TUGAS ALGORITMA DAN STRUKTUR DATA

MUHAMMAD ARIF, D0424507

PROGRAM STUDI SISTEM INFORMASI

"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 dibalik 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 keinginan" tanpa menggunakan strategi khusus atau informasi tambahan untuk mempersempit pencarian.

Contoh Algoritma Brute Force:

- **Pencarian kata dalam String:**

  Misalkan kita ingin menemukan apakah kata "hello" terdapat dalam sidang "Thjs is a string with hello in it". Algoritma brute force akan menerima 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 membandigkan "hello" dengan setiap kemungkinan posisi dalam string hingga menemukan kecocokan.

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 mmencapai solusi. Bayangkan seperti tangga yang menuruni anak tangga sampai mencapai dasar, lalu naik kembali dengan membawa solusi.

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 ini adalah implementasi rekursif dari factorial:

Phyton

Def factorial (n):

If n === 0:

Return 1

Else: returm n* factorial (n-1)

Dalam kode ini:

- Kasus dasar: Jika n adalah 0, fungsi mengembalikan 1 (factorial dari 0 adalah 1).
- Solusi rekursif: Jika n lebih besar dari 0, fungsi memanggil dirinya sendiri dengan n-1, dan mengalihkan hasilnya dengan n.

Kesimpulan:

Algoritma rekursif adalah algoritma yang akan memanggil dirinya sendiri berulang kali hingga masalah yang dihadapi terpecahkan. Algoritma rekursif dapat digunakan untuk memecahkan masalah yang rumit karena dapat memecah masalah besar menjadi submasalah yang lebih kecil dan mudah dikelola.

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.

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.

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 selalu 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 ilmu computer dan memiliki banyak aplikasi dalam berbagai bidang, termasuk basis data, pencarian informasi, dan analisis data.

Contoh Algoritma Sorting:

- **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.

- **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.
- **Selection Sort:**

  Selection sort memilih elemen terkecil dari data yang belum diurutkan dan memukarnya dengan elemen pertama. Proses ini diulang hingga semua elemen telah di urutkan.
- **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.
- **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.
- **Heap Sort:**

  Heap sort adalah algoritma sorting yang menggunakan struktur data heap. Heap adalah struktur data pohon biner yang mempertahankan property 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 computer 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 di implementasikan. 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 computer dan digunakan dalam berbagai aplikasi, dari mesin pencari hingga sistem basis data.

Contoh Algoritma Pencarian:

- **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".
- **Pencarian Biner (Binary Search):**
  Pencarian biner adalah algoritma pencarian yang efisien dibandingkan dengan pencarian linier. Algoritma ini hanya dapat digunakan pada list yang sudah terurut. Algoritma ini hanya dapat digunakan pada list yang sudah terurut. Algoritma ini bekerja dengan membagi list menjadi dua bagian dan memeriksa elemen Tengah.
- **Pencarian Hash (Hash Search):**
  Pencarian hash adalah algoritma pencarian yang sangat efisien. Algoritma ini menggunakan fungsi hash untuk memetakan elemen kedalam table 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 table hash berdasrkan nilai hash-nya.

Kesimpulan:

Algoritma pencarian adalah alat yang penting dalam ilmu computer dan memiliki banyak aplikasi. Memilih algoritma pencarian yang tepat bergantung pada kebutuhan dan karaktreristik data yang akan dicari. Beberapa algoritma lain lebih mudah diimplementasikan. Memahami berbagai jenis algoritma pencarian dan cara kerjanya dapat membantu anda memiliki 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.

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 inout yang berbeda menghasilkan output hash yang sama).
- SHA-1 (Secure Hash Algoritma 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 Algoritma 256): algoritma hashing yang menghasilkan hash 256-bit. SHA-256 dianggap lebih aman dibandingkan dengan MD5 dan SHA-1, dan banyak digunakan dalam teknologi blockhain.
- SHA-3 (Secure Hash Algoritma 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.

Kesimpulan:

Algoritma hashing adalah alat yang penting dalam ilmu computer 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 integritasn data.