LAPORAN DOKUMENTASI TUGAS GREEDY

COIN CHANGE PROBLEM

DESAIN ANALISIS & ALGORITMA

DISUSUN OLEH:

L0123078 – M. IRFAN DHIA ULHAQ

L0123094 – MUHAMMAD FEBRIAN JAMALUDIN

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA

UNIVERSITAS SEBELAS MARET

SURAKARTA

2024

PENJABARAN PROBLEM

Coin Change Problem berguna untuk menyelesaikan sebuah problem penukaran uang. Problem tersebut adalah bagaimana cara menukar sejumlah uang tertentu dengan koin-koin yang tersedia, dengan tujuan menggunakan jumlah koin yang seminimal mungkin.

Problem ini melibatkan dua input utama:
1. Serangkaian nilai koin yang tersedia (misalnya: 100, 200, 500, 1000)
2. Jumlah uang yang ingin ditukar

Program akan mencoba menukar jumlah uang tersebut menggunakan koin-koin yang tersedia, dengan strategi greedy yaitu selalu menggunakan koin dengan nilai terbesar yang mungkin terlebih dahulu.

Problem ini menimbulkan 2 kemungkinan dalam bagaimana problem tersebut diselesaikan:
1. Semua uang dapat ditukar dengan koin-koin yang tersedia.
2. Sebagian uang tidak dapat ditukar karena tidak ada kombinasi koin yang tepat.

Ketika program berhasil menukar semua uang, maka output akan berupa:
- Jumlah dan nilai setiap koin yang digunakan
- Total jumlah koin yang digunakan
- Jika ada sisa uang yang tidak dapat ditukar, program akan memberikan peringatan yang menyatakan jumlah sisa yang tidak dapat ditukar.

Program ini menggunakan pendekatan greedy, yang meskipun efisien, tidak selalu menjamin solusi optimal untuk semua kasus. Namun, untuk sistem mata uang yang umum digunakan, pendekatan ini biasanya memberikan hasil yang optimal.

DEMONSTRASI PROGRAM

1. Jalankan program greedy.py menggunakan Python.
2. Saat diminta, masukkan nilai koin yang tersedia. Contoh: ketik "100 200 500 1000" (tanpa tanda kutip) dan tekan Enter. Anda akan melihat prompt: "Masukkan nilai koin (pisahkan dengan spasi, contoh: 100 200 500 1000)"
3. Selanjutnya, program akan meminta jumlah uang yang ingin ditukar.
   Contoh: ketik "2350" dan tekan Enter.
   Anda akan melihat prompt "Masukkan jumlah uang: "
4. Program akan menjalankan algoritma greedy dan menampilkan hasilnya.
5. Jika ada sisa uang yang tidak dapat ditukar, program akan menampilkan peringatan.

PENJELASAN IMPLEMENTASI

No.	Bagian dan Perannya
1.	```def coin_change_greedy(X, arr): arr.sort(reverse=True) result = {} for coin in arr: if X >= coin: count = X // coin X = X % coin result[coin] = count return result``` Bagian ini merupakan implementasi algoritma Greedy Coin Change. Algoritma ini bertujuan untuk menukar sejumlah uang tertentu (X) menggunakan koin-koin dengan nilai tertentu (arr) dengan jumlah koin seminimal mungkin. Berikut adalah penjelasan dari setiap bagiannya: • Mendefinisikan fungsi dengan nama `coin_change_greedy`. • Menerima dua parameter: `X` (jumlah uang yang akan ditukar) dan `arr` (array nilai koin yang tersedia). • Mengurutkan array koin dari nilai terbesar ke terkecil. • Membuat dictionary kosong untuk menyimpan hasil (jumlah koin untuk setiap nilai). • Iterasi melalui setiap nilai koin dalam array yang sudah diurutkan. • Memeriksa apakah nilai koin saat ini dapat digunakan. • Menghitung berapa banyak koin tersebut yang bisa digunakan (`count`). • Mengurangi jumlah uang yang tersisa (`X`) dengan jumlah yang sudah ditukar. • Menyimpan jumlah koin yang digunakan dalam `result`. • Mengembalikan dictionary yang berisi solusi coin change.
2.	```print("Masukkan nilai koin (pisahkan dengan spasi, contoh: 100 200 500 1000)") while True: try: arr = list(map(int, input().split())) if len(arr) == 0: print("Masukkan setidaknya satu nilai koin.") elif any(coin <= 0 for coin in arr):```

```
            print("Semua nilai koin harus positif. Silakan coba lagi.")
        else:
            break
    except ValueError:
        print("Input tidak valid. Masukkan angka bulat positif, pisahkan dengan
spasi.")
```

Ini adalah implementasi input validation untuk meminta dan memvalidasi input nilai koin dari pengguna. Berikut adalah peran-perannya:

- Meminta dan memvalidasi input nilai koin dari pengguna.
- Memastikan bahwa input adalah angka bulat positif, dipisahkan dengan spasi.
- Memastikan bahwa setidaknya ada satu nilai koin yang dimasukkan.
- Memberikan feedback jika input tidak valid (misalnya: ada nilai negatif, bukan angka bulat, atau tidak ada koin yang dimasukkan).

3.

```
while True:
    try:
        X = int(input("Masukkan jumlah uang: "))
        if X < 0:
            print("Jumlah uang harus positif. Silakan coba lagi.")
        else:
            break
    except ValueError:
        print("Input tidak valid. Masukkan angka bulat positif.")
```

Bagian ini adalah bagian validasi input untuk jumlah uang. Berikut adalah peran-perannya:

- Meminta dan memvalidasi input jumlah uang dari pengguna.
- Memastikan input adalah angka bulat positif.
- Memberikan feedback jika input tidak valid.

4.

```
solution = coin_change_greedy(X, arr)
```

Ini adalah bagian pemanggilan fungsi utama. Perannya adalah memanggil fungsi coin_change_greedy dengan jumlah uang (X) dan array nilai koin (arr), kemudian menyimpan hasil dalam variabel solution.

5.	``` print("\nSolusi coin change:") for coin, count in solution.items(): print(f"{count} koin bernilai {coin}") ``` Bagian ini berfungsi untuk menampilkan solusi coin change dan menunjukkan jumlah koin untuk setiap nilai.
6.	``` total_coins = sum(solution.values()) print(f"\nTotal koin yang digunakan: {total_coins}") ``` Bagian ini berfungsi untuk menghitung total jumlah koin yang digunakan, kemudian menampilkan total koin kepada pengguna.
7.	``` remaining = X - sum(coin * count for coin, count in solution.items()) if remaining > 0: print(f"\nPeringatan: {remaining} tidak dapat ditukar karena tidak ada koin yang sesuai.") ``` Bagian ini adalah bagian untuk mengecek sisa uang yang tidak tertukar. Fungsi ini menghitung sisa uang yang tidak dapat ditukar, kemudian menampilkan peringatan jika ada sisa uang yang tidak dapat ditukar.

PENGUJIAN

```
Masukkan nilai koin (pisahkan dengan spasi, contoh: 100 200 500 1000)
100 200 500 1000
```

1. Pengguna diminta memasukkan nilai koin, dipisahkan oleh spasi. Pada contoh ini, pengguna memasukkan 100 200 500 1000.

```
Masukkan jumlah uang: 2350
```

2. Setelah itu, pengguna memasukkan jumlah uang yang akan ditukarkan dengan koin, pada kasus ini yaitu 2350.

```
Solusi coin change:
2 koin bernilai 1000
1 koin bernilai 200
1 koin bernilai 100

Total koin yang digunakan: 4

Peringatan: 50 tidak dapat ditukar karena tidak ada koin yang sesuai.
```

3. Program menghasilkan solusi dari proses coin change dengan rincian:
    - 2 koin bernilai 1000
    - 1 koin bernilai 200
    - 1 koin bernilai 100
4. Program menampilkan bahwa total koin yang digunakan untuk mencapai jumlah 2350 adalah 4 koin.
5. Program mengeluarkan peringatan bahwa 50 tidak dapat ditukarkan karena tidak ada koin yang sesuai dengan nilai tersebut di dalam set koin yang dimasukkan pengguna.

ANALISIS KOMPLEKSITAS

Fungsi coin_change_greedy(X, arr)

- arr.sort(reverse=True): Melakukan pengurutan nilai koin dalam urutan menurun. Kompleksitas pengurutan dengan algoritma seperti Timsort (yang digunakan di Python) adalah O(n log n), di mana n adalah panjang array arr (jumlah jenis koin).
- Iterasi untuk Menukar Koin: Setelah pengurutan, kita melakukan iterasi melalui seluruh array arr dalam loop for coin in arr:
- Pada setiap iterasi, program menghitung jumlah koin (count = X // coin) dan mengurangi nilai X (X = X % coin). Operasi ini berjalan dengan kompleksitas O(1) untuk setiap iterasi.
- Total iterasi untuk array arr adalah n, di mana n adalah jumlah jenis koin. Kompleksitas loop ini adalah O(n).
- Jadi, total kompleksitas fungsi coin_change_greedy adalah O(nlogn), yang didominasi oleh operasi pengurutan.

Total Kompleksitas Waktu

- Operasi yang paling mendominasi adalah pengurutan array arr dengan kompleksitas O(n log n).
- Bagian lainnya memiliki kompleksitas linear O(n) atau konstan O(1).
- Sehingga total kompleksitas waktu dari keseluruhan kode adalah: **O(n log n)**, dimana n adalah jumlah jenis koin yang dimasukkan pengguna.

Total Kompleksitas Ruang

- Ruang yang digunakan untuk array arr: O(n).
- Dalam kasus terburuk, semua jenis koin digunakan, sehingga ruang yang digunakan adalah O(n).
- Oleh karena itu, kompleksitas ruang juga O(n), dimana n adalah jumlah jenis koin.

APAKAH ALGORITMA GREEDY AKAN SELALU OPTIMAL?

Algoritma greedy sering kali digunakan dalam masalah-masalah optimasi karena kesederhanaannya. Pada dasarnya, algoritma ini bekerja dengan memilih solusi terbaik yang terlihat pada setiap langkah tanpa mempertimbangkan langkah-langkah selanjutnya. Untuk masalah *coin change*, algoritma greedy dapat memberikan solusi optimal jika nilai koin yang tersedia memiliki pola tertentu, seperti dalam mata uang standar (misalnya, koin bernilai 1, 5, 10, 25, 100). Dalam situasi ini, memilih koin terbesar yang mungkin pada setiap langkah memastikan bahwa kita menggunakan jumlah koin paling sedikit.

Namun, algoritma greedy tidak selalu memberikan solusi optimal. Dalam beberapa sistem koin yang tidak mengikuti pola tertentu, algoritma ini bisa gagal. Contohnya, jika koin yang tersedia adalah {1, 3, 4} dan kita ingin membuat 6 unit uang, algoritma greedy akan memilih 1 koin 4 dan 2 koin 1, padahal solusi optimal adalah 2 koin 3. Kegagalan ini terjadi karena algoritma greedy membuat keputusan berdasarkan solusi lokal yang tampak paling baik tanpa memikirkan efek jangka panjang dari setiap pilihan.

Karena algoritma greedy hanya fokus pada keputusan terbaik di setiap langkah, ia tidak selalu memberikan solusi yang optimal untuk semua masalah. Dalam kasus seperti masalah coin change dengan nilai koin yang tidak terstruktur secara efisien, pendekatan lain seperti *dynamic programming* mungkin diperlukan untuk menemukan solusi optimal. Oleh karena itu, penting untuk memahami karakteristik masalah sebelum memutuskan apakah algoritma greedy akan memberikan solusi yang tepat atau tidak.

PERAN ANGGOTA KELOMPOK

M. Irfan Dhia Ulhaq (L0123078)	Membuat kode, dan membuat video dokumentasi.
Muhammad Febrian Jamaludin (L0123094)	Membuat laporan dan membantu menyempurnakan kode.