Nama : Reza Anathasya

Nim : D0424314

Program Study : Sistem Informasi

Tugas : Artikel Tentang 6 Jenis-jenis Algoritma

Enam Jenis Algoritma dan Contohnya

Algoritma adalah serangkaian instruksi langkah demi langkah yang dirancang untuk menyelesaikan masalah atau tugas tertentu. Mereka adalah tulang punggung pemrosesan informasi dan mendasari berbagai teknologi yang kita gunakan setiap hari. Dari mesin pencari hingga aplikasi navigasi, algoritma bekerja di balik layar untuk mengotomatiskan tugas-tugas kompleks dan memberikan hasil yang efisien.

Berikut adalah enam jenis algoritma yang umum digunakan, beserta contohnya yang diambil dari jurnal-jurnal terpercaya:

1. **Algoritma Brute Force: Pendekatan Sederhana yang Seringkali Tidak Efisien**

   Algoritma brute force adalah pendekatan sederhana dan langsung untuk menyelesaikan masalah dengan mencoba semua kemungkinan solusi hingga menemukan solusi yang benar. Dalam arti, algoritma ini "mencoba semua kemungkinan" tanpa menggunakan strategi khusus atau informasi tambahan untuk mempersempit pencarian.

   - **Pengertian menurut para ahli, artikel, dan jurnal:**

   - Metode yang "lempang" dan "naif": Algoritma brute force sering disebut sebagai metode yang "lempang" (straightforward) atau "naif" karena tidak menggunakan strategi khusus dan hanya mencoba semua kemungkinan solusi.

   - Sederhana dan mudah diimplementasikan: Algoritma brute force mudah dipahami dan diimplementasikan, membuatnya berguna sebagai titik awal untuk memecahkan masalah.

   - Tidak selalu mangkus: Algoritma brute force seringkali tidak efisien untuk masalah yang kompleks karena membutuhkan waktu dan sumber daya komputasi yang besar.

   - Basis pembanding: Algoritma brute force sering digunakan sebagai basis pembanding untuk algoritma yang lebih canggih dan efisien.

- Cocok untuk masalah kecil: Algoritma brute force cocok untuk masalah yang berukuran kecil di mana jumlah kemungkinan solusi terbatas.

❖ **Contoh Algoritma Brute Force:**

❖ **Pencarian Kata dalam String**:

Misalkan kita ingin menemukan apakah kata "hello" terdapat dalam string "This is a string with hello in it". Algoritma brute force akan memeriksa setiap kemungkinan posisi kata "hello" dalam string tersebut:

1. Bandingkan "hello" dengan "This" - Tidak cocok

2. Bandingkan "hello" dengan "is" - Tidak cocok

3. Bandingkan "hello" dengan "a" - Tidak cocok

4. ...

5. Bandingkan "hello" dengan "hello" - Cocok!

Algoritma ini akan terus membandingkan "hello" dengan setiap kemungkinan posisi dalam string hingga menemukan kecocokan.

❖ **Kelemahan Algoritma Brute Force:**

- Kompleksitas waktu: Algoritma brute force memiliki kompleksitas waktu yang tinggi, terutama untuk masalah yang kompleks dengan banyak kemungkinan solusi.

- Tidak efisien: Algoritma ini seringkali tidak efisien untuk masalah yang besar karena membutuhkan waktu dan sumber daya komputasi yang besar.

- Tidak selalu optimal: Algoritma brute force tidak selalu menghasilkan solusi optimal, terutama untuk masalah yang memiliki banyak solusi yang mungkin.

❖ **Kapan Menggunakan Algoritma Brute Force:**

Meskipun memiliki kelemahan, algoritma brute force dapat berguna dalam beberapa kasus:

- Masalah sederhana: Untuk masalah sederhana dengan jumlah kemungkinan solusi yang terbatas, algoritma brute force bisa menjadi pilihan yang mudah dan efektif.

- Sebagai titik awal: Algoritma brute force dapat digunakan sebagai titik awal untuk mengembangkan algoritma yang lebih efisien.

- Verifikasi: Algoritma brute force dapat digunakan untuk memverifikasi bahwa solusi yang ditemukan oleh algoritma lain benar.

❖ **Kesimpulan:**

Algoritma brute force adalah pendekatan sederhana dan mudah diimplementasikan, tetapi seringkali tidak efisien untuk masalah yang kompleks. Meskipun demikian, algoritma ini dapat berguna dalam beberapa kasus, terutama sebagai titik awal atau untuk memverifikasi solusi.

2. Algoritma Rekursif: Memecahkan Masalah dengan Diri Sendiri

Algoritma rekursif adalah teknik pemecahan masalah yang melibatkan pemanggilan fungsi itu sendiri, dengan input yang lebih kecil, untuk mencapai solusi. Bayangkan seperti tangga yang menuruni anak tangga sampai mencapai dasar, lalu naik kembali dengan membawa solusi.

❖ **Pengertian menurut para ahli, artikel, dan jurnal:**

- Memecah masalah menjadi sub-masalah: Algoritma rekursif memecah masalah besar menjadi sub-masalah yang lebih kecil, yang memiliki bentuk yang sama dengan masalah aslinya.

- Kasus dasar: Algoritma rekursif memiliki "kasus dasar" yang merupakan kondisi berhenti, di mana solusi dapat langsung dihitung tanpa perlu pemanggilan rekursif lagi.

- Solusi rekursif: Setelah kasus dasar tercapai, solusi sub-masalah digabungkan untuk menghasilkan solusi untuk masalah aslinya.

❖ **Lebih jauh, algoritma recursive masih dibagi lagi menjadi empat tipe spesifik, yaitu:**

- Algoritma Divide and Conquer – Membagi masalah menjadi dua bagian. Yang pertama yaitu masalah itu sendiri dan yang kedua adalah metode pemecahannya.

- Algoritma Dinamis – Menggunakan teknik memoisasi, yaitu menyimpan hasil pemecahan masalah ke memori untuk selanjutnya digunakan lagi di masa mendatang.

-Algoritma Greedy – Bertolak belakang dengan algoritma dinamis, algoritma greedy justru tidak akan mempertimbangkan hasil pemecahan masalah sebelumnya untuk mengambil keputusan.

-Algoritma Backtracking – Menyelesaikan masalah secara bertahap sambil mengeliminasi solusi yang ternyata tidak memecahkan masalah tersebut.

❖ **Contoh Algoritma Rekursif:**

Faktorial:

Faktorial dari bilangan bulat n (dinotasikan sebagai n!) adalah perkalian dari semua bilangan bulat positif dari 1 hingga n. Misalnya, 5! = 5 * 4 * 3 * 2 * 1 = 120.

Berikut adalah implementasi rekursif dari faktorial:

Python

```
def factorial (n):
  if n == 0:
    return 1
  else:
    return n * factorial (n-1)
```

Dalam kode ini:

- Kasus dasar: Jika n adalah 0, fungsi mengembalikan 1 (faktorial dari 0 adalah 1).

- Solusi rekursif: Jika n lebih besar dari 0, fungsi memanggil dirinya sendiri dengan n-1, dan mengalikan hasilnya dengan n.

❖ **Keuntungan Algoritma Rekursif:**

- Kemudahan pemahaman: Algoritma rekursif dapat lebih mudah dipahami dan ditulis untuk beberapa masalah, terutama yang memiliki struktur hierarkis.

- Elegan dan ringkas: Kode rekursif seringkali lebih ringkas dan elegan dibandingkan dengan kode iteratif.

❖ **Kekurangan Algoritma Rekursif:**

- Kompleksitas ruang: Algoritma rekursif dapat memakan banyak ruang memori karena setiap pemanggilan fungsi menempati ruang pada stack.

- Efisiensi: Algoritma rekursif dapat kurang efisien dibandingkan dengan algoritma iteratif, terutama untuk masalah yang besar, karena pemanggilan fungsi berulang dapat memakan waktu.

❖ **Kapan Menggunakan Algoritma Rekursif:**

❖ **Algoritma rekursif cocok untuk masalah yang memiliki struktur rekursif, seperti:**

- Pencarian dalam pohon: Algoritma rekursif sering digunakan untuk melakukan pencarian dalam struktur data pohon.

- Perhitungan rekursif: Algoritma rekursif sangat cocok untuk menghitung nilai rekursif, seperti faktorial, deret Fibonacci, dan sebagainya.

- Pemrosesan data hierarkis: Algoritma rekursif dapat digunakan untuk memproses data yang memiliki struktur hierarkis, seperti folder dan file dalam sistem file.

3. Algoritma Randomized: Menambahkan Unsur Acak untuk Efisiensi dan Ketidakpastian

Algoritma randomized adalah algoritma yang menggunakan keacakan dalam proses pengambilan keputusan atau dalam pemilihan langkah-langkahnya. Mereka memanfaatkan proses acak untuk mencapai solusi yang optimal atau mendekati optimal, atau untuk meningkatkan efisiensi dalam kasus-kasus tertentu.

❖ Pengertian menurut para ahli, artikel, dan jurnal:

- Penggunaan keacakan: Algoritma randomized menggunakan proses acak untuk menentukan langkah-langkah yang diambil dalam menyelesaikan masalah.

- Solusi mendekati optimal: Algoritma randomized seringkali dirancang untuk menghasilkan solusi yang mendekati optimal, bukan solusi optimal yang pasti.

- Peningkatan efisiensi: Dalam beberapa kasus, algoritma randomized dapat meningkatkan efisiensi dibandingkan dengan algoritma deterministik, terutama untuk masalah yang kompleks.

❖ Contoh Algoritma Randomized:

Quicksort Acak:

Quicksort adalah algoritma pengurutan yang efisien. Dalam quicksort acak, pivot (elemen yang digunakan untuk membagi data) dipilih secara acak. Ini membantu mengurangi kemungkinan kasus terburuk di mana data sudah terurut, yang dapat menyebabkan kinerja quicksort menjadi sangat lambat.

Algoritma Monte Carlo:

Algoritma Monte Carlo adalah kelas algoritma yang menggunakan simulasi acak untuk memperkirakan nilai atau solusi untuk masalah yang kompleks. Mereka sering digunakan dalam bidang seperti keuangan, fisika, dan ilmu komputer.

❖ Keuntungan Algoritma Randomized:

- Efisiensi: Algoritma randomized dapat lebih efisien dibandingkan dengan algoritma deterministik untuk masalah tertentu.

- Sederhana: Algoritma randomized dapat lebih mudah diimplementasikan dibandingkan dengan algoritma deterministik.

- Menghindari kasus terburuk: Algoritma randomized dapat membantu menghindari kasus terburuk yang dapat menyebabkan kinerja algoritma deterministik menjadi buruk.

❖ Kekurangan Algoritma Randomized:

- Ketidakpastian: Algoritma randomized tidak memberikan jaminan untuk menemukan solusi optimal.

- Hasil yang bervariasi: Hasil dari algoritma randomized dapat bervariasi dari satu eksekusi ke eksekusi lainnya.

❖ Kapan Menggunakan Algoritma Randomized:

❖ Algoritma randomized cocok untuk masalah yang:

- Memiliki banyak kemungkinan solusi: Algoritma randomized dapat membantu mempersempit pencarian solusi.

- Sulit untuk dipecahkan secara deterministik: Algoritma randomized dapat memberikan solusi yang mendekati optimal.

- Membutuhkan efisiensi: Algoritma randomized dapat meningkatkan efisiensi dibandingkan dengan algoritma deterministik.

❖ Kesimpulan:

Algoritma randomized adalah alat yang kuat yang dapat digunakan untuk memecahkan berbagai masalah. Mereka menawarkan keuntungan dalam hal efisiensi dan kemampuan untuk menghindari kasus terburuk. Meskipun tidak selalu menjamin solusi optimal, algoritma randomized dapat memberikan solusi yang mendekati optimal dengan tingkat ketidakpastian tertentu.

4. Algoritma Sorting: Mengatur Data Menjadi Urutan Tertentu

Algoritma sorting adalah algoritma yang digunakan untuk mengatur elemen dalam kumpulan data (seperti array atau daftar) dalam urutan tertentu, seperti urutan menaik atau menurun. Sorting adalah operasi dasar dalam ilmu komputer dan memiliki banyak aplikasi dalam berbagai bidang, termasuk basis data, pencarian informasi, dan analisis data.

❖ Pengertian menurut para ahli, artikel, dan jurnal:

- Proses pengurutan: Sorting adalah proses menyusun kembali kumpulan objek menggunakan tata aturan tertentu.

- Tujuan utama: Tujuan utama dari sorting adalah untuk mengurutkan data berdasarkan keinginan, baik itu dari yang terendah maupun yang tertinggi, sehingga data yang dihasilkan akan lebih terstruktur, teratur, dan sesuai dengan kebutuhan.

- Keuntungan: Sorting memiliki banyak keuntungan, seperti:

- Meningkatkan efisiensi pencarian data.

- Memudahkan pengolahan data.

- Memudahkan analisis data.

- Meningkatkan keterbacaan data.

❖ Contoh Algoritma Sorting:

1. Bubble Sort:

Bubble Sort adalah algoritma sorting yang sederhana dan mudah dipahami. Algoritma ini membandingkan elemen berdekatan dan menukar mereka jika tidak dalam urutan yang benar. Proses ini diulang hingga tidak ada lagi pertukaran yang terjadi.

2. Insertion Sort:

Insertion Sort bekerja dengan mengambil satu elemen dari data yang belum diurutkan dan menempatkannya pada posisi yang benar dalam data yang sudah diurutkan. Proses ini diulang hingga semua elemen telah diurutkan.

3. Selection Sort:

Selection Sort memilih elemen terkecil dari data yang belum diurutkan dan menukarnya dengan elemen pertama. Proses ini diulang hingga semua elemen telah diurutkan.

4. Merge Sort:

Merge Sort adalah algoritma sorting yang efisien. Algoritma ini memecah data menjadi dua bagian, mengurutkan setiap bagian secara terpisah, dan kemudian menggabungkan kedua bagian yang telah diurutkan menjadi satu bagian yang terurut.

5. Quick Sort:

Quick Sort adalah algoritma sorting yang cepat dan efisien. Algoritma ini memilih pivot (elemen yang digunakan untuk membagi data) dan membagi data menjadi dua bagian: elemen yang lebih kecil dari pivot dan elemen yang lebih besar dari pivot. Proses ini diulang secara rekursif untuk setiap bagian hingga semua elemen telah diurutkan.

6. Heap Sort:

Heap Sort adalah algoritma sorting yang menggunakan struktur data heap. Heap adalah struktur data pohon biner yang mempertahankan properti heap: nilai parent selalu lebih besar dari atau sama dengan nilai child-nya. Heap Sort membangun heap dari data dan kemudian mengekstrak elemen terbesar dari heap secara berulang hingga semua elemen telah diurutkan.

❖ **Kesimpulan:**

Algoritma sorting adalah alat yang penting dalam ilmu komputer dan memiliki banyak aplikasi. Memilih algoritma sorting yang tepat bergantung pada kebutuhan dan karakteristik data yang akan diurutkan. Beberapa algoritma sorting lebih efisien untuk kasus tertentu, sementara algoritma lain lebih mudah diimplementasikan. Memahami berbagai jenis algoritma sorting dan cara kerjanya dapat membantu Anda memilih algoritma yang paling tepat untuk kebutuhan Anda.

5. Algoritma Pencarian: Menemukan Jarum di Tumpukan Jerami

Algoritma pencarian adalah serangkaian langkah yang digunakan untuk menemukan elemen tertentu dalam kumpulan data. Mereka adalah alat penting dalam ilmu komputer dan digunakan dalam berbagai aplikasi, dari mesin pencari hingga sistem basis data.

❖ **Pengertian menurut para ahli, artikel, dan jurnal:**

- Mencari solusi: Algoritma pencarian adalah algoritma yang menerima masukan berupa masalah dan menghasilkan solusi untuk masalah tersebut, yang biasanya didapat dari evaluasi beberapa kemungkinan solusi.

- Metode pencarian: Algoritma pencarian dapat dibagi menjadi dua kategori:

- Pencarian brute-force: Metode sederhana dan intuitif yang mencoba semua kemungkinan solusi.

- Pencarian informed: Metode yang menggunakan informasi tambahan tentang struktur masalah untuk mempersempit pencarian.

- Mencari elemen dalam list: Algoritma pencarian list adalah algoritma paling dasar yang bertujuan mencari elemen tertentu dalam list dengan kunci tertentu.

❖ **Contoh Algoritma Pencarian:**

1. Pencarian Linier (Linear Search):

Pencarian linier adalah algoritma pencarian yang paling sederhana. Algoritma ini memeriksa setiap elemen dalam list secara berurutan hingga elemen yang dicari ditemukan. Jika elemen tidak ditemukan, algoritma akan mengembalikan nilai "tidak ditemukan".

2. Pencarian Biner (Binary Search):

Pencarian biner adalah algoritma pencarian yang lebih efisien dibandingkan dengan pencarian linier. Algoritma ini hanya dapat digunakan pada list yang sudah terurut. Algoritma ini bekerja dengan membagi list menjadi dua bagian dan memeriksa elemen tengah. Jika elemen tengah adalah elemen yang dicari, algoritma akan mengembalikan nilai "ditemukan". Jika elemen tengah lebih besar dari elemen yang dicari, algoritma akan mencari di bagian kiri list. Jika elemen tengah lebih kecil dari elemen yang dicari, algoritma akan mencari di bagian kanan list. Proses ini diulang hingga elemen yang dicari ditemukan atau list habis.

3. Pencarian Hash (Hash Search):

Pencarian hash adalah algoritma pencarian yang sangat efisien. Algoritma ini menggunakan fungsi hash untuk memetakan elemen ke dalam tabel hash. Fungsi hash adalah fungsi yang mengambil elemen sebagai input dan mengembalikan nilai hash sebagai output. Tabel hash adalah struktur data yang menyimpan elemen berdasarkan nilai hash-nya. Untuk mencari elemen tertentu, algoritma hash akan menghitung nilai hash dari elemen tersebut dan kemudian mencari elemen tersebut di tabel hash berdasarkan nilai hash-nya.

❖ **Keuntungan Algoritma Pencarian:**

- Efisiensi: Algoritma pencarian dapat membantu menemukan elemen tertentu dengan cepat, terutama untuk kumpulan data yang besar.

- Fleksibel: Algoritma pencarian dapat digunakan untuk berbagai jenis data dan struktur data.

❖ **Kekurangan Algoritma Pencarian:**

- Kompleksitas: Algoritma pencarian dapat menjadi kompleks untuk diimplementasikan, terutama untuk algoritma pencarian yang lebih canggih.

- Keterbatasan: Beberapa algoritma pencarian, seperti pencarian biner, hanya dapat digunakan pada data yang terurut.

❖ **Kapan Menggunakan Algoritma Pencarian:**

❖ **Algoritma pencarian cocok untuk masalah yang:**

- Membutuhkan pencarian elemen tertentu: Algoritma pencarian dapat membantu menemukan elemen tertentu dengan cepat.

- Memiliki kumpulan data yang besar: Algoritma pencarian dapat membantu mempersempit pencarian elemen tertentu dalam kumpulan data yang besar.

- Membutuhkan efisiensi: Algoritma pencarian dapat membantu meningkatkan efisiensi proses pencarian.

❖ **Kesimpulan:**

Algoritma pencarian adalah alat yang penting dalam ilmu komputer dan memiliki banyak aplikasi. Memilih algoritma pencarian yang tepat bergantung pada kebutuhan dan karakteristik data yang akan dicari. Beberapa algoritma pencarian lebih efisien untuk kasus tertentu, sementara algoritma lain lebih mudah diimplementasikan. Memahami berbagai jenis algoritma pencarian dan cara kerjanya dapat membantu Anda memilih algoritma yang paling tepat untuk kebutuhan Anda.

6. Algoritma Hashing: Mengubah Data Menjadi Kode Unik

Algoritma hashing adalah proses mengubah data dengan panjang variabel menjadi data dengan panjang tetap, yang disebut hash atau message digest. Proses ini menggunakan fungsi hash, yang merupakan fungsi matematika yang menghasilkan output unik untuk setiap input yang berbeda.

❖ **Pengertian menurut para ahli, artikel, dan jurnal:**

- Transformasi data: Algoritma hashing adalah proses transformasi data input menjadi output yang lebih pendek dan unik.

- Fungsi hash: Fungsi hash adalah fungsi matematika yang menerima data input dan menghasilkan hash atau message digest.

- Deterministik: Fungsi hash bersifat deterministik, artinya input yang sama akan selalu menghasilkan output hash yang sama.

- Satu arah: Fungsi hash umumnya bersifat satu arah, artinya sulit untuk mendapatkan input asli dari output hash.

❖ **Contoh Algoritma Hashing:**

- MD5 (Message Digest 5): Algoritma hashing yang menghasilkan hash 128-bit. MD5 dulunya banyak digunakan, tetapi sekarang dianggap tidak aman karena telah

ditemukan metode untuk menghasilkan tabrakan (dua input yang berbeda menghasilkan output hash yang sama).

- SHA-1 (Secure Hash Algorithm 1): Algoritma hashing yang menghasilkan hash 160-bit. SHA-1 juga dianggap tidak aman karena telah ditemukan metode untuk menghasilkan tabrakan.

- SHA-256 (Secure Hash Algorithm 256): Algoritma hashing yang menghasilkan hash 256-bit. SHA-256 dianggap lebih aman dibandingkan dengan MD5 dan SHA-1, dan banyak digunakan dalam teknologi blockchain.

- SHA-3 (Secure Hash Algorithm 3): Algoritma hashing yang menghasilkan hash dengan berbagai panjang, termasuk 224-bit, 256-bit, 384-bit, dan 512-bit. SHA-3 dirancang untuk menjadi lebih aman daripada SHA-1 dan SHA-2.

❖ **Keuntungan Algoritma Hashing:**

- Integritas data: Algoritma hashing dapat digunakan untuk memverifikasi integritas data. Jika hash dari data berubah, berarti data telah dimodifikasi.

- Keamanan: Algoritma hashing dapat digunakan untuk melindungi data sensitif, seperti kata sandi. Hash dari kata sandi disimpan, bukan kata sandi asli, sehingga jika data dicuri, kata sandi asli tidak akan terungkap.

- Efisiensi: Algoritma hashing dapat digunakan untuk meningkatkan efisiensi operasi pencarian. Data dapat diindeks menggunakan hash, sehingga pencarian data dapat dilakukan dengan cepat.

❖ **Kekurangan Algoritma Hashing:**

- Tabrakan: Meskipun fungsi hash dirancang untuk menghasilkan output unik untuk setiap input yang berbeda, ada kemungkinan terjadi tabrakan (dua input yang berbeda menghasilkan output hash yang sama). Namun, kemungkinan terjadinya tabrakan sangat kecil untuk algoritma hashing yang kuat.

- Tidak dapat dibalik: Fungsi hash umumnya bersifat satu arah, artinya sulit untuk mendapatkan input asli dari output hash. Ini dapat menjadi masalah jika Anda perlu memulihkan data asli dari hash-nya.

❖ **Kapan Menggunakan Algoritma Hashing:**

❖ **Algoritma hashing cocok untuk masalah yang:**

- Membutuhkan verifikasi integritas data: Algoritma hashing dapat digunakan untuk memverifikasi integritas data, seperti file atau dokumen.

- Membutuhkan perlindungan data sensitif: Algoritma hashing dapat digunakan untuk melindungi data sensitif, seperti kata sandi atau informasi keuangan.

- Membutuhkan peningkatan efisiensi pencarian: Algoritma hashing dapat digunakan untuk meningkatkan efisiensi operasi pencarian, seperti pencarian data dalam database.

❖ **Kesimpulan:**

Algoritma hashing adalah alat yang penting dalam ilmu komputer dan memiliki banyak aplikasi. Memilih algoritma hashing yang tepat bergantung pada kebutuhan dan tingkat keamanan yang dibutuhkan. Algoritma hashing yang kuat dapat membantu melindungi data sensitif dan memverifikasi integritas data.