Greedy

m.Syahdan.rifaldi Isfandiar

Definisi

- *Greedy=Tamak/Rakus*
- *Algoritma greedy merupakan metode untuk memecahkan persoalan optimasi.*
- *Persoalan optimasi (optimization problems):*

1. *Maksimasi (maximization)*
2. *Minimasi (minimization)*

Definisi

- Algoritma *greedy* adalah algoritma yang memecahkan masalah langkah per langkah, pada setiap langkah:

  1. Ambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan (prinsip "*take what you can get now!*")

  2. Berharap bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

PENERAPAN GREEDY ALGORITHM

Contoh persoalan optimasi:

(Masalah Penukaran Uang): Diberikan uang senilai RP *36*. Tukar dengan pecahan uang yang ada. Berapa jumlah minimum lembar uang yang diperlukan untuk penukaran tersebut?
Contoh 1: lembaran uang yang tersedia RP 1,RP 5,RP 10,RP 20

PENERAPAN GREEDY ALGORITHM

Contoh persoalan optimasi:

(Masalah Penukaran Uang): Diberikan uang senilai RP *36*. Tukar dengan pecahan uang yang ada. Berapa jumlah minimum lembar uang yang diperlukan untuk penukaran tersebut?
Contoh 1: lembaran uang yang tersedia RP 1,RP 5,RP 10,RP 20

- Penyelesaian
- Uang senilai RP 36 dapat ditukar dengan banyak cara berikut:
  36 = 1 + 1 + … + 1 (36 lembar uang)
  36 = 5 + 5 + 5 + 5 + 10 + 5 + 1 (7 lembar uang)
  36 = 10 + 10 + 10 + 5 + 1 (5 lembar uang)
  … dst
- Minimum: 36 = 20 + 10 + 5 + 1 (4 lembar uang)

PENERAPAN GREEDY ALGORITHM

- Tinjau masalah penukaran uang:
  Strategi *greedy*:

  Pada setiap langkah, pilih lembaran uang dengan nilai terbesar dari himpunan uang yang tersisa.
- Aturan :
- Misal: uang RP 36, lembaran yang tersedia:RP 1,RP 5,RP 10, dan RP 20

PENERAPAN GREEDY ALGORITHM

- Tinjau masalah penukaran uang:
  Strategi *greedy*:

  Pada setiap langkah, pilih lembaran uang dengan nilai terbesar dari himpunan uang yang tersisa.
- Aturan :
- Misal: uang RP 36, lembaran yang tersedia:RP 1,RP 5,RP 10, dan RP 20

  Langkah 1: pilih 1 lembar uang RP 20 (Total =RP 20)

  Langkah 2: pilih 1 lembar uang RP 10 (Total = RP 20 + RP 10 = RP 30)

PENERAPAN GREEDY ALGORITHM

- Tinjau masalah penukaran uang:
  Strategi *greedy*:

  Pada setiap langkah, pilih lembaran uang dengan nilai terbesar dari himpunan uang yang tersisa.
- Aturan :
- Misal: uang RP 36, lembaran yang tersedia:RP 1,RP 5,RP 10, dan RP 20

  Langkah 1: pilih 1 lembar uang RP 20 (Total =RP 20)

  Langkah 2: pilih 1 lembar uang RP 10 (Total = RP 20 + RP 10 = RP 30)

  Langkah 3: pilih 1 lembar uang RP 5 (Total = RP 20 + RP 10 + RP 5 = RP 35)

  Langkah 4: pilih 1 lembar uang RP 1 (Total =RP 20 + RP 10 + RP 5 + RP 1 = RP 36)
- Solusi: Jumlah lembar uang minimum = 4 lembar (solusi optimal!)

PENERAPAN GREEDY ALGORITHM

```
def penukaran_uang(jumlah_uang):
    daftar_koin = [1000, 500, 200, 100, 50, 20, 10, 5, 1]  # Koin yang tersedia dalam denominasi
    solusi = []  # Himpunan untuk menyimpan koin-koin yang digunakan

    for koin in daftar_koin:
        while jumlah_uang >= koin:
            # Ambil sebanyak mungkin koin tersebut
            jumlah_koin = jumlah_uang // koin
            solusi.append((koin, jumlah_koin))
            jumlah_uang -= koin * jumlah_koin

    return solusi

# Contoh penggunaan
jumlah_uang = 1234  # Jumlah uang yang akan ditukar
hasil_tukar = penukaran_uang(jumlah_uang)
print(hasil_tukar)
```

kekurangan GREEDY ALGORITHM

1. Algoritma *greedy* tidak beroperasi secara menyeluruh terhadap semua alternatif solusi yang ada (sebagaimana pada metode *exhaustive search)*.
2. Terdapat beberapa fungsi SELEKSI yang berbeda, sehingga kita harus memilih fungsi yang tepat jika kita ingin algoritma menghasilkan solusi optimal.

- Optimum global belum tentu merupakan solusi optimum (terbaik), tetapi *sub-optimum*.
- Jadi, pada permasalahan tertentu algoritma *greedy* tidak selalu dapat memberikan solusi yang optimal.

KESIMPULAN

- Jika jawaban terbaik mutlak tidak diperlukan, maka algoritma *greedy* sering berguna untuk menghasilkan solusi perkiraan (*approximation*) yang lebih cepat dan mudah, dibandingkan menggunakan algoritma yang lebih rumit untuk menghasilkan solusi yang pasti.

- Bila algoritma *greedy* optimum, maka keoptimalannya itu dapat dibuktikan secara matematis