

PROJECT DESAIN ANALISIS ALGORITMA COIN CHANGE PROBLEM DENGAN ALGORITMA GREEDY

Laporan Desain Analisis Algoritma
Disusun untuk Memenuhi Tugas Mata Kuliah Desain Analisis Algoritma
Dosen Pengampu:
Fajar Muslim, S.T, M.T.



Disusun Oleh:

- | | |
|--------------------|------------|
| 1. Mutiara Hasanah | (L0123102) |
| 2. Nayla Amira | (L0123108) |

**PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA
UNIVERSITAS SEBELAS MARET
SURAKARTA
2024**

BAB 1 PENDAHULUAN

Permasalahan Coin Change adalah tantangan yang penyelesaiannya dengan melibatkan eksplorasi kombinasi yang mungkin untuk menemukan solusi yang optimal. Dalam kasus ini, Dalam kasus ini, kita harus mencari kombinasi koin dengan denominasi yang berbeda-beda untuk mencapai sejumlah nilai tertentu dengan menggunakan jumlah koin sesedikit mungkin. Setiap denominasi koin dapat digunakan lebih dari satu kali, dan urutan penggunaan koin dapat bervariasi, yang berarti ada banyak kemungkinan kombinasi yang harus dieksplorasi.

Tantangan utama dari masalah ini adalah kompleksitas dalam eksplorasi semua kemungkinan kombinasi koin yang akan menghasilkan jumlah koin sesedikit mungkin tetapi mencapai nilai tertentu. Pendekatan untuk menyelesaikan masalah ini sering menggunakan metode algoritma *greedy*. Dalam algoritma *greedy*, kita selalu memilih koin dengan nilai terbesar yang masih bisa digunakan pada setiap langkah.

BAB 2 DASAR TEORI

Landasan teori yang akan digunakan pada penerapan strategi algoritma dalam kasus ‘Coin Change Problem’ menggunakan algoritma *Greedy*.

A. Definisi *Greedy*

Greedy algorithm merupakan salah satu pendekatan yang populer untuk menyelesaikan Persoalan Optimasi (*Optimization Problem*) dimana persoalan ini menuntut pencarian solusi optimal. Dalam persoalan optimasi, hanya terdapat 2 macam yaitu maksimasi (*maximization*) dan minimasi (*minimization*). Algoritma ini merupakan kelompok yang selalu mengambil penyelesaian sementara yang terbaik dalam setiap langkahnya untuk menyelesaikan suatu permasalahan. Algoritma *Greedy* memilih opsi terbaik pada setiap tahap tanpa mempertimbangkan dampaknya pada tahap berikutnya.

B. Karakteristik Algoritma *Greedy*

Algoritma *Greedy* memiliki karakteristik dimana ia selalu memilih opsi terbaik yang tersedia pada setiap langkahnya. Algoritma ini tidak mempertimbangkan keputusan selanjutnya dan hanya fokus pada opsi terbaik di langkah tersebut.

C. Keunggulan dan Kekurangan Algoritma *Greedy*

Ada beberapa keunggulan dan kekurangan pada algoritma *Greedy*. Algoritma ini sangat cepat dalam mengambil keputusan, sehingga hal tersebut membuat efisiensi pada algoritma ini cukup baik dan tidak memerlukan waktu banyak. Tidak hanya itu, algoritma ini sangat mudah diimplementasikan pada beberapa permasalahan yang ada. Namun, karena keputusan yang cepat tersebut, algoritma *Greedy* memiliki hasil akhir yang kurang baik dibanding dengan *Brute-force* dan tidak ada opsi lain jika persoalan tidak dapat diselesaikan.

BAB 3 PENERAPAN GREEDY

1. Algoritma *Greedy* pada permasalahan Coin Change

Algoritma *Greedy* untuk masalah ini melibatkan pencarian jumlah minimal koin yang diperlukan untuk mencapai suatu jumlah target tertentu, dengan memanfaatkan denominasi koin yang tersedia. Prinsip utama dari algoritma *Greedy* adalah selalu memilih koin dengan nilai terbesar yang masih memungkinkan untuk digunakan hingga mencapai target jumlah. Langkah-langkah implementasi nya sebagai berikut:

a. Inisialisasi Nilai Target

Input nilai target yang ingin dicapai dan denominasi koin yang tersedia, misalnya A dan a,b,c,d

b. Penggunaan OperatorAritmatika

Operator dasar aritmatika yang digunakan adalah pembagian (/) dan modulo (%)

- **Pembagian (/)**: Untuk menentukan berapa banyak koin dari denominasi tertentu yang diperlukan.
- **Modulus (%)**: Untuk menghitung sisa nilai yang belum tercapai.

c. Mencari Kombinasi Operasi

- Memilih koin terbesar yang memungkinkan untuk operasi

$$a > b > c > d$$

- Membagi nilai target dengan koin terbesar untuk menghitung banyak koin terbesar yang terpakai

$$a / a = m$$

- Mencari sisa pembagian untuk mengurangi nilai target dengan koin selanjutnya

$$a \% a = n$$

- Mengulangi proses

$$n / b = o$$

$$n \% b = p$$

d. Memeriksa Solusi

Setelah setiap langkah perhitungan selesai, nilai target diperiksa apakah sudah mencapai 0. Jika nilai target sudah mencapai 0, algoritma telah menemukan solusi optimal dan berhenti. Solusi berupa jumlah minimal koin yang diperlukan untuk mencapai nilai target akan ditampilkan.

Jika hasil akhir sesuai, algoritma berhenti, dan kombinasi koin yang digunakan akan ditampilkan.

e. *Worst-case Scenario*

Dalam *worst-case Scenario*, jika tidak ada koin yang sesuai untuk mencapai nilai target secara optimal, algoritma akan tetap berlanjut hingga seluruh kombinasi koin yang memungkinkan diuji. Selain itu, jika nilai target yang dimasukkan lebih kecil dari nilai koin, maka program tidak menjalankan apa apa. Namun, dalam kasus denominasi standar seperti contoh di atas, algoritma Greedy selalu memberikan solusi optimal.

2. Implementasi Solusi pada Pemrograman

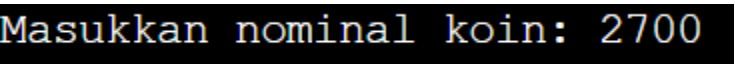
Program ini memecah nominal uang yang dimasukkan pengguna menjadi pecahan koin terbesar yang tersedia. Dengan menggunakan loop, program membagi nominal uang dengan koin terbesar yang mungkin, lalu menghitung jumlah koin untuk setiap pecahan. Setelah setiap pembagian, nominal diperbarui dengan sisa yang belum terpakai, dan proses berlanjut dengan pecahan koin yang lebih kecil hingga nominalnya habis.

Alur Umum pada program ini yaitu :

1. Pengguna memasukkan input angka sebagai target
2. Program akan mencari nilai array terbesar untuk membagi dan mengambil sisa pembagian dari target
3. Program akan mengulangi operasi dari sisa pembagian target
4. Jika nilai target sudah habis, maka program akan menampilkan hasil
5. Jika nilai target tidak bisa habis, algoritma akan tetap berlanjut untuk mencari solusi

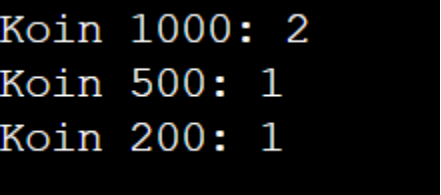
3. Testing dan Hasil Tangkapan Layar Input dan Output

a. Input pada terminal

A screenshot of a terminal window showing the prompt "Masukkan nominal koin:" followed by the user input "2700".

```
Masukkan nominal koin: 2700
```

b. Output pada terminal

A screenshot of a terminal window showing the output of the program. It lists the number of coins for each denomination: "Koin 1000: 2", "Koin 500: 1", and "Koin 200: 1".

```
Koin 1000: 2
Koin 500: 1
Koin 200: 1
```

BAB 4 PROGRAM DALAM BAHASA PYTHON

Berikut ini adalah kode program yang dapat menyelesaikan coin change dengan metode Algoritma *Greedy* dan ditulis dalam bahasa pemrograman Python.

```
1  def main():
2      x = int(input("Masukkan nominal koin: "))
3
4      arr = [1000, 500, 200, 100]
5      i = 0
6
7      while x > 0 and i < len(arr):
8          jumlahKoin = x // arr[i]
9          if jumlahKoin > 0:
10             print(f"Koin {arr[i]}: {jumlahKoin}")
11             x = x % arr[i]
12             i += 1
13
14  if __name__ == "__main__":
15      main()
16
17
```

BAB 5 TABEL PENILAIAN

Poin	Ya	Tidak
Program berhasil dikompilasi tanpa kesalahan	✓	
Program berhasil running.	✓	
Program dapat membaca input / generate sendiri dan memberikan luaran	✓	
Solusi yang diberikan program memenuhi (berhasil mencari jumlah koin minimal)	✓	

BAB 6 KESIMPULAN

Dalam melakukan penyelesaian permasalahan coin change menggunakan algoritma *Greedy*, yaitu: dengan memilih koin terbesar yang masih dapat digunakan untuk mencapai sisa nilai target. Pada Laporan Desain Analisis Algoritma ini, penulis membuat program untuk menyelesaikan permasalahan coin change yang ditulis dalam bahasa pemrograman Python.

Algoritma Greedy pada permasalahan coin change sangat efektif dalam menentukan jumlah minimal koin yang diperlukan untuk mencapai nilai target dengan cepat. Meskipun algoritma ini tidak mengeksplorasi semua kemungkinan kombinasi koin seperti pendekatan brute force, ia tetap memberikan solusi yang mendekati optimal dengan memilih koin terbesar yang mungkin pada setiap langkah. Keunggulan utama dari algoritma Greedy adalah kesederhanaannya dan kecepatan eksekusi, namun ia memiliki keterbatasan pada konfigurasi koin tertentu, di mana solusi yang dihasilkan mungkin tidak selalu optimal. Meski demikian, dalam kasus denominasi standar, algoritma ini menawarkan hasil yang efisien tanpa memerlukan kompleksitas tambahan seperti dynamic programming.

BAB 7 REFERENSI

https://kantinit.com/kecerdasan-buatan/algoritma-greedy-konsep-karakteristik-dan-contohnya/#Karakteristik_Algoritma_Greedy

<https://www.anakblogger.com/2020/02/kelebihan-kekurangan-algoritma-greedy.html>

Roihan, A., Nasution, K., & Siambaton, M. Z. (2022). Implementasi Algoritma Greedy

Kombinasi dengan Perulangan pada Aplikasi Penjadwalan Praktikum. *Sudo Jurnal*

Teknik Informatika, 1(2), 42–50. <https://doi.org/10.56211/sudo.v1i2.8>