Laporan Dokumentasi Tugas Coin Problem dengan Algoritma Greedy Desain Analisis & Algoritma

Disusun oleh :

1. **M. Maulana Gian Pranaja (L0123077)**
2. **Primus Putra Prima (L0123111)**

Dosen Pengampu :
Fajar Muslim S.T., M.T

Program Studi Informatika
Fakultas Teknologi Informasi dan Sains Data
Universitas Sebelas Maret

2024

PENJABARAN PROBLEM

Coin Change Problem adalah sebuah masalah dalam ilmu komputer dan matematika, khususnya pada bidang algoritma dan optimalisasinya. Inti dari permasalahan ini adalah seseorang akan diberikan sejumlah koin dengan nominal yang berbeda-beda. Lalu seseorang itu diminta untuk menggunakan koin-koin tersebut untuk menggantikan sebuah nominal yang diinginkan (misalkan 20). Dalam **Coin Change Problem** ini, seseorang tersebut diminta untuk menggantikan sebuah nominal uang dengan koin-koin yang telah dia miliki, namun dengan jumlah koin paling sedikit (e.g Untuk mengganti uang **5** maka dapat menggunakan koin dengan nilai **5** dibandingkan dengan menggunakan uang dengan nilai **1** sebanyak 5 kali)

Coin Change Problem dapat diselesaikan dengan berbagai cara, antara lain adalah dengan menggunakan algoritma **Greedy**, Dynamic Programming, atau dengan rekursif dengan memoisasi. Dalam kasus ini, **Coin Change Problem** akan diselesaikan dengan menggunakan algoritma **Greedy**. Algoritma **Greedy** bekerja dengan konsep "*take the best that we can get now*" tanpa melihat kemungkinan lainnya. Artinya algoritma **Greedy** akan selalu mengambil nominal koin atau uang yang paling besar terlebih dahulu.

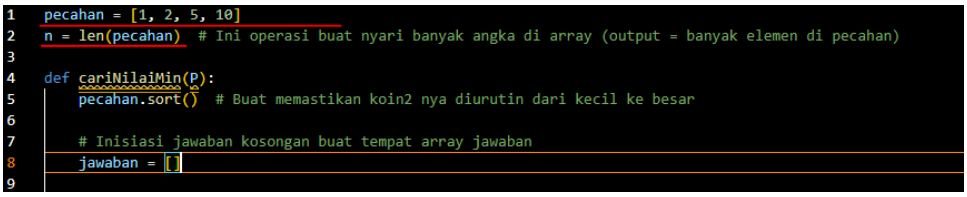
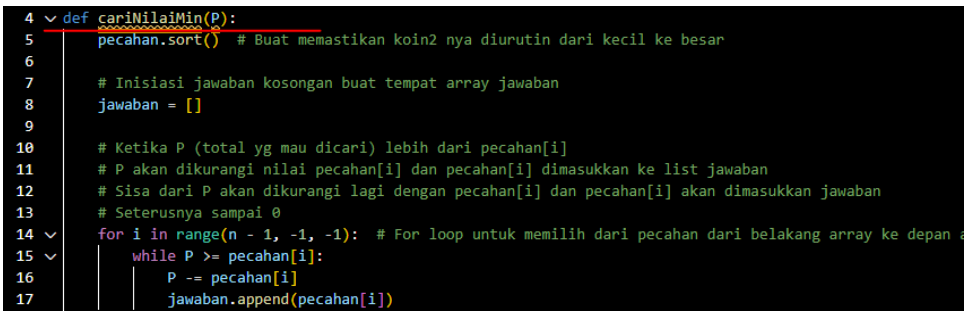
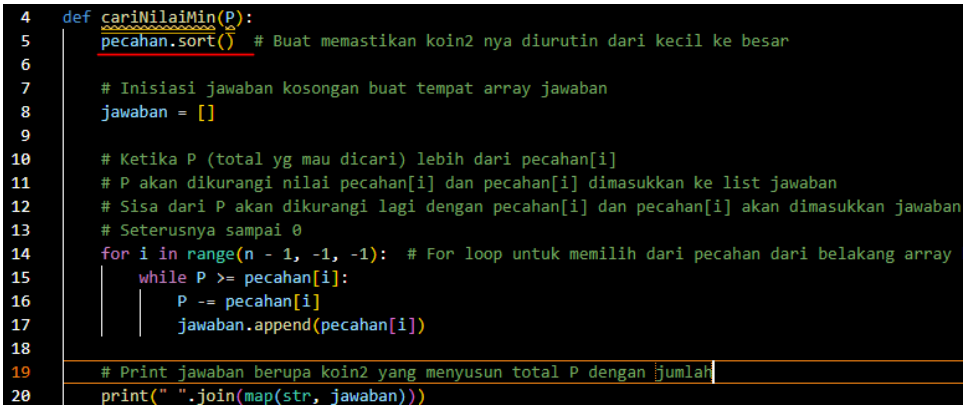
DEMONSTRASI PROGRAM (DEMO)

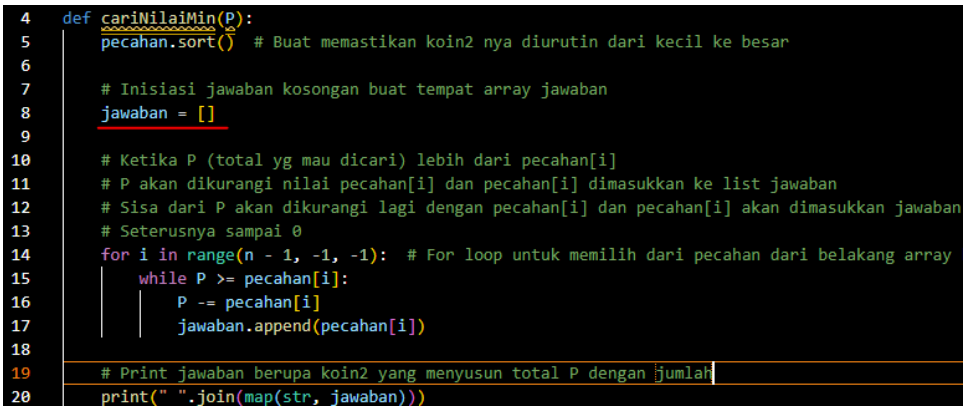
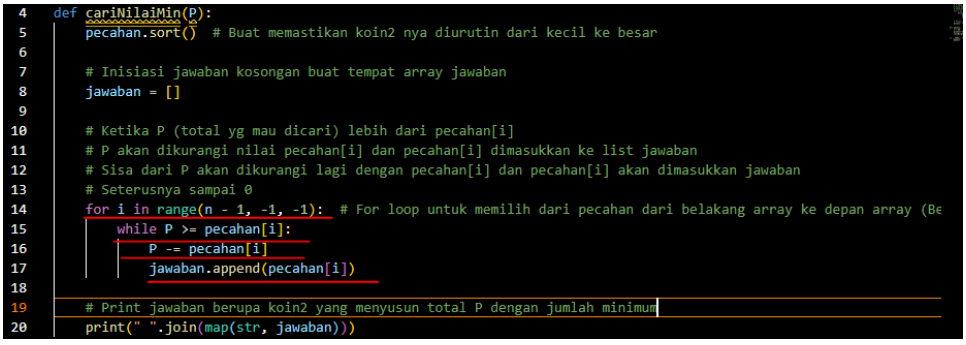
Tutorial Greedy Method

Link youtube : https://youtu.be/uqF3jLPTzwc

Link github : Kingshellter/GreedyPython-Tugas-DAA (github.com)

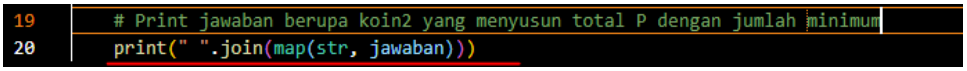
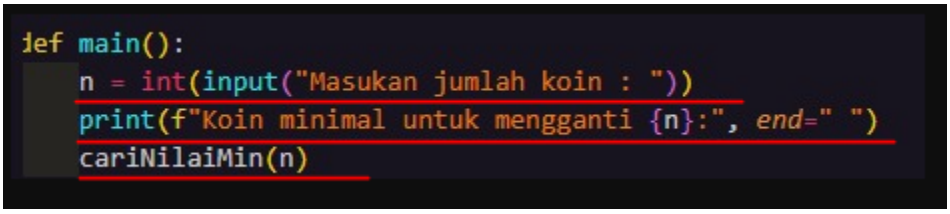
Penjelasan Implementasi

No.	Implementasi	PERAN
1.	Pengaturan Nominal	 Pada line ini, terjadi pemberian nominal koin. Di dalam ‘pecahan’ merepresentasikan nilai-nilai koin yang akan diberikan, yaitu 1, 2, 5, dan 10. Kemudian diikuti dengan ‘n = len(pecahan)’. Line ini akan memberikan variabel ‘n’ dengan jumlah elemen yang ada pada ‘pecahan’
2.	Fungsi cariNilaiMin	 Fungsi ini akan mencari jumlah minimum koin yang diperlukan untuk memenuhi nilai ‘P’
3.	Pengurutan Koin	

		Fungsi dimulai dengan mengurutkan koin-koin yang dimasukkan. Fungsi pengurutan ini memastikan agar ‘pecahan’ yang dimasukkan sudah benar-benar diurutkan dari kecil ke besar. Meskipun **Greedy** akan memilih nilai dari besar ke kecil, pengurutan ini dilakukan agar memudahkan algoritma untuk menemukan mana nilai terbesar dan mana nilai terkecil. Pengurutan menggunakan ‘pecahan.sort()’ yang dimana ‘sort()’ merupakan bagian dari Python untuk mengurutkan sebuah array.
4.	Inisiasi Jawaban	 Setelah ‘pecahan’ diurutkan, akan dilanjutkan dengan inisiasi jawaban. Inisiasi ini berguna sebagai tempat dari jawaban nantinya setelah algoritma menemukan solusi optimal untuk **Coin Change Problem** menggunakan algoritma **Greedy**. Line ‘jawaban = []’ adalah berupa array kosong untuk menyimpan jawaban.
5.	Algoritma Utama	 Bagian ini merupakan bagian utama yang menentukan jalannya algoritma **Greedy**.

```
4  def cariNilaiMin(P):
5      pecahan.sort()  # Buat memastikan koin2 nya diurutin dari kecil ke besar
6
7      # Inisiasi jawaban kosongan buat tempat array jawaban
8      jawaban = []
9
10     # Ketika P (total yg mau dicari) lebih dari pecahan[i]
11     # P akan dikurangi nilai pecahan[i] dan pecahan[i] dimasukkan ke list jawaban
12     # Sisa dari P akan dikurangi lagi dengan pecahan[i] dan pecahan[i] akan dimasukkan jawaban
13     # Seterusnya sampai 0
14     for i in range(n - 1, -1, -1):  # For loop untuk memilih dari pecahan dari belakang array
15         while P >= pecahan[i]:
16             P -= pecahan[i]
17             jawaban.append(pecahan[i])
18
19     # Print jawaban berupa koin2 yang menyusun total P dengan jumlah
20     print(" ".join(map(str, jawaban)))
```

```
4  def cariNilaiMin(P):
5      pecahan.sort()  # Buat memastikan koin2 nya diurutin dari kecil ke besar
6
7      # Inisiasi jawaban kosongan buat tempat array jawaban
8      jawaban = []
9
10     # Ketika P (total yg mau dicari) lebih dari pecahan[i]
11     # P akan dikurangi nilai pecahan[i] dan pecahan[i] dimasukkan ke list jawaban
12     # Sisa dari P akan dikurangi lagi dengan pecahan[i] dan pecahan[i] akan dimasukkan jawaban
13     # Seterusnya sampai 0
14     for i in range(n - 1, -1, -1):  # For loop untuk memilih dari pecahan dari belakang array ke depan array (Be
15         while P >= pecahan[i]:
16             P -= pecahan[i]
17             jawaban.append(pecahan[i])
18
19     # Print jawaban berupa koin2 yang menyusun total P dengan jumlah minimum
20     print(" ".join(map(str, jawaban)))
```

		1. for loop akan mengiterasikan angka-angka yang ada dalam ‘pecahan’ mulai dari nilai terbesar hingga terkecil, sehingga ‘for’ loop diatur agar mengakses array ‘pecahan’ dari belakang ke depan 2. ‘while’ loop: a. Akan mengecek nilai pada ‘pecahan’ (menggunakan ‘pecahan[i]’) apakah bisa digunakan, dengan mengecek apakah ‘pecahan[i]’ >= nilai ‘P’. Apabila iya maka: i. ‘pecahan[i]’ tersebut akan dikurangkan terhadap P dan dalam waktu yang sama, dimasukkan ke dalam array ‘jawaban’ b. Loop ini akan berlangsung terus menerus hingga nilai P bernilai lebih kecil dari ‘pecahan[i]’. i. Loop hanya akan berhenti jika sisa nilai P bernilai 0, atau jika sisa nilai P tidak bisa disusun lagi oleh elemen-elemen ‘pecahan’ (e.g jika diberi koin 2, dan 3 dan haru menyusun nilai P sebesar 4, maka Greedy akan memilih 3, sehingga sisa nilai P adalah 1 yang tidak bisa disusun lagi oleh elemen koin yang diberikan, sehingga loop berhenti)
6.	Join	 Ketika algoritma sudah menyelesaikan fungsi-fungsinya, maka array ‘jawaban’ akan di print. Array yang di print dipisahkan oleh karakter space.
7.	Fungsi Main	 Bagian fungsi main: 1. Mengatur nilai ‘n’ dengan nilai yang diinginkan. Nilai ‘n’ ini nanti yang akan merepresentasikan nilai ‘P’. Dalam line ini, pengguna bisa memasukkan nilai ‘n’ sebagai nilai total yang diinginkan

		2. Kemudian akan memprint jawaban yang ditemukan oleh algoritma 3. Memanggil fungsi ‘cariNilaiMin’ dengan argument ‘n’

PENGUJIAN KODE

1. Solusi Optimal

```
PS D:\Proyek Python> & D:/Python/filez/python.exe "d:/Proyek Python/coin.py"
Koin minimal untuk mengganti 20: 10 10
PS D:\Proyek Python>
```

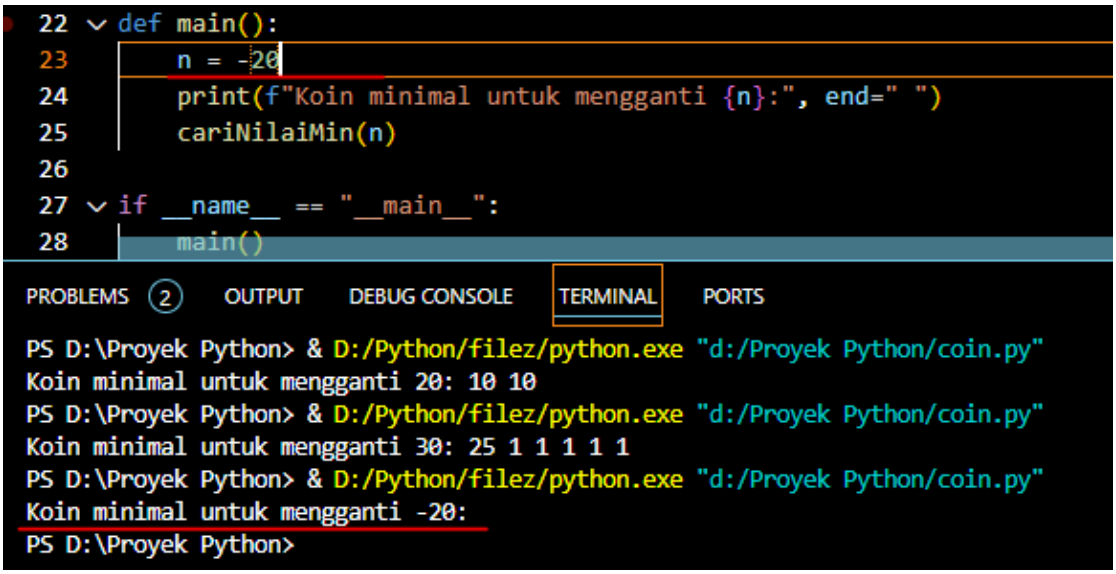
2. Solusi Kurang Optimal

```
1   pecahan = [1, 15, 25]
2   n = len(pecahan)  # Ini operasi buat nyari banyak angka di array (output = ba
3
4   def cariNilaiMin(P):
5       pecahan.sort()  # Buat memastikan koin2 nya diurutin dari kecil ke besar
6
7       # Inisiasi jawaban kosongan buat tempat array jawaban
8       jawaban = []
9
10      # Ketika P (total yg mau dicari) lebih dari pecahan[i]
11      # P akan dikurangi nilai pecahan[i] dan pecahan[i] dimasukkan ke list jaw
12      # Sisa dari P akan dikurangi lagi dengan pecahan[i] dan pecahan[i] akan d
13      # Seterusnya sampai 0
14      for i in range(n - 1, -1, -1):  # For loop untuk memilih dari pecahan dar
15          while P >= pecahan[i]:
16              P -= pecahan[i]
17              jawaban.append(pecahan[i])
18
19      # Print jawaban berupa koin2 yang menyusun total P dengan jumlah minimum
20      print(" ".join(map(str, jawaban)))
```

```
PROBLEMS (2)   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS
PS D:\Proyek Python> & D:/Python/filez/python.exe "d:/Proyek Python/coin.py"
Koin minimal untuk mengganti 20: 10 10
PS D:\Proyek Python> & D:/Python/filez/python.exe "d:/Proyek Python/coin.py"
Koin minimal untuk mengganti 30: 25 1 1 1 1 1
PS D:\Proyek Python>
```

Solusi Optimal : **15 15**

3. Tidak Ditemukan Solusi



PERAN ANGGOTA KELOMPOK	
Anggota	Peran
M. Maulana Gian Pranaja	Video demo, Upload github, Pengujian kode
Primus Putra Prima	Kode konsep (CPP), Implementasi kode, Pengujian kode