# LAPORAN IMPLEMENTASI GREEDY ALGORITHM

**DISUSUN OLEH**


**MOHAMMAD ADZKA CROSAMER** **L0123083**

**NAUFAL NARENDRO KUSUMO** **L0123107**

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA**

**UNIVERSITAS SEBELAS MARET**


**2024**

# PENDAHULUAN

Prinsip dasar algoritma greedy adalah membuat pilihan yang tampak terbaik pada setiap langkah tanpa mempertimbangkan konsekuensi jangka panjang, dengan harapan bahwa solusi lokal terbaik pada setiap langkah akan menghasilkan solusi global yang optimal atau mendekati optimal. Algoritma greedy adalah salah satu pendekatan penyelesaian masalah yang sering digunakan dalam bidang ilmu komputer, khususnya dalam pencarian solusi yang efisien.

Untuk masalah dengan struktur spesifik seperti knapsack, Huffman coding, minimum spanning tree, dan masalah pemilihan aktivitas, algoritma greedy biasanya digunakan. Algoritma ini memiliki keunggulan dalam hal kesederhanaan, kecepatan, dan efisiensi komputasi, meskipun tidak selalu menghasilkan solusi terbaik untuk semua jenis masalah.

Laporan ini akan membahas penggunaan algoritma greedy untuk masalah perubahan koin, termasuk penjelasan tentang desain algoritma, analisis kode, dan pengujian dari program. Dengan adanya laporan ini, diharapkan pembaca dapat memahami bagaimana algoritma greedy bekerja serta kapan algoritma ini dapat digunakan secara efektif dalam penyelesaian masalah.

## DESAIN ALGORITMA

Algoritma greedy yang digunakan untuk menyelesaikan masalah coin change bertujuan untuk menemukan jumlah minimum koin yang diperlukan untuk mencapai nilai target tertentu. Dalam algoritma ini, kita selalu memilih koin dengan nilai terbesar yang dapat digunakan pada setiap langkah, dengan harapan bahwa solusi lokal terbaik akan mengarah pada solusi global yang optimal. Algoritma ini sangat sederhana dan efisien untuk beberapa jenis denominasi koin, seperti koin [1, 2, 5, 10] pada kasus ini, meskipun tidak selalu memberikan solusi optimal untuk semua jenis koin.

Pertama-tama,kita mengurutkan semua jenis koin yang tersedia dari yang terbesar hingga yang terkecil. Misalnya, jika koin yang tersedia adalah [1, 2, 5, 10], maka setelah diurutkan, kita akan memiliki urutan [10, 5, 2, 1]. Setelah jenis koin diurutkan, langkah selanjutnya adalah mencoba memilih koin terbesar yang bisa digunakan untuk mengurangi jumlah target. Kita akan terus menggunakan koin terbesar yang memungkinkan hingga tidak ada lagi yang bisa digunakan, kemudian beralih ke koin terbesar berikutnya.

Pada setiap langkah, kita membandingkan koin terbesar yang tersisa dengan nilai target. Jika koin tersebut lebih kecil atau sama dengan nilai target, kita mengurangkan nilai koin tersebut dari target dan menambahkannya ke total koin yang digunakan. Langkah ini diulang hingga nilai target menjadi 0, yang menunjukkan bahwa solusi telah ditemukan. Terkadang ada kasus di mana nilai target tidak bisa menjadi 0, ini menandakan bahwa nilai target yang kita masukkan tidak bisa dibagi jenis koin yang ada dengan metode greedy.

Setelah pengurangan target dengan jenis koin, kita menghitung berapa banyak koin tersebut digunakan. Untuk setiap jenis yang ada, kita membagi sisa target dengan nilai koin untuk menghitung jumlah koin yang bisa digunakan. Sebagai contoh, jika sisa target adalah 12 dan kita memiliki koin dengan nilai 10, kita dapat menggunakan satu koin 10 dan mengurangi sisa target menjadi 2. Proses ini berlanjut hingga semua koin terpakai dan sisa target mencapai 0. Penghentian algoritma ini terjadi apabila target habis atau sudah tidak bisa dibagi lagi.

Algoritma greedy sangat efisien dalam menyelesaikan masalah dengan denominasi koin standar. Namun, dalam beberapa kasus di mana jenis koin memiliki hubungan yang sedikit kompleks, algoritma ini mungkin tidak selalu menghasilkan solusi optimal. Contohnya target 6 dengan jenis [4, 3, 1], algoritma ini akan mengurangi 6 dengan 4 yang

membuatnya harus mengurangi 2 dengan 1, 1 yang mengakibatkan penggunaan 3 koin, padahal bisa diselesaikan dengan 2 koin yaitu 3, 3. Untuk mengatasi keterbatasan ini, metode lain seperti dynamic programming dapat digunakan sebagai alternatif yang lebih efektif dalam beberapa kasus.

Dengan desain algoritma seperti ini, solusi greedy memanfaatkan strategi memilih koin terbesar terlebih dahulu dan melanjutkan hingga mencapai nilai target. Algoritma ini sederhana dan cepat, tetapi dengan beberapa kelemahan pada jenis denominasi koin tertentu.

## ANALISIS KODE

```
uang = int(input("Jumlah uang: "))
jenis_koin = input("Masukkan koin (pisahkan dengan spasi): ")
```

Potongan berikut digunakan untuk mendeklarasikan uang dan jenis_koin yang membolehkan pengguna untuk memasukkan nilai.

- int(input(): input digunakan bagi pengguna untuk memasukkan nilai, sedangkan int digunakan untuk mengubah tipe data yang awalnya string “Jumlah uang: “ menjadi integer.
- jenis_koin = input(): digunakan user untuk menginput / memasukkan nilai dari jenis_koin.

```
coins = list(map(int, jenis_koin.split()))
coins.sort(reverse=True)
```

Fungsi berikut digunakan untuk mengubah tipe data dan mengurutkannya.

- split(): digunakan untuk memisahkan satu string, dalam kasus ini digunakan untuk memisahkan (“1 2 5”) menggunakan spasi menjadi (“1”, “2”, “5”).
- map(int): berguna untuk mengubah string menjadi integer pada setiap index.
- list(): mengubah input yang sudah diubah menjadi integer menjadi array
- sort(reverse=True): berguna untuk mengurutkan list, reverse=True digunakan untuk membuat list menjadi descending.

```
def kembalian(kebutuhan):
    result = []
    for koin in coins:
        while kebutuhan >= koin:
            result.append(koin)
            kebutuhan -= koin
            print(f"Memilih koin {koin}, sisa uang {kebutuhan}")
    if kebutuhan > 0:
        return None
    return result
```

Ini adalah fungsi utama pada algoritma greedy kami. Fungsi kembalian digunakan untuk mendeklarasi array hasil dari banyak koin yang digunakan dan mencetaknya.

- result = []: deklarasi hasil dari koin pembagi uang yang nantinya akan dimasukkan.
- for koin in coins: for loop biasa yang mana koin adalah tiap index dari coins.
- while digunakan untuk terus mengulang selama target masih lebih besar atau sama dengan koin yang akan dilakukan pengurangan nantinya.
- append(): memasukkan koin ke dalam array result
- print(f””): mencetak koin yang dipakai untuk pengurangan serta mencetak sisa target. f pada print adalah formatted string literal yang membolehkan kita mencetak variabel tanpa mengubah tipe data dari variabel tersebut terlebih dahulu.
- jika target(kebutuhan) tidak berhasil dibagi habis oleh koin, maka program akan mengembalikan None (dalam kasus ini berarti tidak ada solusi), sedangkan jika fungsi berjalan lancar tanpa terhambat if statement maka fungsi akan langsung mengembalikan result.

```
result = kembalian(uang)
```

menginisialisasikan variabel result menggunakan fungsi yang bertarget uang

```
if result is None:
        print(f"\nUang {uang} tidak bisa ditukar dengan koin yang
tersedia.")
else:
    print(f"\nUang {uang} bisa ditukar menjadi: {result}")
```

Pada fungsi sebelumnya sudah dijelaskan jika kebutuhan(target) masih lebih besar dari 0 berarti algoritma greedy tidak berhasil menemukan koin pembaginya yang menyebabkan return None, pada if statement ini jika result adalah None, program akan mencetak jumlah uang diikuti dengan pemberitahuan tidak bisa ditukar. Sedangkan else yang berarti return result(tidak None) akan mencetak jumlah uang dan array dari koin yang bisa membaginya.

**PENGUJIAN**

Pertama-tama setelah membuka program, kita akan menjalankan program. Setelah menjalankan, kita harus mengisi informasi yang diminta oleh program, seperti berikut:

```
Jumlah uang:
```

```
Masukkan koin (pisahkan dengan spasi):
```

Masukkan nilai uang dan jenis koin yang ada, dalam kasus ini kami menggunakan array coin yang sama dengan contoh pada google slides:

```
Jumlah uang: 27
Masukkan koin (pisahkan dengan spasi): 1 2 5 10
```

Setelah memasukkan informasi, program akan mencetak koin yang digunakan untuk mengurangi target

```
Memilih koin 10, sisa uang 17
Memilih koin 10, sisa uang 7
Memilih koin 5, sisa uang 2
Memilih koin 2, sisa uang 0
```

Setelah dikurangi, kita bisa mengetahui berapa saja koin yang bisa digunakan untuk mengurangi target dalam bentuk array:

```
Uang 27 bisa ditukar menjadi: [10, 10, 5, 2]
```

**PEMBAGIAN TUGAS**

| Nama | Peran |
|---|---|
| Mohammad Adzka Corsamer | Mengerjakan kode menggunakan Python, membantu pembuatan laporan, membuat video, mengatur repository GitHub. |
| Naufal Narendro Kusumo | Membuat laporan, membantu pengerjaan python, membantu troubleshooting |