Algoritma linier waktu adalah algoritma yang kompleksitas waktunya proporsional terhadap ukuran input. Dalam notasi Big-O, kompleksitas waktu algoritma ini dinyatakan sebagai ONo, di mana n adalah ukuran input. Artinya, jika ukuran input meningkat, waktu yang diperlukan untuk menyelesaikan algoritma akan meningkat dengan laju yang sama.

Contoh Algoritma Linier Waktu

Pencarian Linier (Linear Search):

Dalam pencarian linier, setiap elemen dalam array diperiksa satu per satu hingga elemen yang dicari ditemukan atau seluruh array telah diperiksa. Jika ukuran array adalah n, maka dalam kasus terburuk (elemen yang dicari berada di posisi terakhir atau tidak ada dalam array), algoritma ini memerlukan n pemeriksaan.

Kompleksitas waktu: ONo.

Penghitungan Jumlah Elemen dalam Array:

Misalkan Anda ingin menghitung jumlah total elemen dalam sebuah array. Anda harus mengunjungi setiap elemen dalam array sekali untuk menambahkannya ke total.

Kompleksitas waktu: ONo.

Algoritma Kadane untuk Subarray Maksimal:

Algoritma Kadane digunakan untuk menemukan subarray kontinu dengan jumlah terbesar dalam array yang mengandung bilangan bulat negatif dan positif.

Kompleksitas waktu: ONo.

Analisis Detil Algoritma Linier Waktu

Notasi Big-O (ONo):

Notasi Big-O adalah cara untuk menggambarkan kompleksitas waktu sebuah algoritma dalam hal pertumbuhan fungsionalnya seiring dengan peningkatan ukuran input. Dalam konteks ONo, ini berarti bahwa waktu eksekusi algoritma akan meningkat secara linear dengan ukuran input.

Kasus Terbaik, Rata-rata, dan Kasus Terburuk:

Kasus Terbaik: Situasi di mana algoritma membutuhkan jumlah langkah minimum untuk menyelesaikan tugasnya. Misalnya, dalam pencarian linier, elemen yang dicari mungkin berada di posisi pertama.

Kasus Rata-rata: Situasi di mana algoritma beroperasi dengan jumlah langkah rata-rata yang dapat diharapkan.

Kasus Terburuk: Situasi di mana algoritma membutuhkan jumlah langkah maksimum untuk menyelesaikan tugasnya. Dalam pencarian linier, ini terjadi jika elemen yang dicari berada di posisi terakhir atau tidak ada dalam array.

Kelebihan dan Kekurangan Algoritma Linier Waktu:

Kelebihan:

Sederhana dan mudah diimplementasikan.

Efisien untuk ukuran input yang kecil hingga sedang.

Kekurangan:

Meskipun linier, algoritma ini mungkin tidak efisien untuk ukuran input yang sangat besar jika ada algoritma lain yang lebih efisien untuk masalah yang sama.

Berikut ini adalah contoh implementasi algoritma pencarian linier dalam bahasa pemrograman Python:

Penjelasan Kode

Fungsi linear\_search:

Fungsi ini menerima dua parameter: arr (array yang akan dicari) dan target (elemen yang dicari).

Fungsi ini mengiterasi setiap elemen dalam array menggunakan loop for.

Jika elemen yang dicari ditemukan, fungsi mengembalikan indeks dari elemen tersebut.

Jika elemen tidak ditemukan setelah memeriksa semua elemen dalam array, fungsi mengembalikan -1.

Contoh Penggunaan:

array adalah array yang diberikan.

target adalah elemen yang ingin dicari dalam array.

Hasil pencarian disimpan dalam result.

Jika result bukan -1, itu berarti elemen ditemukan dan indeks elemen ditampilkan.

Jika result adalah -1, pesan bahwa elemen tidak ditemukan ditampilkan.

Algoritma Coin Change adalah algoritma yang digunakan untuk menemukan cara optimal untuk memberikan sejumlah uang tertentu menggunakan koin-koin dengan denominasi tertentu. Algoritma ini sering digunakan dalam masalah optimasi dan pemrograman dinamis.

**Contoh Masalah Coin Change**

Misalkan Anda memiliki koin dengan denominasi 1, 5, dan 10. Anda ingin memberikan sejumlah uang sebanyak 18. Berapa jumlah minimum koin yang diperlukan untuk mencapai jumlah tersebut?

**Solusi dengan Pendekatan Pemrograman Dinamis**

Pendekatan pemrograman dinamis melibatkan penyimpanan hasil perhitungan sebelumnya untuk menghindari perhitungan ulang yang tidak perlu. Kita dapat membuat sebuah array dp di mana dp[i] menyimpan jumlah minimum koin yang diperlukan untuk mencapai jumlah i.

**Langkah-langkah Algoritma**

1. **Inisialisasi Array dp**:
   * Buat array dp dengan ukuran amount + 1 dan inisialisasi dengan nilai yang sangat besar (misalnya, inf), kecuali dp[0] yang diinisialisasi dengan 0 karena tidak ada koin yang diperlukan untuk mencapai jumlah 0.
2. **Iterasi Denominasi Koin**:
   * Untuk setiap koin dalam denominasi yang diberikan, perbarui nilai dalam array dp dengan iterasi dari jumlah koin tersebut hingga jumlah yang diinginkan.
3. **Perbarui Array dp**:
   * Untuk setiap jumlah i dari denominasi koin hingga amount, perbarui dp[i] dengan memilih nilai minimum antara dp[i] saat ini dan dp[i - coin] + 1.
4. **Hasil Akhir**:
   * Nilai dp[amount] akan menyimpan jumlah minimum koin yang diperlukan untuk mencapai jumlah amount. Jika nilainya masih inf, berarti tidak ada kombinasi koin yang dapat mencapai jumlah tersebut.

Berikut ini adalah contoh implementasi dalam bahasa pemrograman Python:

**Penjelasan Kode**

1. **Fungsi coinChange**:
   * Fungsi ini menerima dua parameter: coins (array dari denominasi koin yang tersedia) dan amount (jumlah uang yang ingin dicapai).
   * dp adalah array yang menyimpan jumlah minimum koin untuk setiap jumlah dari 0 hingga amount. Array ini diinisialisasi dengan nilai inf (tak terhingga) kecuali dp[0] yang diinisialisasi dengan 0.
2. **Iterasi Denominasi Koin**:
   * Untuk setiap koin dalam coins, iterasi dilakukan dari nilai koin tersebut hingga amount.
   * dp[i] diperbarui dengan nilai minimum antara dp[i] saat ini dan dp[i - coin] + 1, yang berarti menggunakan satu koin dari denominasi coin untuk mencapai jumlah i.
3. **Hasil Akhir**:
   * Jika dp[amount] bukan inf, maka jumlah minimum koin untuk mencapai amount ditemukan dan dikembalikan. Jika dp[amount] adalah inf, itu berarti tidak mungkin mencapai amount dengan denominasi koin yang diberikan, dan fungsi mengembalikan -1.

Deret Fibonacci adalah serangkaian angka di mana setiap angka (setelah dua angka pertama) adalah jumlah dari dua angka sebelumnya. Deret ini dinamai sesuai dengan nama matematikawan Italia, Leonardo Fibonacci, yang memperkenalkannya ke dunia Barat dalam buku "Liber Abaci" yang diterbitkan pada tahun 1202.

**Definisi Formal**

Secara formal, deret Fibonacci didefinisikan sebagai berikut:

* F(0) = 0F(0)=0
* F(1)=1
* Untuk n≥2,  
  F = F(n-1) + F(n-2)F=F(n−1)+F(n−2)

Ini menghasilkan deret seperti ini: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, dan seterusnya.

**Ciri-ciri Deret Fibonacci**

1. **Sifat Rekursif**: Setiap elemen dalam deret adalah jumlah dari dua elemen sebelumnya.
2. **Rasio Emas**: Rasio dua angka berturut-turut dalam deret Fibonacci mendekati Rasio Emas (ϕ≈1.618) seiring bertambahnya angka dalam deret.
3. **Polinomial Generating Function**: Polinomial generating function untuk deret Fibonacci adalah x1−x−x2\frac{x}{1 - x - x^2}​.

**Penggunaan Deret Fibonacci**

1. **Matematika dan Komputasi**:
   * Digunakan dalam analisis algoritma, seperti dalam algoritma pencarian dan pengurutan.
   * Basis untuk pendekatan dinamis dalam menyelesaikan masalah yang lebih kompleks, seperti masalah knapsack dan masalah pemotongan batang.
2. **Biologi**:
   * Muncul dalam pola pertumbuhan tanaman, seperti jumlah daun dalam spiral atau distribusi bunga matahari.
   * Digunakan untuk menggambarkan pola reproduksi pada organisme tertentu.
3. **Ekonomi dan Keuangan**:
   * Digunakan dalam analisis teknis untuk menentukan level support dan resistance dalam pasar saham melalui urutan Fibonacci retracement.
4. **Seni dan Arsitektur**:
   * Banyak digunakan dalam desain dan arsitektur karena hubungan eratnya dengan Rasio Emas, yang dianggap estetis.

[Permalink](https://e-learning.unpam.ac.id/mod/forum/discuss.php?d=455759#p32358536)[Reply](https://e-learning.unpam.ac.id/mod/forum/post.php?reply=32358536#mformforum)