Nam: Paldiansyah

Nim: D0424503

Prodi: Sistem informasi

→ Apa yang di maksud dengan algoritma pemrograman.. ?

• Algoritma pemrograman adalah serangkaian langkah langkah logos yang di rancang untuk mencegah masalah tertentu dalam cara sistematis.

→ Cara kerja algoritma pemrograman yaitu:

• Input= Adalah tahap memasukkan data.

• Proses= adalah data yang sudah di masukkan kemudian di proses ke sistem.

• Output= Adalah data yang selesai di proses oleh sistem.

→ Jenis jenis algoritma yaitu:

1. Algoritma Brute Force

Algoritma brute force adalah pendekatan yang sederhana dan langsung untuk menyelesaikan masalah. Algoritma ini mencoba semua kemungkinan solusi hingga menemukan solusi yang benar. Meskipun efektif untuk masalah sederhana, metode ini kurang efisien untuk masalah yang lebih kompleks karena membutuhkan waktu yang lama.

Contoh: Mencari pasangan bilangan dalam array yang jika dijumlahkan sama dengan nilai tertentu. Algoritma brute force akan mencoba menjumlahkan setiap pasangan bilangan hingga menemukan hasil yang diinginkan.

def brute\_force\_sum(arr, target):

for i in range(len(arr)):

for j in range(i + 1, len(arr)):

if arr[i] + arr[j] == target:

return (arr[i], arr[j])

return None

2. Algoritma Divide and Conquer

Algoritma divide and conquer membagi masalah menjadi sub-masalah yang lebih kecil, menyelesaikan sub-masalah tersebut secara rekursif, dan menggabungkan solusi untuk mendapatkan solusi akhir. Metode ini sangat efisien dalam memecahkan masalah yang kompleks dengan cara memecahnya menjadi bagian yang lebih mudah.

Contoh: Algoritma Merge Sort adalah salah satu implementasi dari divide and conquer, yang memecah array menjadi dua bagian, mengurutkannya secara rekursif, lalu menggabungkannya kembali.

def merge\_sort(arr):

if len(arr) <= 1:

return arr

mid = len(arr) // 2

left = merge\_sort(arr[:mid])

right = merge\_sort(arr[mid:])

return merge(left, right)

def merge(left, right):

sorted\_arr = []

while left and right:

if left[0] <= right[0]:

sorted\_arr.append(left.pop(0))

else:

sorted\_arr.append(right.pop(0))

return sorted\_arr + left + right

3. Algoritma Greedy

Algoritma greedy bekerja dengan memilih pilihan terbaik yang tersedia pada setiap langkah tanpa memikirkan konsekuensi jangka panjang. Pendekatan ini mungkin tidak selalu menghasilkan solusi optimal, tetapi sering digunakan karena cepat dan mudah diimplementasikan.

Contoh: Algoritma Huffman Coding adalah algoritma greedy yang digunakan untuk kompresi data, di mana karakter yang paling sering muncul diberi kode pendek.

import heapq

from collections import defaultdict

def huffman\_encoding(freq):

heap = [[weight, [char, ""]] for char, weight in freq.items()]

heapq.heapify(heap)

while len(heap) > 1:

lo = heapq.heappop(heap)

hi = heapq.heappop(heap)

for pair in lo[1:]:

pair[1] = '0' + pair[1]

for pair in hi[1:]:

pair[1] = '1' + pair[1]

heapq.heappush(heap, [lo[0] + hi[0]] + lo[1:] + hi[1:])

return sorted(heapq.heappop(heap)[1:], key=lambda p: (len(p[-1]), p))

freq = {'a': 5, 'b': 9, 'c': 12, 'd': 13, 'e': 16, 'f': 45}

huffman\_encoding(freq)

4. Algoritma Dynamic Programming

Algoritma dynamic programming (pemrograman dinamis) digunakan untuk memecahkan masalah kompleks dengan memecahnya menjadi sub-masalah yang tumpang tindih dan menyelesaikannya secara efisien dengan menyimpan hasil sub-masalah untuk digunakan kembali.

Contoh: Masalah Fibonacci sering diselesaikan menggunakan dynamic programming untuk menghindari perhitungan berulang.

def fibonacci(n, memo={}):

if n in memo:

return memo[n]

if n <= 2:

return 1

memo[n] = fibonacci(n - 1, memo) + fibonacci(n - 2, memo)

return memo[n]

print(fibonacci(10)) # Output: 55

5. 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. [1][2][5]

- Kasus dasar: Algoritma rekursif memiliki "kasus dasar" yang merupakan kondisi berhenti, di mana solusi dapat langsung dihitung tanpa perlu pemanggilan rekursif lagi. [1][5]

- Solusi rekursif: Setelah kasus dasar tercapai, solusi sub-masalah digabungkan untuk menghasilkan solusi untuk masalah aslinya. [1][5]

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

Salin

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.