LAPORAN DOKUMENTASI GREEDY ALGORITHM
DESAIN DAN ANALISIS ALGORITMA

Disusun untuk Memenuhi Tugas Mata Kuliah

Desain Analisis Algoritma

Dosen Pengampu:

Fajar Muslim S.T., M.T.

Disusun Oleh:

1. Muhammad Farhan Khairullah (L0123093)
2. Muhammad Rifai Ageng Pambudi (L0123098)
3. Muiz Afif Mirza Lindu Aji (L0123099)

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA

UNIVERSITAS SEBELAS MARET

2024

Pengertian Masalah

Greedy Algorithm: Algoritma greedy adalah algoritma yang mengikuti heuristik pemecahan masalah dengan membuat pilihan optimal lokal di setiap tahap. Dalam banyak masalah, strategi greedy tidak menghasilkan solusi optimal, tetapi heuristik greedy dapat menghasilkan solusi optimal lokal yang mendekati solusi optimal global dalam waktu yang wajar.

Greedy Algorithm: Coin Exchange merupakan sebuah program yang melakukan penukaran beberapa denominasi koin dengan sejumlah uang yang ditentukan agar dapat menggunakan kuantitas koin seminim mungkin (semakin sedikit koin yang digunakan untuk mencapai uang yang diinginkan maka disebut **solusi optimum**).

Pendekatan greedy untuk masalah ini adalah selalu menggunakan denominasi terbesar yang tidak melebihi jumlah yang tersisa, hingga seluruh jumlah uang terbayarkan.

Dalam kasus ini adalah bagaimana kita menukar uang jumlah yang kita input, dan bagaimana meraih jumlah uangnya dengan pecahan koin 1, 2, 5, 10.
Tapi pada kode kelompok kami, kami membiarkan user bisa menginput uang dan juga koin yang bisa digunakan untuk melakukan penukaran.

Demonstrasi Program

Cara menjalankan kode greedy coin exchange problem.

1. Klik run code pada file python-nya di IDE pilihan kalian.
2. Masukkan angka-angka pecahan koin yang kalian inginkan (dalam tugas ini 1, 2, 5, 10)
3. Masukkan angka uang total yang ingin ditukar menggunakan pecahan pecahan koin ini.
4. Lihat hasil yang dikeluarkan.

Dokumentasi Program

```
coin_denominations = input("Masukkan pecahan/denominasi koin : ")
coins = list(map(int, coin_denominations.split()))
coins.sort(reverse=True)   # Urutkan dari besar ke kecil
```

- **input("Masukkan pecahan/denominasi koin : "):** Program meminta pengguna untuk memasukkan pecahan koin. Misalnya, pengguna bisa

memasukkan 1, 2 , 5, 10.

- **coin_denominations.split()**: Memisahkan input integer berdasarkan spasi, menghasilkan daftar seperti ['1', '2', '5', '10'].
- **coins.sort(reverse=True)**: Mengurutkan list coins dari yang terbesar ke yang terkecil, sehingga memudahkan proses pemberian kembalian menggunakan algoritma greedy (mengambil denominasi terbesar terlebih dahulu).

```
def make_change(change_needed):
    result = []
    for coin in coins:
        while change_needed >= coin:
            result.append(coin)
            change_needed -= coin
            print(f"Pilih koin {coin}, sisa uang {change_needed}")
    return result
```

- **def make_change(change_needed):** Mendefinisikan fungsi make_change yang menerima parameter change_needed (jumlah uang yang ingin dikembalikan).
- **result = []:** Menginisialisasi list kosong untuk menyimpan denominasi koin yang dipilih.
- **for coin in coins** Iterasi melalui setiap pecahan koin, mulai dari yang terbesar karena list sudah diurutkan.
- **while change_needed >= coin**: Selama jumlah uang yang dibutuhkan masih lebih besar atau sama dengan pecahan koin saat ini, kode akan melakukan:
    - **result.append(coin):** Tambahkan pecahan koin tersebut ke dalam list hasil.
    - **change_needed -= coin:** Kurangi jumlah uang yang dibutuhkan dengan nilai pecahan koin yang dipilih.
    - **print(f"Pilih koin {coin}, sisa uang {change_needed}"):** Cetak informasi mengenai koin yang dipilih dan sisa uang yang masih harus dikembalikan.
- **return result:** Setelah semua pecahan koin diproses, kembalikan list result yang berisi pecahan-pecahan koin yang dipilih.

```
uang = int(input("Jumlah uang: "))

print(f"\nUang {uang} bisa ditukar menjadi: {make_change(uang)}")
```

- **uang = int(input("Jumlah uang: ")):** Meminta pengguna memasukkan jumlah uang yang ingin ditukar menjadi koin.
- **make_change(uang)**: Memanggil fungsi **make_change** dengan parameter uang

yang telah dimasukkan.
- **print(f"\nUang {uang}** bisa ditukar menjadi: **{make_change(uang)}"):** Menampilkan hasil kembalian dalam bentuk list pecahan koin yang dipilih.

Analisis Kompleksitas

Kompleksitas Waktu:

Menganalisis kompleksitas waktu dengan denominasi koin spesifik 1, 2, 5, 10, kita perlu mempertimbangkan bagaimana algoritma berperilaku dengan set koin ini.

- Pengurutan denominasi koin dalam array membutuhkan waktu **O(nlogn)**, di mana **n** adalah jumlah jenis koin (denominasi). Dalam kasus ini, n = 4 karena ada empat nilai koin: [1, 2, 5, 10].

- Pemilihan koin, untuk setiap denominasi, algoritma melakukan iterasi melalui koin dan terus mengurangi koin dari **change_needed** sampai tersisa 0.
    - Pada kasus terburuk, fungsi ini mungkin harus mengurangkan koin terkecil berulang kali, sehingga kompleksitas tergantung pada jumlah uang yang harus ditukar (m).
    - Jika koin terbesar dapat membagi uang dengan sempurna, iterasi lebih sedikit. Namun, jika menggunakan koin terkecil berulang-ulang, kompleksitasnya bisa linear terhadap jumlah uang (O(m)).
- Jadi, total kompleksitas waktu adalah gabungan dari kedua bagian tersebut, yaitu O(n log n + m)

Kompleksitas Ruang:

- O(n) untuk menyimpan daftar denominasi koin, di mana n adalah jumlah jenis koin yang diberikan sebagai input.
- O(m) untuk menyimpan hasil (daftar koin yang dipilih dalam result), di mana m adalah jumlah uang yang harus ditukar.

Jadi, **Kompleksitas Ruangnya**:

- Penyimpanan koin: O(n)
- Penyimpanan hasil (result): O(m)
- Secara keseluruhan: O(n+m).

Uji Kode

```
Masukkan pecahan/denominasi koin : 1 2 5 10
```

- Kode pertama kali di run akan membuat pengguna harus memasukkan pecahan koin yang akan digunakan, pada kasus ini kita menggunakan 1 2 5 dan 10.

```
Jumlah uang: 33
```

- Selanjutnya kode mengharuskan pengguna untuk memasukkan jumlah uang yang ingin ditukar dengan koin, sekarang kita coba masukkan angka 33.

```
Pilih koin 10, sisa uang 23
Pilih koin 10, sisa uang 13
Pilih koin 10, sisa uang 3
Pilih koin 2, sisa uang 1
Pilih koin 1, sisa uang 0
```

- Kode dijalankan menggunakan algoritma greedy dimana akan mengambil pilihan koin dengan denominasi paling besar terlebih dahulu. Yakni 10, untuk 30 maka akan mengambil koin 10 sebanyak 3 kali hingga uang sisa 3. Setelah itu algoritma greedy harus mengambil koin 2 sebagai angka pecahan terbesar yang tidak melebihi sisa uang yang harus ditukar, dengan begitu sisa uang yang perlu ditukar menjadi 1. Kemudian, setelahnya algoritma harus mengambil 1 untuk menghabiskan sisa uang menjadi 0.

```
Uang 33 bisa ditukar menjadi: [10, 10, 10, 2, 1]
```

- Kode akan menampilkan koin berapa saja yang diambil pada sebuah list, disini kita bisa lihat 10 koin ada 3 kali, 2 koin ada 1 kali, dan 1 koin ada 1 kali.

Apakah greedy algorithm menghasilkan optimum?

1. Pada kasus tertentu, greedy akan optimum
   Algoritma greedy memberikan solusi optimal jika kumpulan koin memenuhi sifat khusus, seperti pada koin [1, 2, 5, 10]. Dalam kasus ini, setiap nilai koin yang besar dapat dibentuk menggunakan kombinasi koin dengan denominasi lebih kecil, sehingga pendekatan greedy selalu menghasilkan solusi terbaik.
2. Algoritma greedy **tidak selalu memberikan solusi optimal** untuk semua kasus **coin change problem**, terutama ketika sistem koin yang digunakan bukan **canonical**. Sistem koin canonical adalah sistem di mana setiap nilai dapat dibuat dengan kombinasi koin yang lebih kecil, sehingga algoritma greedy selalu optimal.
   Misal pada koin: 1, 3, 4 ingin menukar uang 6.
   Algoritma greedy akan menggunakan koin 4, 1, 1 (3 koin) padahal seharusnya solusi optimum dilakukan pengambilan koin 3 sebanyak dua kali.

Peran Anggota Kelompok:
1. Muhammad Farhan Khairullah: Membuat Laporan, Membuat Video Dokumentasi, Membantu Muiz dalam menyempurnakan kode. 2. Muhammad Rifai Ageng Pambudi: Membuat Laporan, Membuat Video, memberi ide untuk menyempurnakan kode. 3. Muiz Afif Mirza Lindu Aji: Membuat kode python, dan laporan.