

LAPORAN GREEDY

DESAIN & ANALISA ALGORITMA



DISUSUN OLEH :

Michael Johanes Johansyah L0123081

Muhammad Firman Ghani L0123095

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA

UNIVERSITAS SEBELAS MARET

2024

BAB I

PENJABARAN PROBLEM

Coin Change Problem merupakan salah satu masalah klasik dalam algoritma Greedy yang bertujuan untuk menemukan jumlah koin minimal yang diperlukan untuk mencapai jumlah tertentu dengan menggunakan sekumpulan denominasi koin yang telah diberikan. Diberikan sejumlah koin dengan berbagai nilai yaitu (1, 2, 5, 10) dan sebuah jumlah total uang yang harus dicapai (misalnya 18). Tugasnya adalah menentukan jumlah minimal koin yang harus digunakan untuk mencapai jumlah total tersebut.

Algoritma Greedy akan selalu mencoba untuk memilih nilai koin terbesar yang mungkin pada setiap langkah, dengan harapan bahwa pilihan ini akan menghasilkan solusi optimal. Namun, pendekatan ini tidak selalu optimal untuk semua set koin. Misalnya, jika nilai koin adalah $\{1, 3, 4\}$ dan kita perlu mencapai target nilai 6, pendekatan Greedy akan memilih koin 4 terlebih dahulu (karena itu yang terbesar), lalu koin 1 dua kali, menghasilkan total 3 koin. Padahal solusi optimal adalah memilih dua koin 3 (hanya 2 koin).

BAB II

PENJELASAN IMPLEMENTASI

No.	Implementasi	Penjelasan
1.	Mengimport streamlit	<pre>import streamlit as st</pre> <p>Baris ini mengimpor Streamlit, yang merupakan framework Python untuk membangun aplikasi web interaktif dengan cepat.</p>
2.	Fungsi coin_change	<pre>def coin_change(X, arr): arr.sort(reverse=True) result = [] count_dict = {coin: 0 for coin in arr} for coin in arr: while X >= coin: X -= coin result.append(coin) count_dict[coin] += 1 if X != 0: st.write("Tidak bisa mendapatkan jumlah yang tepat dengan koin yang tersedia.") return result, count_dict</pre> <p>Fungsi ini digunakan untuk menghitung kombinasi koin yang digunakan untuk mencapai jumlah target X dengan koin yang tersedia dalam arr.</p> <ol style="list-style-type: none">1. <code>arr.sort(reverse=True)</code>: Mengurutkan daftar koin arr secara menurun, sehingga program akan mencoba menggunakan koin dengan nilai terbesar terlebih dahulu (algoritma "greedy").2. <code>result = []</code> dan <code>count_dict</code>:<ul style="list-style-type: none">• <code>result</code> digunakan untuk menyimpan urutan koin yang digunakan untuk mencapai jumlah target X.• <code>count_dict</code> adalah kamus (dictionary) yang menyimpan jumlah setiap koin yang digunakan. Awalnya, setiap koin dalam arr diinisialisasi dengan nilai 0.3. Iterasi <code>for coin in arr</code>:: Mengiterasi setiap koin dalam daftar arr. Untuk setiap koin: <code>while X >= coin</code>:: Selama X lebih besar atau sama dengan nilai koin yang sedang diiterasi, nilai koin tersebut dikurangi dari X, dan koin

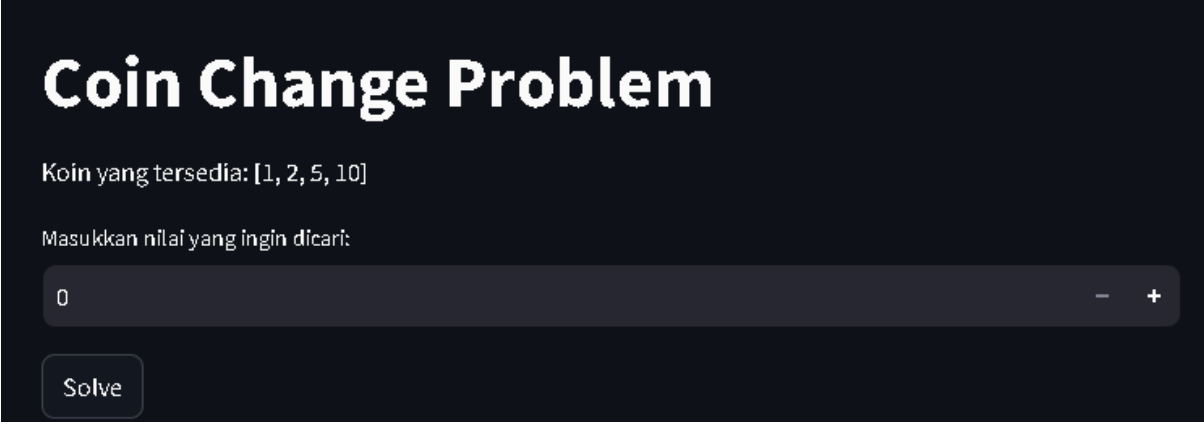
		<p>tersebut ditambahkan ke dalam result. count_dict[coin] += 1 menambah jumlah koin tersebut yang digunakan dalam kamus count_dict.</p> <ol style="list-style-type: none"> 4. if X != 0:: Jika nilai X tidak menjadi 0 setelah iterasi, maka tidak mungkin mencapai jumlah target dengan kombinasi koin yang tersedia. 5. Fungsi mengembalikan result, yang berisi urutan koin yang digunakan, dan count_dict, yang berisi jumlah setiap koin yang digunakan.
3.	Fungsi main	<pre>def main(): st.title('Coin Change Problem') arr = [1, 2, 5, 10] st.write(f"Koin yang tersedia: {arr}") X = st.number_input("Masukkan nilai yang ingin dicari:", min_value=0, step=1) if st.button('Solve'): result, count_dict = coin_change(X, arr) if result: st.write(f"Kombinasi koin untuk mencapai {X} adalah: {result}") st.write("Jumlah setiap koin yang digunakan:") for coin, count in count_dict.items(): if count > 0: st.write(f"Koin {coin}: {count} kali")</pre> <p>Fungsi ini mengatur antarmuka pengguna aplikasi menggunakan Streamlit.</p> <ol style="list-style-type: none"> 1. st.title('Coin Change Problem'): Menampilkan judul aplikasi pada halaman web. 2. arr = [1, 2, 5, 10]: Mendefinisikan daftar koin yang tersedia untuk digunakan. 3. st.write(f"Koin yang tersedia: {arr}"): Menampilkan koin yang tersedia. 4. X = st.number_input("Masukkan nilai yang ingin dicari:", min_value=0, step=1): <ul style="list-style-type: none"> • Menyediakan input untuk pengguna memasukkan jumlah target X yang ingin dicari.

		<ul style="list-style-type: none"> • Nilai X harus berupa bilangan bulat minimal 0 dan bertambah dengan langkah 1. <p>5. if st.button('Solve'):: Saat pengguna menekan tombol "Solve", program akan menjalankan fungsi coin_change(X, arr) untuk mencari kombinasi koin.</p> <p>6. Menampilkan Hasil:</p> <ul style="list-style-type: none"> • Jika kombinasi koin ditemukan, hasilnya ditampilkan dengan menggunakan st.write(). • Hasil yang ditampilkan adalah kombinasi koin dalam result dan jumlah setiap koin yang digunakan dalam count_dict.
4.	Memanggil fungsi main	<pre>if __name__ == "__main__": main()</pre> <p>Bagian ini memastikan bahwa fungsi main() hanya akan dipanggil jika file ini dieksekusi sebagai skrip utama, bukan jika diimport sebagai modul ke skrip lain.</p>

BAB III

PENGUJIAN

1. Tampilan Web



Coin Change Problem

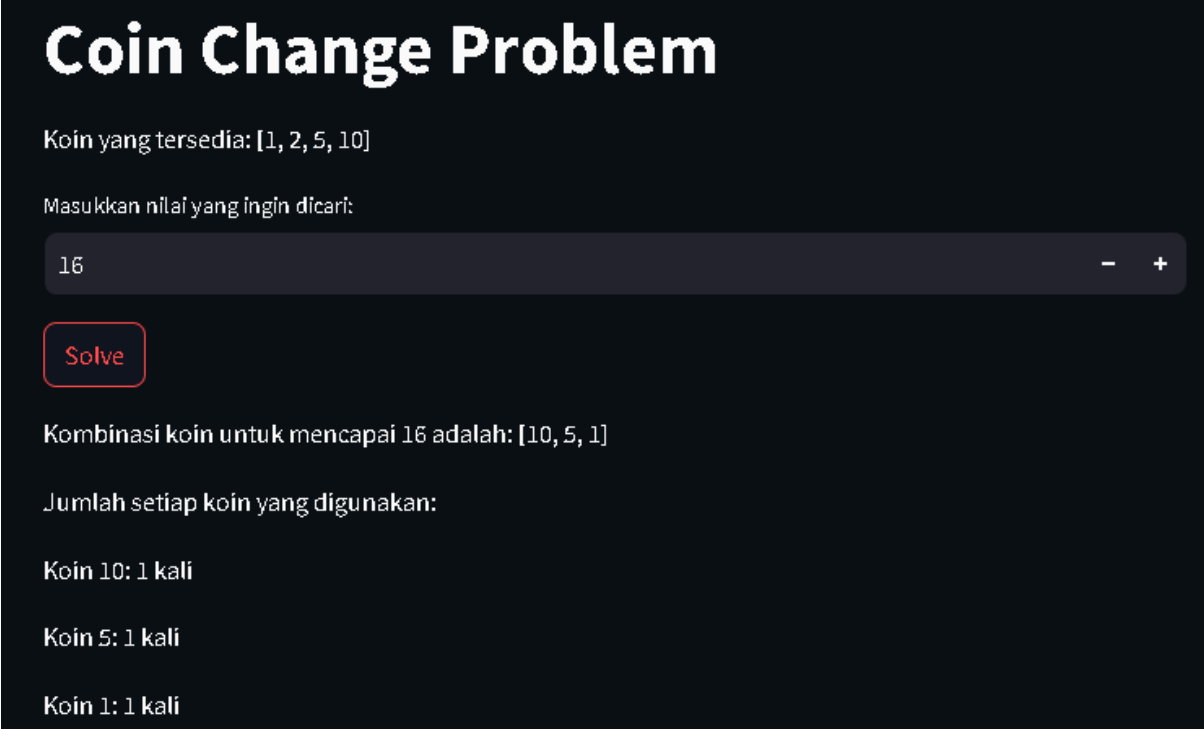
Koin yang tersedia: [1, 2, 5, 10]

Masukkan nilai yang ingin dicari:

- +

Solve

2. Tampilan ketika program menemukan solusi



Coin Change Problem

Koin yang tersedia: [1, 2, 5, 10]

Masukkan nilai yang ingin dicari:

- +

Solve

Kombinasi koin untuk mencapai 16 adalah: [10, 5, 1]

Jumlah setiap koin yang digunakan:

Koin 10: 1 kali

Koin 5: 1 kali

Koin 1: 1 kali

BAB IV

PEMBAGIAN TUGAS

Kami mengerjakan tugas 20 Solver ini dengan rincian pembagian tugas sebagai berikut:

1. Michael Johanes Johansyah
 - Membuat video dokumentasi.
2. Muhammad Firman Ghani
 - Membuat sourcecode menjadi web
 - Membuat laporan