LAPORAN PROGRAM COIN CHANGE
MENGGUNAKAN ALGORITMA GREEDY
DESAIN ANALISIS DAN ALGORITMA

DISUSUN OLEH:

Mohammad Nazhiif Al-Ghoniy	L0123084
Muh Taqiyudin Ibadurrahman	L0123086

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA

UNIVERSITAS SEBELAS MARET

2024

BAB I
PENJABARAN PROBLEM

1.1 Latar Belakang

Masalah penukaran koin (coin change problem) merupakan salah satu permasalahan klasik dalam teori algoritma dan optimasi. Masalah ini melibatkan pencarian kombinasi sejumlah koin untuk mencapai jumlah nominal tertentu. Masalah ini sering ditemukan dalam kehidupan sehari-hari, misalnya saat seseorang ingin menukarkan sejumlah uang dalam bentuk koin yang tersedia.

Pada tugas ini, kita dihadapkan pada situasi di mana user menginputkan nominal uang yang ingin ditukarkan, dan koin yang tersedia adalah {1, 2, 5, 10}. Untuk menyelesaikan masalah ini, salah satu pendekatan yang sering digunakan adalah algoritma greedy. Algoritma greedy bekerja dengan memilih koin terbesar yang bisa digunakan untuk nominal yang tersisa pada setiap langkah. Meskipun demikian, algoritma greedy tidak selalu memberikan solusi yang optimal untuk semua jenis masalah penukaran koin, tergantung pada denominasi yang digunakan.

1.2 Rumusan Masalah

Rumusan masalah dari tugas ini adalah sebagai berikut:

1. Bagaimana cara menentukan kombinasi koin yang tersedia (yaitu 1, 2, 5, dan 10) untuk menukar sejumlah uang yang diinputkan oleh pengguna?
2. Bagaimana cara menyelesaikan masalah penukaran koin ini menggunakan algoritma greedy?
3. Bagaimana program dapat menampilkan jumlah koin dan jenis koin yang digunakan untuk mencapai nominal yang diinginkan?

1.3 Tujuan

Adapun tujuan dari tugas ini adalah:

1. Mengimplementasikan algoritma penukaran koin menggunakan pendekatan greedy.
2. Mengembangkan program yang mampu menerima input dari pengguna berupa nominal uang yang ingin ditukarkan.
3. Menampilkan solusi berupa kombinasi koin yang digunakan serta jumlahnya untuk mencapai nominal yang diinputkan.
4. Melatih pemahaman konsep algoritma greedy dan batasannya dalam menyelesaikan masalah optimasi sederhana.

1.4 Ruang Lingkup

Tugas ini terbatas pada permasalahan penukaran koin dengan nilai-nilai koin yang telah ditentukan, yaitu 1, 2, 5, dan 10. Program yang dikembangkan hanya akan menerima input berupa nominal bilangan bulat positif, dan solusi yang ditampilkan akan berupa kombinasi koin untuk mencapai jumlah yang diinginkan. Algoritma greedy digunakan dalam penyelesaian masalah ini, tetapi tidak selalu dapat dijamin memberikan solusi optimal untuk semua jenis denominasi koin. Dalam hal ini, program akan menyelesaikan masalah menggunakan pendekatan greedy sesuai dengan koin yang tersedia.

1.5 Manfaat

Manfaat yang diharapkan dari tugas ini adalah:

1. Meningkatkan pemahaman mahasiswa tentang penerapan algoritma greedy dalam memecahkan permasalahan optimasi.
2. Memperkuat keterampilan pemrograman dalam menyusun algoritma yang efisien.
3. Memberikan wawasan tentang cara menyelesaikan masalah nyata dengan pendekatan komputasional, serta memahami batasan algoritma greedy dalam konteks tertentu.
4. Mengembangkan keterampilan analisis untuk mengevaluasi solusi yang dihasilkan, termasuk mempertimbangkan kemungkinan adanya solusi yang lebih optimal di luar pendekatan greedy.

BAB II
IMPLEMENTASI PROGRAM

2.1 Pendahuluan

Pada bagian ini, akan dibahas secara komprehensif tentang implementasi program penukaran koin **coin change problem**. Program ini menggunakan pendekatan algoritma **greedy** untuk menyelesaikan permasalahan dengan memberikan pecahan koin minimal yang dibutuhkan untuk menukarkan sejumlah uang yang diinputkan oleh pengguna. Program juga menggunakan pustaka Colorama untuk memberikan efek warna pada teks yang ditampilkan di terminal, sehingga output lebih interaktif dan mudah dipahami.

2.2 Struktur Kode

Berikut adalah analisis dari implementasi program yang diberikan:

1. Inisialisasi Colorama:

```
from colorama import Fore, Style, init
```

Program ini memanfaatkan pustaka Colorama untuk mewarnai output yang ditampilkan di terminal. Pustaka ini diinisialisasi dengan init(autoreset=True), yang berarti setelah setiap pemakaian warna, format akan direset secara otomatis. Ini membuat output lebih menarik dan lebih mudah dibaca oleh pengguna.

2. Fungsi minimumCoin:

```
def minimumCoin(uangTotal, pecahan):
    hasil = []

    # Melakukan iterasi melalui setiap pecahan uang
    i = len(pecahan) - 1
    while (i >= 0):
        while (uangTotal >= pecahan[i]):
            uangTotal -= pecahan[i]
            hasil.append(pecahan[i])

        i -= 1
```

Fungsi minimumCoin bertanggung jawab untuk menghitung kombinasi minimal dari pecahan koin yang dibutuhkan untuk menukarkan uang. Input dari fungsi ini adalah uangTotal (nominal uang yang diinginkan) dan pecahan (list pecahan koin yang tersedia).

- List hasil: Berfungsi untuk menyimpan pecahan koin yang digunakan dalam penukaran.

- Variabel i: Inisialisasi variabel i dengan indeks terakhir dari list pecahan, yang akan memulai iterasi dari pecahan terbesar.
- Looping while pertama: Melakukan iterasi dari pecahan terbesar ke pecahan terkecil. Pada setiap iterasi, program mengecek apakah pecahan koin yang sedang diperiksa bisa digunakan untuk mengurangi nominal uangTotal. Jika ya, maka uangTotal dikurangi oleh pecahan tersebut, dan pecahan tersebut ditambahkan ke list hasil.
- Looping while kedua: Jika nominal koin terbesar bisa mengurangi uangTotal, maka loop ini akan terus mengurangi nominal uangTotal dengan pecahan koin tersebut hingga tidak bisa lagi, dan baru kemudian berlanjut ke pecahan yang lebih kecil.

Setelah semua iterasi selesai, list hasil akan berisi pecahan-pecahan koin yang digunakan untuk mencapai nominal uangTotal.

3. Menampilkan hasil:

```
  # Cetak hasil
   print(f"\n{Fore.YELLOW}Pecahan yang digunakan:")
   for i in range(len(hasil)):
       print(f"{Fore.CYAN}{hasil[i]}", end=" ")

         print(f"\n\n{Fore.GREEN}Total   pecahan   yang   digunakan:
{len(hasil)}")
```

- Setelah koin-koin yang digunakan untuk mencapai nominal uangTotal ditentukan, program mencetak pecahan yang digunakan dengan efek warna menggunakan Colorama.
- Pada bagian akhir, jumlah total koin yang digunakan ditampilkan untuk memberikan informasi lebih lengkap kepada pengguna.

4. Bagian main:

```
if __name__ == '__main__':
   print(f"{Fore.MAGENTA}{Style.BRIGHT}--- Tukar Uang ---")
   print(f"{Fore.MAGENTA}{'-' * 30}")

   total = int(input(f"{Fore.CYAN}Input uang yang ingin ditukar: "))

   print(f"\n{Fore.CYAN}Input pecahan uang yang tersedia")
   pecahan_input = input(f"{Fore.CYAN}(pisahkan dengan spasi): ")

   # Mengubah input pecahan menjadi list of integers
   pecahan = list(map(int, pecahan_input.split()))
```

```
   pecahan.sort()  # Mengurutkan pecahan dari kecil ke besar
```

Bagian ini adalah fungsi utama yang mengendalikan alur program:

- Input nominal uang (total): Program menerima input berupa jumlah uang yang ingin ditukarkan oleh pengguna.
- Input pecahan koin (pecahan_input): Pengguna diminta untuk menginputkan pecahan koin yang tersedia, dipisahkan oleh spasi. Input ini kemudian diubah menjadi list integer dengan list(map(int, pecahan_input.split())).
- Pengurutan Pecahan: Pecahan diurutkan menggunakan pecahan.sort() agar program dapat bekerja dengan baik dalam menentukan kombinasi pecahan koin mulai dari yang terbesar.

5. Menampilkan hasil akhir:

```
# Menampilkan jumlah minimal uang pecahan
   print(f"{Fore.MAGENTA}\n{'-' * 30}")
      print(f"{Fore.YELLOW}Jumlah minimal pecahan untuk {total}: ",
end="")
   minimumCoin(total, pecahan)
```

Setelah input pengguna diproses dan diurutkan, program memanggil fungsi minimumCoin untuk menghitung dan menampilkan jumlah minimal koin yang dibutuhkan.

2.3 Algoritma Greedy yang Digunakan

Program ini menggunakan pendekatan algoritma greedy, yang memilih koin dengan nilai terbesar yang bisa mengurangi uangTotal pada setiap langkah. Algoritma ini terus mengurangi uangTotal dengan pecahan terbesar hingga tidak memungkinkan lagi, dan kemudian beralih ke pecahan yang lebih kecil. Kelebihan dari algoritma greedy adalah:

- Sederhana dan efisien: Algoritma ini mudah dipahami dan memiliki kompleksitas waktu yang rendah, yaitu O(n), di mana n adalah jumlah pecahan yang tersedia.
- Dapat memberikan solusi optimal pada kondisi tertentu: Untuk pecahan yang umum seperti {1, 2, 5, 10}, algoritma greedy biasanya memberikan solusi optimal.

Namun, kelemahan dari algoritma ini adalah:

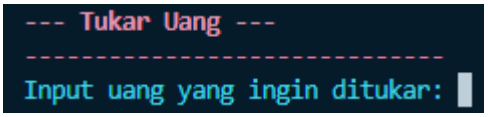
- Tidak selalu optimal: Pada jenis pecahan tertentu, algoritma greedy mungkin tidak memberikan solusi terbaik. Misalnya, jika pecahan yang tersedia adalah {1, 3, 4}, algoritma ini tidak akan selalu menghasilkan solusi minimal.

BAB III
PENGUJIAN PROGRAM DAN KESIMPULAN

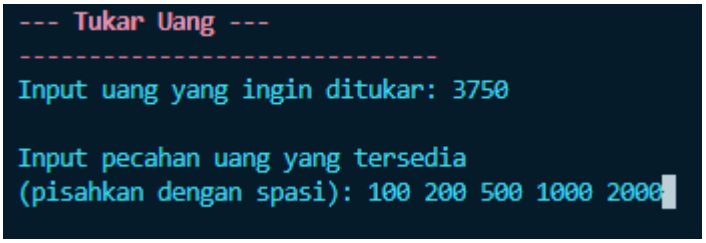
3.1 Pengujian Program

Pengujian program "Coin Change" yang menggunakan modul colorama untuk menampilkan hasil dalam warna-warna tertentu. Program ini dirancang untuk menghitung jumlah minimal pecahan uang yang diperlukan untuk mencapai total uang yang diinputkan.

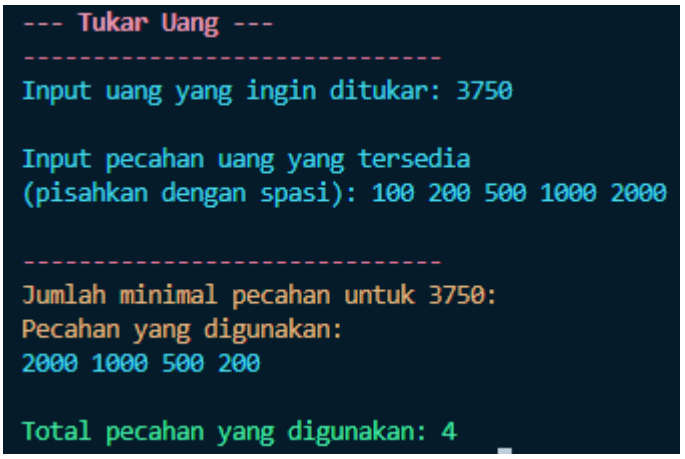
1. Inputkan jumlah uang yang ingin ditukar, misalnya: 3750



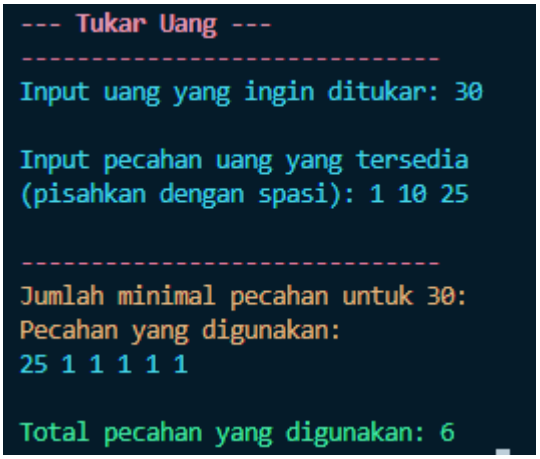
2. Inputkan pecahan uang yang tersedia, misalnya: 100 200 500 1000 2000.



3. Program menampilkan jumlah minimal (secara algoritme *greedy*) pecahan yang digunakan untuk mencapai jumlah total.



4. Kasus kode tidak optimum pada saat jumlah uang 30 dan pecahannya 1 10 25



3.2 Kesimpulan

Implementasi program penukaran koin ini berhasil menggunakan pendekatan algoritma greedy untuk menemukan kombinasi minimal pecahan koin. Meskipun algoritma greedy sederhana dan efisien, dalam kasus tertentu solusinya mungkin tidak optimal tergantung pada konfigurasi pecahan koin yang diberikan. Program ini juga menggunakan Colorama untuk memperkaya tampilan hasil yang lebih interaktif dan informatif bagi pengguna.

3.3 Pembagian Tugas

Mohammad Nazhiif Al-Ghoniy L0123084

- Pembuatan source + readme
- Pembuatan video demo

Muh Taqiyudin Ibadurrahman L0123086

- Pembuatan laporan dokumentasi
- Pembuatan source code