Algoritma greedy (serakah) adalah salah satu pendekatan dalam pemecahan masalah optimisasi yang bertujuan untuk menemukan solusi yang optimal dengan mengambil keputusan yang terbaik pada setiap langkah secara bertahap tanpa mempertimbangkan konsekuensi jangka panjang dari keputusan tersebut. Pendekatan ini bekerja dengan prinsip memilih langkah terbaik yang tersedia saat itu, dengan harapan bahwa serangkaian keputusan lokal yang optimal akan mengarah pada solusi global yang optimal.

Berikut adalah penjelasan detail tentang konsep dasar algoritma greedy dan bagaimana algoritma ini bekerja:

**Konsep Dasar Algoritma Greedy**

1. **Memilih Langkah Terbaik Secara Lokal:**
   * Pada setiap langkah dalam proses, algoritma greedy memilih opsi yang terlihat paling baik atau paling menguntungkan saat itu, tanpa memikirkan implikasi jangka panjang.
2. **Irreversible Decisions:**
   * Keputusan yang diambil pada setiap langkah bersifat tetap dan tidak dapat diubah di kemudian hari. Ini berarti bahwa algoritma tidak mundur atau mengubah keputusan sebelumnya.
3. **Sifat Masalah:**
   * Algoritma greedy bekerja dengan baik pada masalah yang memiliki properti tertentu, seperti masalah yang dapat dibagi menjadi submasalah independen atau masalah yang mematuhi kriteria greedy-choice (pilihan serakah).
4. **Langkah-langkah umun dalam algoritma greedy**

 **Inisialisasi:**

* Mulai dari keadaan awal atau kondisi dasar.

 **Iterasi:**

* Pada setiap langkah, pilih opsi terbaik yang tersedia menurut kriteria tertentu.
* Tambahkan opsi yang dipilih ke solusi sementara.

 **Kondisi Terminasi:**

* Ulangi langkah iterasi sampai kondisi terminasi tercapai, yaitu ketika semua elemen telah diproses atau ketika solusi lengkap telah ditemukan.

 **Output:**

* Hasilkan solusi yang dibangun dari pilihan-pilihan lokal yang telah diambil.

### 3. Contoh Masalah yang Cocok untuk Algoritma Greedy

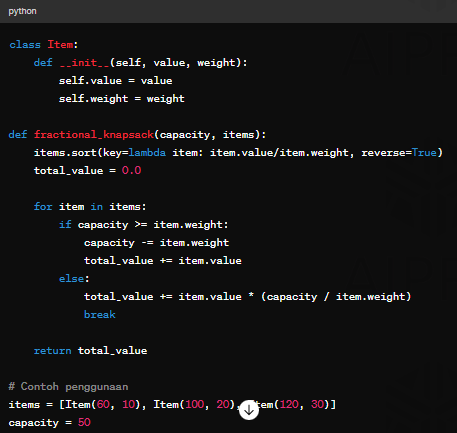
1. **Masalah Knapsack (Fractional Knapsack):**
   * Diberikan sejumlah item, masing-masing dengan berat dan nilai, tentukan kombinasi item yang dapat dimasukkan ke dalam knapsack (tas) dengan kapasitas maksimum untuk memaksimalkan nilai total. Pada versi fractional, item dapat dibagi-bagi.
2. **Masalah Pemilihan Aktivitas:**
   * Diberikan sejumlah aktivitas dengan waktu mulai dan waktu selesai, pilih maksimal aktivitas yang tidak tumpang tindih.
3. **Pohon Rentang Minimum (Minimum Spanning Tree - MST):**
   * Algoritma seperti Prim dan Kruskal digunakan untuk menemukan pohon rentang minimum dari graf yang terhubung dan berberat.

### 4. Contoh Implementasi Algoritma Greedy: Masalah Knapsack Fractional

Misalkan kita memiliki item dengan berat dan nilai berikut:

* Item 1: Berat = 10, Nilai = 60
* Item 2: Berat = 20, Nilai = 100
* Item 3: Berat = 30, Nilai = 120

Kapasitas knapsack adalah 50. Algoritma greedy akan memilih item dengan rasio nilai per berat terbesar terlebih dahulu.



langkah2 pada algoritma pemograman diatas adalah

* Membuat 2 klasifikasi item value dengan item berat
* membuat fungsi dengan 2 parameter diatas yaitu (capacity dan items)
* Langkah pertama parameter item di sorting menggunakan library item (pengurutan)
* Mendifinisakan vataibel total\_value = 0
* Melakukan iterasi (looping ) setiap item dengan kondisi

1. Jika Variable capacity lebih besar sama dengan item.weight (berat item)

maka capacity – item.weight dan total\_value + item.value

1. Jika capacity Tidak lebih besar (artinya lebih kecil) dari item weight

maka total value = total value + item.value x (capacity/item.weight)

* Data yang dikembalikan adalah nilai total (total\_value)

Jadi jika dilakukan perhitungan analitik dari input data diatas seperti ini

perhitungan analitik

Item 1: nilai = 60, berat = 10, total = 60 / 10 = 6.0

Item 2: nilai = 100, berat = 20, total = 100 / 20 = 5.0

Item 3: nilai = 120, berat = 30, total = 120 / 30 = 4.0

Kemudian diurutkan berdasarkan berat

Kemudai masuk keiterasi (looping) sesuai dengan kondisi:

Diketahu Kapasitas awal: 50

Ambil item 1: kapasitas tersisa = 50 - 10 = 40, nilai total = 60

Ambil item 2: kapasitas tersisa = 40 - 20 = 20, nilai total = 60 + 100 = 160

Ambil sebagian dari item 3: kapasitas tersisa = 20, nilai yang diambil = 120 \* (20 / 30) = 80, nilai total = 160 + 80 = 240

Maka nilai maksimun yg didapat 240