# JENIS-JENIS ALGORITMA PEMROGRAMAN

**Isda Yunisari**

**Universitas Sulawesi Barat**

Isdayunisari053@gmail.com

## PENDAHULUAN

Saat ini, Anda mungkin telah mendengar banyak tentang apa itu algoritma. Ya, algoritma sangat penting dalam dunia pemrograman.

Bagi Anda yang sudah lama berkecimpung di dunia teknik komputer pasti mengenal istilah ini. Namun, bagi yang baru menggunakan perangkat ini, tentu tidak jelas mengenai komponen, fitur, dan tujuan penggunaan perangkat ini. Algoritma pemrograman adalah serangkaian langkah-langkah yang terstuktur dan logis untuk menyelesaikan masalah atau tugas tertentu dalam pemrograman komputer.

Perkembangan ilmu pengetahuan dan teknologi memungkinkan manusia menghasilkan karya yang semakin canggih dan kompleks. Meskipun komputer dapat melakukan perhitungan lebih cepat daripada manusia pada umumnya, komputer tidak dapat memecahkan masalah tanpa diajarkan oleh manusia melalui urutan langkah (algoritma) yang telah ditentukan sebelumnya.

Selain digunakan untuk menyelesaikan masalah oleh komputer, algoritma juga dapat diterapkan untuk menyelesaikan masalah sehari-hari yang membutuhkan serangkaian proses atau langkah-langkah prosedural. Beberapa ahli menganggap algoritma sebagai urutan langkah-langkah yang harus diikuti dalam matematika atau perhitungan untuk memecahkan masalah lain, terutama komputer. Selain itu, Algoritma juga berisi serangkaian instruksi terbatas serta instruksi khusus untuk menghitung fungsi yang, ketika dieksekusi dan diproses, menghasilkan output tertentu dan kemudian berhenti pada kondisi terminasi yang ditentukan. Algoritma adalah suatu langkah atau metode yang telah direncanakan secara matang sehingga terurut dan terorganisir dengan baik dan biasanya digunakan untuk memecahkan suatu masalah dengan memberikan suatu instruksi sehingga menjadi suatu tindakan.. Ketika algoritma tersebut digunakan pada komputer, maka akan menghasilkan output yang kemudian akan berhenti pada keadaan awal. Tidak hanya pada komputer atau dalam kehidupan sehari-hari, algoritma juga digunakan oleh banyak perusahaan terutama yang bergerak di bidang keuangan. Hampir semua bidang perusahaan keuangan menggunakan algoritma, mulai dari perdagangan saham dan aset hingga manajemen utang dan penetapan harga pinjaman. Trading menggunakan algoritma sering disebut sebagai trading otomatis. Perdagangan otomatis sering menggunakan program komputer untuk menjual atau membeli sekuritas dengan cepat, mengapa menggunakan program komputer? Hal ini karena transaksi bisnis terjadi sangat cepat, sehingga kecepatan tidak mungkin dihitung sama manusia. Jadi algoritma jelas merupakan salah satu metode

dimana suatu masalah dapat diselesaikan dengan mudah, sehingga suatu operasi atau proses produksi dapat berlanjut. Selanjutnya, algoritma merupakan metode yang dapat dipelajari oleh semua orang.

## PEMBAHASAN

Di dunia pemrograman, algoritma berfungsi sebagai peta jalan yang menuntun kita untuk menyelesaikan berbagai masalah secara sistematis. Dengan memahami berbagai jenis algoritma dan aplikasinya, programmer dapat merancang solusi yang lebih efisien dan efektif, sehingga mengoptimalkan performa perangkat lunak yang dihasilkan. Berikut beberapa jenis algoritma:

A. Algoritma Recursive

Menurut Omkar Prabhakar Ghadage Algoritme Recursive memanggil dirinya sendiri dengan nilai masukan yang lebih kecil dan mengembalikan hasil untuk masukan saat ini dengan melakukan operasi dasar pada nilai yang dikembalikan untuk masukan yang lebih kecil. Secara umum jika suatu masalah dapat dipecah dengan menerapkan solusi pada versi yang lebih kecil menyusut contoh yang mudah dipecahkan, maka masalah tersebut dapat dipecahkan menggunakan algoritma recurvise.

Contoh:

Untuk membuat algoritma rekursif, Anda akan membagi pernyataan masalah yang diberikan menjadi dua bagian. Bagian pertama adalah kasus dasar, dan bagian kedua adalah langkah rekursif.

- Kasus Dasar: Ini hanyalah contoh paling sederhana dari suatu masalah, yang terdiri dari suatu kondisi yang mengakhiri fungsi rekursif. Kasus dasar ini mengevaluasi hasil ketika suatu kondisi tertentu terpenuhi.

-Langkah Rekursif: Menghitung hasil dengan melakukan pemanggilan rekursif ke fungsi yang sama, tetapi dengan input yang dikurangi ukuran atau kompleksitasnya.

Misalnya, perhatikan pernyataan masalah ini: Cetak jumlah n bilangan asli menggunakan rekursi. Pernyataan ini menjelaskan bahwa kita perlu merumuskan fungsi yang akan menghitung penjumlahan semua bilangan asli dalam rentang 1 hingga n. Oleh karena itu, secara matematis Anda dapat merepresentasikan fungsi tersebut sebagai:

$F(n) = 1+2+3+4+.....+ (n-2) + (n-1) + n$

Hal ini dapat disederhanakan lebih lanjut sebagai berikut:

$$F(n) = \sum_{k=1}^{k=n} (k)$$

B. Agoritma Sorting

Menurut Daisma Bali Sorting adalah proses pengurutan data yang sebelumnya disusun secara acak sehingga tersusun secara teratur menurut aturan tertentu. Pengurutan dapat dilakukan secara ascending (urut naik) dan descending (urut turun).

Contoh:

Sebagai Contoh Sebuah Kamus. Jika Anda ingin mencari terjemahan pada kamus pasti melihat urutan alphabet bukan? Secara tidak langsung itulah manfaat sorting. Jadi akan lebih mudah mencari kata dalam kamus ketika abjad diurutkan sesuai urutan alfabet.
Algoritma Sorting berdasarkan perbandingan meliputi sebagai berikut :

Bubble Sort ( Pengurutan Gelembung )

Bubble sort (metode gelembung) adalah metode/algoritma pengurutan dengan dengan cara melakukan penukaran data dengan data sebelahnya secara terus menerus sampai bisa dipastikan dalam satu iterasi tertentu tidak ada lagi perubahan (data sudah terurut secara benar).
Selection Sort

Selection Sort adalah algoritma pengurutan data dengan cara membandingkan elemen yang sekarang dengan elemen yang berikutnya sampai dengan elemen yang terakhir. Jika ditemukan elemen lain yang lebih kecil dari elemen sekarang maka dicatat posisinya dan kemudian ditukar.
Insertion Sort

Insertion Sort adalah Sebuah algoritma pengurutan yang membandingkan dua elemen data pertama, mengurutkannya, kemudian mengecek elemen data berikutnya satu persatu dan membandingkannya dengan elemen data yang telah diurutkan.

Quick Sort

Quick Sort adalah salah satu algoritma pengurutan data yang paling cepat, yaitu dengan membagi list menggunakan sebuah pivot. pemilihan pivot adalah hal yang menentukan apakah algoritma quick sort tersebut akan memberikan performa terbaik atau terburuk.

C. Algoritma Searching

Menurut Annisa Algoritma Searching adalah serangkain langkah atau instruksi yang digunakan untuk mencari elemen atau informasi twrtentu di dalam suatu dataset. Tujuannya adalah untuk menemukan posisi atau keberadaan elemen yang dicari. Dalam pemrograman, algoritma search menjadi salah satu teknik penting dalam menyelesaikan berbagai masalah.
Jenis-jenisnya:
Linear Search, atau pencarian linear, adalah algoritma pencarian yang paling sederhana. Ini bekerja dengan cara memeriksa setiap elemen dalam kumpulan data secara berurutan hingga elemen yang dicari ditemukan atau sampai seluruh kumpulan data telah diperiksa.
Depth First Search (DFS) adalah algoritma pencarian yang digunakan dalam struktur data seperti graf. Ia mengikuti jalur sejauh mungkin sebelum kembali dan mengeksplorasi cabang lain.
Pencarian Binary adalah algoritma pencarian yang efisien digunakan untuk mencari elemen dalam kumpulan data yang sudah terurut. Algoritma ini membagi data menjadi dua bagian dan membandingkan elemen tengah dengan elemen yang dicari.

Pencarian Jump adalah algoritma pencarian yang efisien digunakan untuk mencari elemen dalam kumpulan data yang sudah terurut. Ia melompati beberapa langkah sekaligus dalam pencarian, memungkinkan efisiensi yang lebih baik daripada pencarian linear.
Contoh:
Sistem Rekomendasi
Algoritma pencarian juga digunakan dalam sistem rekomendasi seperti Netflix atau Spotify. Mereka mencari konten atau musik yang sesuai dengan sejarah penelusuran Anda atau preferensi yang telah di tentukan.

D. Algoritma Greedy

Menurut Annisa Algoritma Greedy adalah pendekatan dalam pemrograman yang memecahkan persoalan optimasi dengan cara yang tampaknya rakus. Pendekatan ini berfokus pada pengambilan keputusan sekarang dengan harapan bahwa setiap langkah akan membawa kita lebih dekat ke solusi akhir yang optimal.

Dalam konteks greedy, kita selalu memilih opsi yang paling menguntungkan saat ini tanpa mempertimbangkan konsekuensi di masa depan. Ini mirip dengan mengambil sejumlah uang tunai yang tersedia dari mesin ATM tanpa memikirkan bagaimana pengeluaran itu akan memengaruhi saldo akhir . Kegunaan utama dari algoritma greedy adalah untuk menemukan solusi optimal dalam persoalan optimasi dengan cepat. Pendekatan ini sangat berguna dalam banyak kasus di mana kita perlu memaksimalkan atau meminimalkan sesuatu dengan cara yang efisien. Contoh penerapannya termasuk perencanaan jadwal, pengkodean data, manajemen sumber daya, dan banyak lagi.

1. Huffman Coding

   Digunakan dalam kompresi data. Ini adalah metode yang efisien untuk mengurangi ukuran data dengan mengassign kode biner yang lebih pendek untuk karakter yang lebih sering muncul dalam teks.

2. Activity Selection

   Algoritma ini digunakan dalam manajemen waktu dan penjadwalan. Ketika Anda memiliki sejumlah kegiatan dengan waktu mulai dan selesai yang berbeda, algoritma ini membantu Anda memilih sejumlah kegiatan yang tidak saling tumpang tindih untuk mendapatkan jadwal yang paling efisien.

3. Kruskal Algorithm

   Digunakan dalam masalah pohon minimal (Minimum Spanning Tree) pada grafik. Ini membantu menemukan subset dari semua edge dalam grafik yang membentuk pohon tanpa siklus dengan total bobot yang minimal.

4. Prim’s Algorithm

   Seperti Kruskal, Prim’s Algorithm juga digunakan dalam masalah pohon minimal, tetapi fokus pada membangun pohon dari satu simpul awal dengan memilih edge terkecil yang terhubung.

Contoh:
Berikut adalah contoh sederhana implementasi algoritma greedy dalam bahasa Python untuk menyelesaikan masalah Fractional Knapsack:

```
def fractional_knapsack(items, capacity):
```

```
    items.sort(key=lambda x: x[1] / x[0], reverse=True)
    knapsack = []
    total_value = 0

    for item in items:
        if capacity >= item[0]:
            knapsack.append(item)
            total_value += item[1]
            capacity -= item[0]
        else:
            fraction = capacity / item[0]
            knapsack.append((item[0] * fraction, item[1] * fraction))
            total_value += item[1] * fraction
            break

    return knapsack, total_value

# Contoh penggunaan
items = [(2, 10), (3, 5), (5, 15), (7, 7), (1, 6)]
capacity = 15
result, total = fractional_knapsack(items, capacity)
print("Barang yang dipilih:", result)
print("Total nilai yang diperoleh:", total)
return knapsack, total_value

# Contoh penggunaan
items = [(2, 10), (3, 5), (5, 15), (7, 7), (1, 6)]
capacity = 15
result, total = fractional_knapsack(items, capacity)
print("Barang yang dipilih:", result)
print("Total nilai yang diperoleh:", total)
```

E. Algoritma baektracking

Menurut Kantinit Algoritma Backtracking adalah sebuah pendekatan dalam pemrograman yang bertujuan untuk mencari solusi dari permasalahan dengan melakukan pencarian secara sistematis pada semua kemungkinan yang ada. Pencarian solusi dilakukan dengan melakukan langkah mundur (backtracking) dari satu langkah ke langkah sebelumnya jika tidak ditemukan solusi yang memadai. Algoritma ini sering digunakan dalam permasalahan yang membutuhkan pemilihan dari sejumlah opsi yang tersedia. Algoritma ini dapat digunakan untuk menyelesaikan game sudoku dengan mencoba angka-angka yang mungkin di setiap kotak. Jika angka yang dicoba tidak memenuhi kriteria, maka algoritma melakukan backtracking ke langkah sebelumnya dan mencoba angka lain. Adapun prinsip kerjanya adalah melibatkan eksplorasi secara sistematis melalui semua kemungkinan solusi yang mungkin. Pada setiap langkah, algoritma akan memilih satu opsi dari himpunan opsi yang tersedia, dan kemudian melakukan pemanggilan rekursif untuk langkah berikutnya. Jika langkah tersebut mengarah ke solusi yang valid, algoritma akan

mencatat solusi tersebut. Namun, jika langkah tersebut mengarah ke jalan buntu, algoritma akan mundur ke langkah sebelumnya dan mencoba opsi lain yang belum dijelajahi.

Dengan menggunakan pendekatan ini, algoritma ini secara bertahap membangun solusi langkah demi langkah, memeriksa setiap kemungkinan sebelum membuat keputusan. Hal ini memungkinkan algoritma untuk menemukan solusi yang optimal atau menghasilkan semua solusi yang mungkin.

Contoh:

Salah satu contoh penggunaan algoritma backtracking adalah dalam menyelesaikan game sudoku. Dalam game ini, terdapat sejumlah kotak yang harus diisi dengan angka dari 1 hingga 9. Namun, setiap angka harus berbeda pada setiap baris, kolom, dan kotak kecil yang sudah ditentukan.

F. Algoritma Randomized

Menurut faradlla A, Algoritma randomized adalah algoritma yang menggunakan angka acak untuk menentukan langkah selanjutnya dalam proses pemecahan masalah. Algoritma ini juga dikenal sebagai algoritma acak. Algoritma acak bekerja dengan menggunakan mekanisme pemilihan acak, seperti pengambilan sampel acak, iterasi acak, atau operasi acak lainnya. Algoritma ini dapat digunakan untuk mengurangi waktu berjalan, kompleksitas waktu, memori yang digunakan, atau kompleksitas ruang dalam algoritme standar. Dalam praktiknya, algoritma acak sering menggunakan generator bilangan pseudo acak sebagai pengganti sumber bit acak yang sebenarnya. Implementasi seperti ini dapat menyimpang dari perilaku teoritis dan jaminan matematis yang diharapkan.