LAPORAN DESAIN ANALISIS DAN ALGORITMA
COIN CHANGE PROBLEM WITH GREEDY ALGORITHM

DOSEN PENGAJAR:

Fajar Muslim S.T., M.T.

DI SUSUN OLEH:

Moh Ferdinand Ramdhani	L0123082
Muhammad Al Fathi Ayyash	L0123088

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA

UNIVERSITAS SEBELAS MARET

2024

BAB I
PENDAHULUAN

A. Penjabaran dari masalah

Coin Change Problem adalah masalah dalam menentukan jumlah minimum koin yang diperlukan untuk mencapai suatu nilai tertentu, menggunakan denominasi koin yang tersedia. Masalah ini umumnya diselesaikan menggunakan algoritma greedy, yang bekerja dengan prinsip memilih koin terbesar yang tidak melebihi sisa jumlah, hingga jumlah target terpenuhi. Dengan Algoritma Greedy masalah tersebut akan optimal terhadap koin yang terstandar.

B. Pembagian tugas dalam kelompok

1. Moh Ferdinand Ramdhani: Membuat video demo, menyusun laporan.
2. M Al Fathi Ayyash : Membuat program, menyusun laporan.

BAB II
IMPLEMENTASI

A. Penjelasan implementasi/source code

Link Github : https://github.com/Alfaashh/Greedy-Coin-Change

```
coins = [1, 2, 5, 10]
print(f"Koin yang tersedia: {coins}")

try:
  amount = int(input("Nilai uang yang ingin ditukarkan: "))
  count, used = coin_change_greedy(coins, amount)
  if count != -1:
    print(f"Koin minimum yang dibutuhkan: {count}")
    print(f"Koin yang terpakai: {used}")
  else:
    print("Tidak bisa menukar nilai uang dengan nominal koin yang
ada.")


except ValueError:
  print("Input harus berupa integer.")
```

Saat program dijalankan, kode diatas akan berjalan. Program akan mencetak koin-koin yang tersedia (1, 2, 5, 10), lalu program akan meminta masukan atau *input* dari pengguna, yakni Nilai uang yang ingin ditukarkan, dan masukan dari pengguna akan disimpan pada variabel *amount*.

Setelah itu program akan menggunakan fungsi *coin_change_greedy()* dengan parameter *coins* dan *amount* yang telah di-*input* pengguna. Jika fungsi *coin_change_greedy()* berjalan dan memiliki kembalian tidak sama dengan -1, maka program akan mencetak koin minimum yang dibutuhkan *(count)* dan koin yang terpakai apa saja *(used)*. Namun, jika kembalian *coin_change_greedy()* adalah sama dengan -1, maka program akan mencetak "Tidak bisa menukar nilai uang dengan nominal koin yang ada.".

Program juga melakukan *Error Handling*, yakni apakah *input* pengguna sesuai (integer). Jika tidak sesuai, maka akan mencetak "Input harus berupa integer."

```python
def coin_change_greedy(coins, amount):
  coin_count = 0
  coin_used = []

  coins.sort(reverse=True)

  for coin in coins:
    while amount >= coin:
      amount -= coin
      coin_count += 1
      coin_used.append(coin)

  if amount == 0:
    return coin_count, coin_used
  else:
    return -1, []
```

Fungsi *coin_change_greedy()* menerima parameter *coins* dan *amount,* dengan melakukan deklarasi dimana *coin_count* bernilai 0 (akan bertambah jika ada koin yang ditukarkan) dan *coin_used* memiliki array yang kosong.

Lalu akan dilakukan sorting (*coins.sort())* yang melakukan sorting *ascending* berkat *reverse=True*.

Pada for *loop*, akan mengiterasi isi array *coins,* dan akan menggunakan nilai *coin* terbesar selama nilai *amount* tidak lebih besar sama dengan nilai *coin.* Jika nilai *amount* lebih kecil daripada nilai *coin*, maka akan melanjutkan iterasi ke index(*coin)* selanjutnya. Selanjutnya, nilai *amount* akan dikurangi senilai dengan nilai coin yang sedang diiterasikan, dan juga *coin_count* akan bertambah satu yang menunjukkan jumlah koin yang digunakan bertambah serta memasukkan *coin* yang digunakan ke dalam array *coin_used.*

Jika nilai *amount* telah habis atau sama dengan 0, maka akan mengembalikan *coin_count* dan *coin_used.* Namun, jika nilai *amount* tidak menyentuh 0 atau tidak sama dengan 0, maka akan mengembalikan nilai -1 yang nantinya akan menampilkan pesan error.

BAB III
HASIL

A. Pengujian

Beberapa bentuk input untuk menguji program, diantaranya:

1. Invalid case input
   a. Angka desimal

```
Koin yang tersedia: [1, 2, 5, 10]
Nilai uang yang ingin ditukarkan: 5.6
Input harus berupa integer.
```

```
amount = int(input("Nilai uang yang ingin ditukarkan: "))
```

   b. Angka kurang dari nol

```
Koin yang tersedia: [1, 2, 5, 10]
Nilai uang yang ingin ditukarkan: -10
Tidak bisa menukar nilai uang dengan nominal koin yang ada.
```

```
amount = int(input("Nilai uang yang ingin ditukarkan: "))
```

   c. Input dengan bentuk string dan *special character*

```
Koin yang tersedia: [1, 2, 5, 10]
Nilai uang yang ingin ditukarkan: hai
Input harus berupa integer.
```

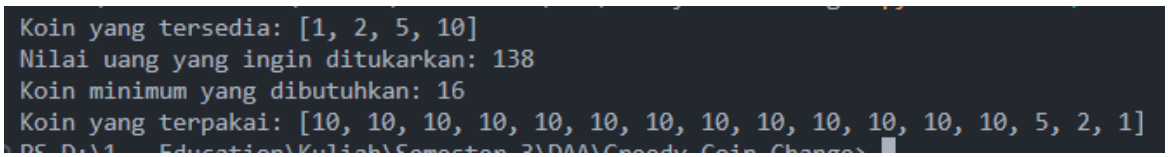
```
Koin yang tersedia: [1, 2, 5, 10]
Nilai uang yang ingin ditukarkan: %n
Input harus berupa integer.
```

```
amount = int(input("Nilai uang yang ingin ditukarkan: "))
```

2. Valid input
   a. Angka bilangan bulat

```
Koin yang tersedia: [1, 2, 5, 10]
Nilai uang yang ingin ditukarkan: 32
Koin minimum yang dibutuhkan: 4
Koin yang terpakai: [10, 10, 10, 2]
```

```
Koin yang tersedia: [1, 2, 5, 10]
Nilai uang yang ingin ditukarkan: 4
Koin minimum yang dibutuhkan: 2
Koin yang terpakai: [2, 2]
```


Koin yang tersedia: [1, 2, 5, 10]
Nilai uang yang ingin ditukarkan: 138
Koin minimum yang dibutuhkan: 16
Koin yang terpakai: [10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 5, 2, 1]

BAB IV
KESIMPULAN

Program Coin Change Problem menggunakan algoritma Greedy untuk menentukan jumlah minimum koin yang diperlukan untuk mencapai suatu nilai tertentu. Program ini bekerja dengan meminta input nilai uang yang ingin ditukarkan dari pengguna, kemudian menggunakan fungsi *coin_change_greedy()* untuk menghitung jumlah koin minimum yang dibutuhkan. Fungsi tersebut bekerja dengan menyortir koin-koin yang tersedia dari nilai terbesar ke terkecil, lalu secara iteratif memilih koin terbesar yang tidak melebihi sisa jumlah hingga nilai target terpenuhi. Program juga menangani kasus-kasus input yang tidak valid, seperti angka desimal, angka negatif, atau input bukan angka.

Dalam kasus ini, dengan set koin yang tersedia (1, 2, 5, 10), algoritma Greedy akan menghasilkan solusi yang optimal. Ini karena set koin tersebut memiliki sifat "*canonical coin system*" di mana setiap denominasi koin adalah kelipatan dari denominasi yang lebih kecil. Dalam situasi seperti ini, pendekatan Greedy selalu menghasilkan solusi optimal dengan mengambil koin terbesar yang mungkin pada setiap langkah. Namun, penting untuk dicatat bahwa algoritma Greedy tidak selalu optimal untuk semua set koin. Untuk set koin yang tidak standar atau tidak memiliki sifat kelipatan, algoritma Greedy mungkin menghasilkan solusi suboptimal, dan dalam kasus tersebut, pendekatan lain seperti dynamic programming mungkin diperlukan untuk menemukan solusi optimal.