**ARTIKEL INFORMATIF**

JENIS-JENIS ALGORITMA DALAM PEMPROGRAMAN

BESERTA DENGANN CONTOHNYA



SERLIA TURU’ ALLO\_ D0424312

FAKULTAS TEKNIK SISTEM INFORMASI

UNIVERSITAS SULAWESI BARAT

Dalam era digital saat ini, algoritma memainkan peran penting dalam berbagi aspek kehidupan sehari-hari, mulai dari pencarian informasi di internet hingga dapat menyelesaikan permasalahan dengan mudah. Oleh karena itu, algoritma adalah sekumpulan Langkah-langkah yang terstruktur dan sistematis yang dirancang untuk menyelesaikan suatu masalah untuk mencapai tujuan tertentu. Algoritma biasanya digunakan dalam Bahasa Pemprograman Komputer untuk menjalankan tugas secara efisien untuk memberikan keluaran dan masukan yang diberikan. Algoritma dan pemrograman adalah tempat di mana kreativitas bertemu dengan logika dan imajinasi dipadukan dengan struktur. Memahami jenis-jenis algoritma sangatlah penting, karna setiap jenis algoritma memiliki karakteristik dan aplikasi yang berbeda. Artikel ini akan membahas jenis-jenis algoritma serta contohnya.

Ada beberapa pengeritian Algoritma menurut parah ahli yaitu:

1. Menurut Donal Knuth dalam buku “*The Art of Computer Programming”* Algoritma adalah langkah-langkah dalam menyelesaikan suatu masalah. Jika seseorang menginginkan algoritma yang baik dalan difinisi sedehana dan estetika. Ada kriteria yang harus dimiliki yaitu waktu yang dibutuhkan untuk berjalannya algoritma. Kriteria yang lain adalah adaptasi dari algoritma ke komputer program.
2. Menurut definisi Thomas H.Cormen algoritma sebagai serangkaian instruksi yang dapat dijalankan untuk melakukan komputasi atau pemprosesan. Dalam buku “*Introduction to Algorithms”* yang ditulis oleh Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, dan Clifford Stein, mendefinisikan algoritma sebagai prosedur langkah demi langkah yang memiliki tujuan tertentu.
3. Menurut pandangan Robert E. Tarjan algoritma sebagai prosedur yang terdiri dari Langkah-langkah yang dapat diikuti untuk mencapai tujuan tertentu.

Berikut jenis-jenis Algoritma dalam pemprograman Komputer

* Algoritma Recursive

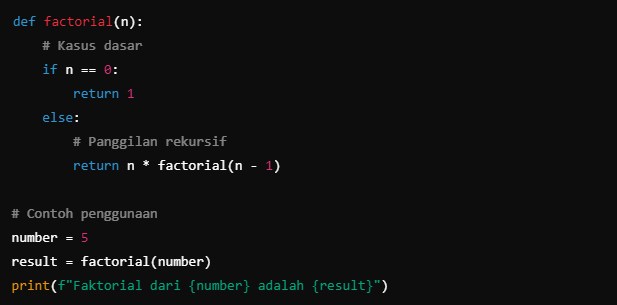
Algoritma Recursive adalah jenis algoritma yang memanggil dirinya sendiri untuk menyelesaikan sub-masalah dari masalah yang kecil hingga masalah yang besar. Menurut Thomas H. Cermen et al. Dalam buku “*Introduction to Algorithms”* Algoritma Recursive adalah cara untuk memecahkan membagi masalah menjadi sub-masalah yang lebih kecil dan serupa. Setiap panggilan Algoritma Recursive menyelesaikan bagian dari masalah hingga mencapai kondisi dasar.

Dalam buku tersebut yang telah di rangkum bersama Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, dan Clifford Stein, mendefinisikan algoritma sebagai prosedur langkah demi langkah yang memiliki tujuan tertentu.

Contohnya Algoritma Recursive:

Faktorial dari bilangan bulat positif nnn (ditulis n!n!n!) adalah hasil kali semua bilangan bulat positif dari 1 hingga nnn. Secara formal:

* N! = n×(n−1)! n! = n \times (n - 1)! n! = n×(n−1)! untuk n>0n > 0n>0
* 0! =10! = 10! =1



**Penjelasan Contoh**

1. **Definisi Fungsi**: Fungsi factorial(n) menerima satu parameter nnn.
2. **Kasus Dasar**:
   * Jika nnn adalah 0, fungsi mengembalikan 1. Ini adalah kondisi yang menghentikan rekursi, karena 0! = 10! = 10! = 1.
3. **Panggilan Rekursif**:
   * Jika nnn lebih besar dari 0, fungsi akan mengembalikan hasil kali nnn dengan hasil dari factorial (n - 1).
   * Ini berarti fungsi memanggil dirinya sendiri dengan nilai nnn yang lebih kecil hingga mencapai kasus dasar.
4. **Contoh Penggunaan**:
   * Jika kita memanggil factorial (5), berikut adalah urutan panggilan fungsi:
     + 5! = 5×4!5! = 5 \times 4!5! = 5×4!
     + 4! = 4×3!4! = 4 \times 3!4! = 4×3!
     + 3! = 3×2!3! = 3 \times 2!3! = 3×2!
     + 2! = 2×1!2! = 2 \times 1!2! = 2×1!
     + 1! = 1×0!1! = 1 \times 0!1! = 1×0!
     + 0! = 10! = 10! = 1 (kasus dasar)
   * Jadi, 5! = 5×4×3×2×1=1205! = 5 \times 4 \times 3 \times 2 \times 1 = 1205! = 5×4×3×2×1=120

**Penjelasan Contoh**

1. **Basis Kasus**:
   * if n == 0 or n == 1:: Ketika sama n dengan 0 atau 1, fungsi mengembalikan 1.
2. **Rekurens**:
   * return n \* factorial (n - 1): Jika nnn lebih besar dari 1, fungsi memanggil dirinya sendiri dengan n−1n - 1n−1 dan mengalikan hasilnya dengan nnn.

**Proses Rekursi**

Jika kita memanggil factorial (5), proses yang terjadi adalah:

* factorial (5) → 5 \* factorial (4)
* factorial (4) → 4 \* factorial (3)
* factorial (3) → 3 \* factorial (2)
* factorial (2) → 2 \* factorial (1)
* factorial (1) → 1 (basis kasus)

Kemudian, hasilnya akan dihitung kembali dari bawah ke atas:

* factorial (2) → 2 \* 1 = 2
* factorial (3) → 3 \* 2 = 6
* factorial (4) → 4 \* 6 = 24
* factorial (5) → 5 \* 24 = 120

Hasil

Ketika menjalankan kode, outputnya akan:

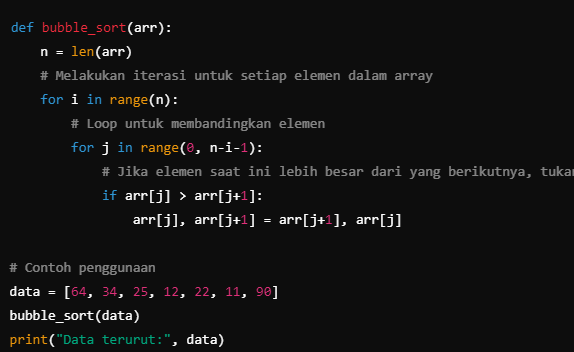
Faktorial dari 5 adalah 120

* Algoritma sorting

Algoritma sorting adalah metode yang digunakan untuk mengurutkan elemen dalam daftar atau array dalam urutan tertentu, baik secara ascending (menaik) atau descending (menurun). Menurut Robert Sedgewick dalam “Algorithms” Algoritma sorting mengatur elemen dalam urutan tertentu untuk mempermudah pencarian atau pemrosesan lebih lanjut. Contoh yang umum termasuk Bubble Sort, Quick Sort, dan Merge Sort, masing-masing dengan kelebihan dan kekurangan dalam hal kompleksitas waktu dan penggunaan memori.

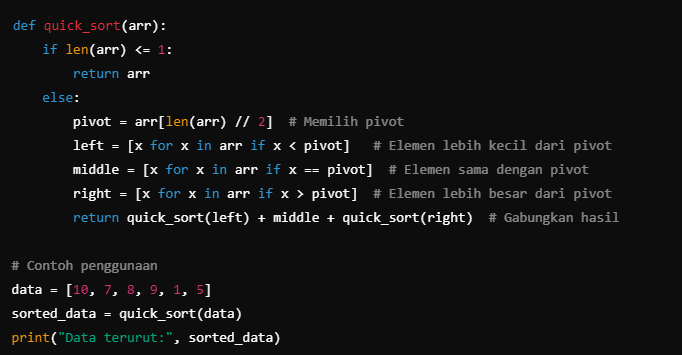
* **Bubble Sort**

Bubble Sort adalah algoritma paling sederhana mudah dipahami yang bekerja berulang kali melangkahi daftar yang perlu disortir membandingkan elemen yang berdekatan dan menukarnya jika mereka berada pada urutan yang salah.



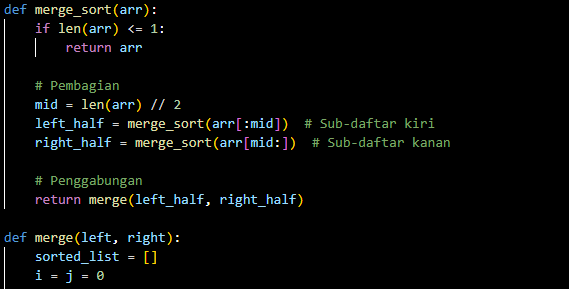
* **Quick Sort**

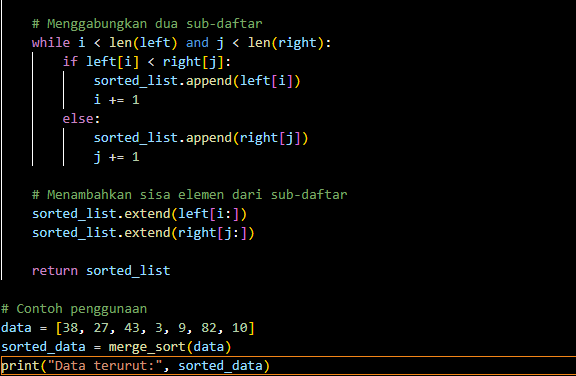
Quick Sort adalah algoritma pengurutan yang paling cepat dan paling tepat sehingga algoritma Quick Sort paling banyak digunakan dalam parktik karna cara pengerjaannya yang cepat dan efisen.



* **Merge Sort**

Merge Sort adalah algoritma pengurutan yang menggunakan pendekatan divide and conquer (pembagian dan penaklukan). Algoritma ini membagi daftar yang akan diurutkan menjadi dua sub-daftar, mengurutkan masing-masing sub-daftar secara rekursif, dan kemudian menggabungkan (merge) kedua sub-daftar tersebut untuk menghasilkan daftar yang terurut.





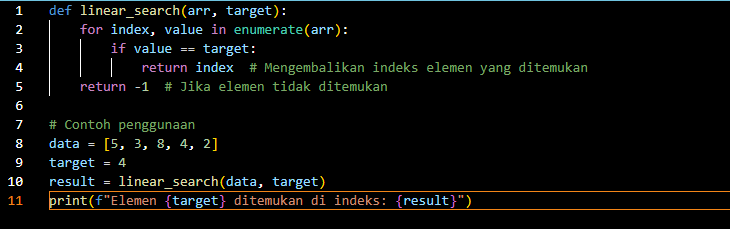
* Algoritma Searching

Algoritma Searching adalah metode linear dan binner yang digunakan untuk menemukan elemen tertentu dalam Kumpulan data, seperti array dan daftar. Algoritma Searching sangat penting dalam pemprograman komputer karna banyak aplikasi yang bergantung pada kemampuan mencari data yang efisien. Menurut William F. Trench dalam bukunya yang berjudul *"An Introduction to Computer Algorithms,"* Trench mendefinisikan algoritma searching sebagai metode untuk menemukan elemen dalam struktur data. Ia menyatakan bahwa pemilihan metode yang tepat tergantung pada jenis data dan kebutuhan aplikasi.

Contohnya

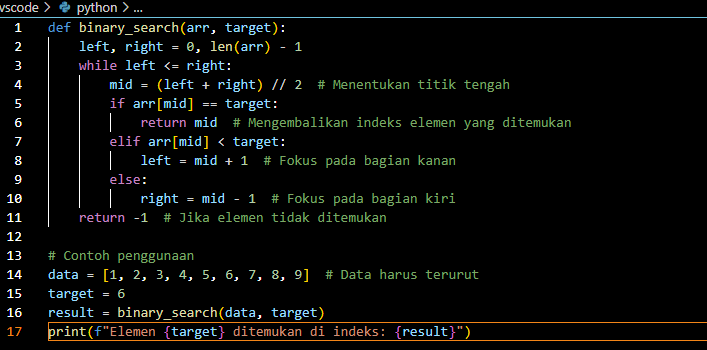
* Pencarian Linear

Pencarian linear adalah metode yang paling sederhana, di mana setiap elemen dalam daftar diperiksa satu per satu hingga elemen yang dicari ditemukan.



* Pencarian Biner

Pencarian biner adalah metode yang lebih efisien tetapi memerlukan daftar yang sudah terurut. Algoritma ini membagi daftar menjadi dua bagian dan memeriksa bagian yang relevan.



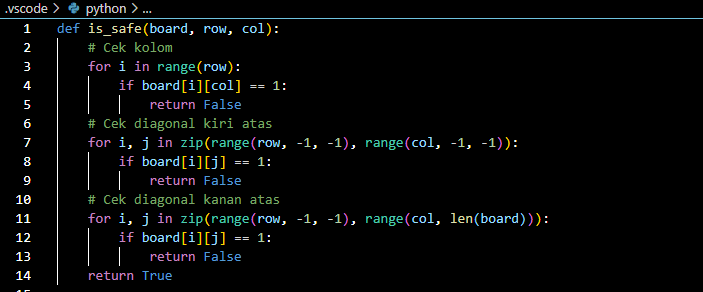
* **Hasil**

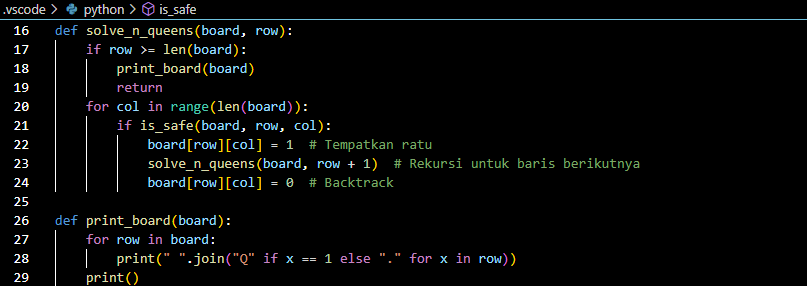
****

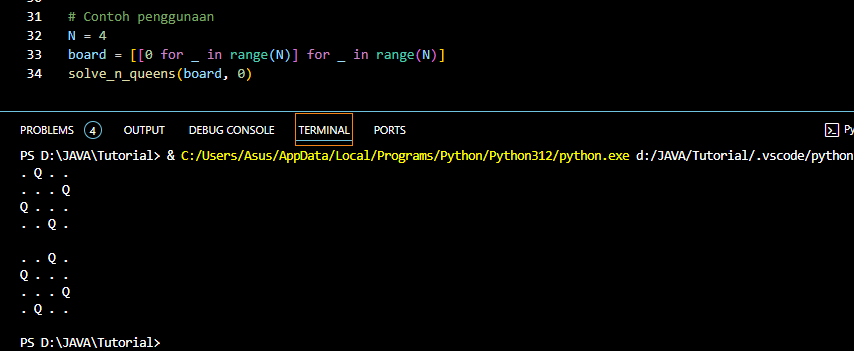
* Algortitma Backtracking

Algortitma Backtracking merupakan salah satu algoritma yang sangat penting dalam pemprograman karna mampu menyelesaikan permasalahan yang sulit yang efisien atau pencarian Solusi misalnya pada masalah N-Queens. Menurut Cormen, Leiserson, Rivest, dan Stein. Dalam *"Introduction to Algorithms,"* para penulis menyatakan bahwa backtracking dapat digunakan untuk menyelesaikan masalah dengan eksplorasi ruang solusi, dan menghindari langkah yang tidak produktif dengan menggunakan teknik "pruning."

Contohnya Implementasi N-Queens







* **Algoritma Greedy**

Algoritma Greedy adalah pendekatan untuk memecahkan masalah optimasi dengan membuat pilihan terbaik (lokal) di setiap langkah, dengan harapan bahwa pilihan tersebut akan mengarah pada solusi optimal secara keseluruhan. Algoritma ini tidak selalu memberikan solusi optimal untuk semua masalah, tetapi sering kali cukup efisien dan efektif untuk masalah tertentu. Menurut V. F. Fomin dan F. A. Fradkin Dalam buku yang berjudul *"Algorithmic Graph Theory,"* menjelaskan bahwa algoritma greedy sering digunakan dalam teori graf dan masalah jaringan, di mana pengambilan keputusan lokal dapat menghasilkan solusi yang baik.

Contoh Impelemtasi



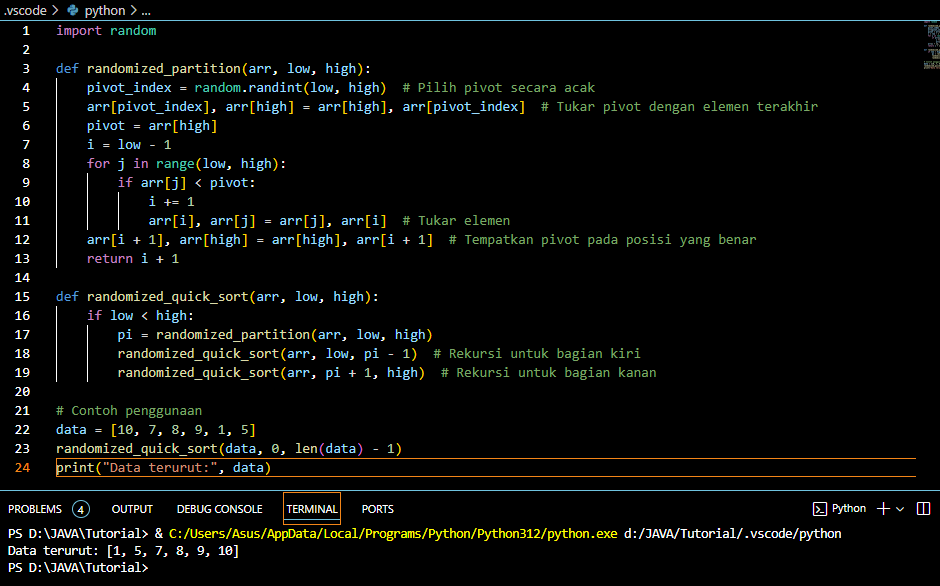
* Algoritma Randomized

Algoritma Randomized adalah algoritma yang kuat dalam pemprograman dan analisis algoritma, memungkinkan penyelesaian masalah yang kompleks dengan cara yang lebih efisien. Pendekatan ini sering kali digunakan untuk meningkatkan efisiensi dalam pemecahan masalah, di mana solusi deterministik mungkin memerlukan waktu yang lebih lama. Dengan menggunakan unsur acak, algoritma randomized dapat menghasilkan solusi yang baik dalam waktu yang lebih singkat.

Menurut Mitzenmacher dan Upfal. Dalam buku *"Probability and Computing,"* menggambarkan bagaimana probabilitas dapat digunakan untuk analisis algoritma, termasuk penggunaan teknik acak untuk meningkatkan kinerja dan mengurangi kompleksitas dalam masalah tertentu dan menurut R. Motwani dan P. Raghavan

Dalam buku *"Randomized Algorithms,"* mereka membahas berbagai metode dan teknik dalam algoritma randomized, serta memberikan contoh aplikasinya dalam masalah nyata, seperti penjadwalan dan pemrograman graf.

Contoh Implementasi Randomized QuickSor



**Penjelasan Kode**

1. **Fungsi** randomized\_partition
   * Memilih pivot secara acak dari array.
   * Memindahkan pivot ke posisi akhir dan membagi array berdasarkan pivot.
   * Mengembalikan indeks pivot yang baru.
2. **Fungsi** randomized\_quick\_sort
   * Menggunakan rekursi untuk membagi array menjadi dua bagian berdasarkan pivot, dan mengurutkan keduanya secara terpisah.
3. **Contoh Penggunaan**:
   * Array yang tidak terurut diurutkan menggunakan Randomized QuickSort, dan hasilnya dicetak.