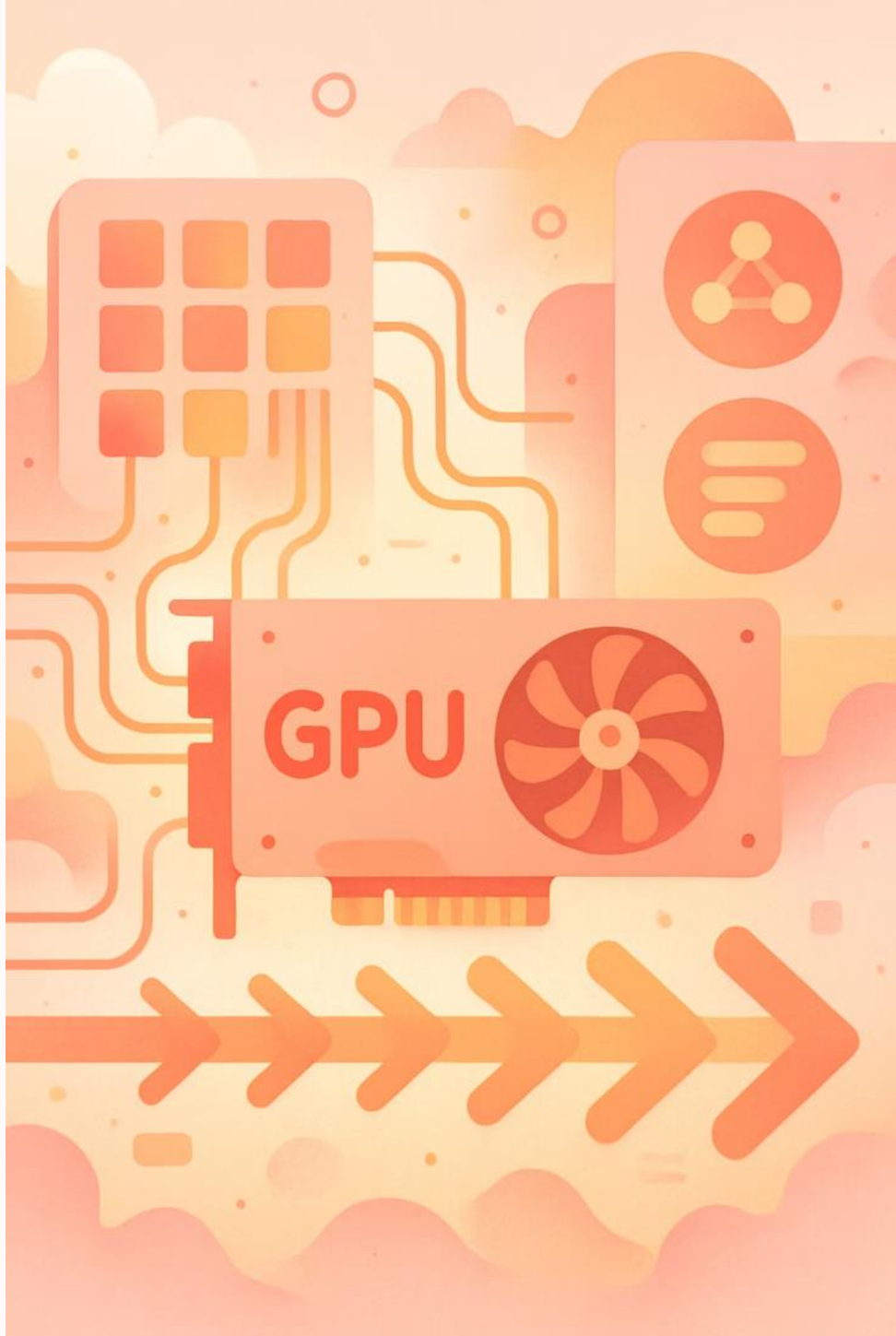
Memanfaatkan kekuatan GPU untuk operasi array paralel seperti pada machine learning.

Library Modern

NumPy, TensorFlow, dan PyTorch memanfaatkan optimasi tingkat rendah untuk operasi array.

Penelitian Terkini

Algoritma baru untuk processing big data, compression, dan sparse arrays.



Kesimpulan dan Ringkasan

Konsep Dasar

Array adalah struktur data linier dengan akses elemen O(1).

Algoritma Penting

Searching, sorting, dan manipulasi array adalah fondasi pemrograman.

Kelebihan dan Batasan

Efisien untuk akses acak tetapi kurang fleksibel untuk manipulasi data dinamis.

Aplikasi Praktis

Array mendasari banyak aplikasi komputasi modern dan struktur data kompleks.

Daftar Pustaka

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to Algorithms* (4th ed.). MIT Press.
- Sedgewick, R., & Wayne, K. (2021). *Algorithms* (4th ed.). Addison-Wesley.
- Skiena, S. S. (2020). *The Algorithm Design Manual* (3rd ed.). Springer.
- McDowell, G. L. (2020). *Cracking the Coding Interview* (6th ed.). CareerCup.
- Bhargava, A. (2016). *Grokking Algorithms*. Manning Publications.

Referensi tambahan tersedia online di repositori GitHub universitas terkemuka dan platform pembelajaran seperti Coursera dan edX.