LAPORAN TUGAS BESAR

“Perbandingan Efisiensi Algoritma Iteratif dan Algoritma Rekursif dalam Penyelesaian Masalah Knapsack"

Disusun oleh :

Nevan Nabiil Firmansyah Har (103012300273)

Muhammad Nayubi Putra Adiva (103012300469)

IF-47-08

S1 Informatika

Analisis Kompleksitas Algoritma

2024

1. Deskripsi Studi Kasus

Permasalahan Knapsack adalah permasalahan klasik pada masalah optimasi di mana seseorang harus memilih kombinasi barang yang tepat untuk dimasukkan ke dalam knapsack (tas/kantong) yang memiliki kapasitas terbatas.

Sumber :https://www.researchgate.net/figure/Gambar-1-Ilustrasi-Knapsack-Knapsack-problem-secara-secara-matematis-dapat-ditulis_fig1_324078835

Dalam masalah ini, terdapat 2 atribut tertentu, yaitu berat dan nilai. Berat menunjukkan seberapa banyak ruang yang diambil barang di dalam knapsack, sementara nilai menunjukkan seberapa berharga barang tersebut. Keputusan ini harus diambil dengan tujuan untuk memaksimalkan total nilai dari item yang dimasukkan ke dalam knapsack.

2. Deskripsi Algoritma

Untuk menyelesaikan masalah Knapsack, Terdapat 2 algoritma yang dipilih, yaitu algoritma iteratif (Greedy) dan algoritma rekursif.

A. Algoritma Iteratif

Algoritma ini menyelesaikan masalah Knapsack menggunakan pendekatan **Greedy** dengan memprioritaskan barang berdasarkan **rasio nilai terhadap berat (value-to-weight ratio)**. Tujuannya adalah untuk memaksimalkan nilai total barang yang dimasukkan ke dalam knapsack dengan kapasitas terbatas pada kedua atribut, yaitu ukuran dan berat,

dengan cara memilih barang yang memberikan nilai tertinggi per unit berat terlebih dahulu.

a. Kodingan

```python
def FungsiRanselGreedyIteratif(daftarBarang, kapasitasUkuran, kapasitasBerat):
    for brg in daftarBarang:
        brg.rasio = brg.nilai / brg.berat

    daftarBarang.sort(key=lambda brg: brg.rasio, reverse=True)

    totalNilai = 0.0
    totalUkuran = 0
    totalBerat = 0

    for brg in daftarBarang:
        if totalUkuran + brg.ukuran <= kapasitasUkuran and totalBerat + brg.berat <= kapasitasBerat:
            totalNilai += brg.nilai
            totalUkuran += brg.ukuran
            totalBerat += brg.berat
        else:
            sisaUkuran = kapasitasUkuran - totalUkuran
            sisaBerat = kapasitasBerat - totalBerat
            jumlahDiambil = min(sisaUkuran, sisaBerat) / brg.berat
            totalNilai += jumlahDiambil * brg.nilai
            break

    return totalNilai
```

B. Algoritma Rekursif

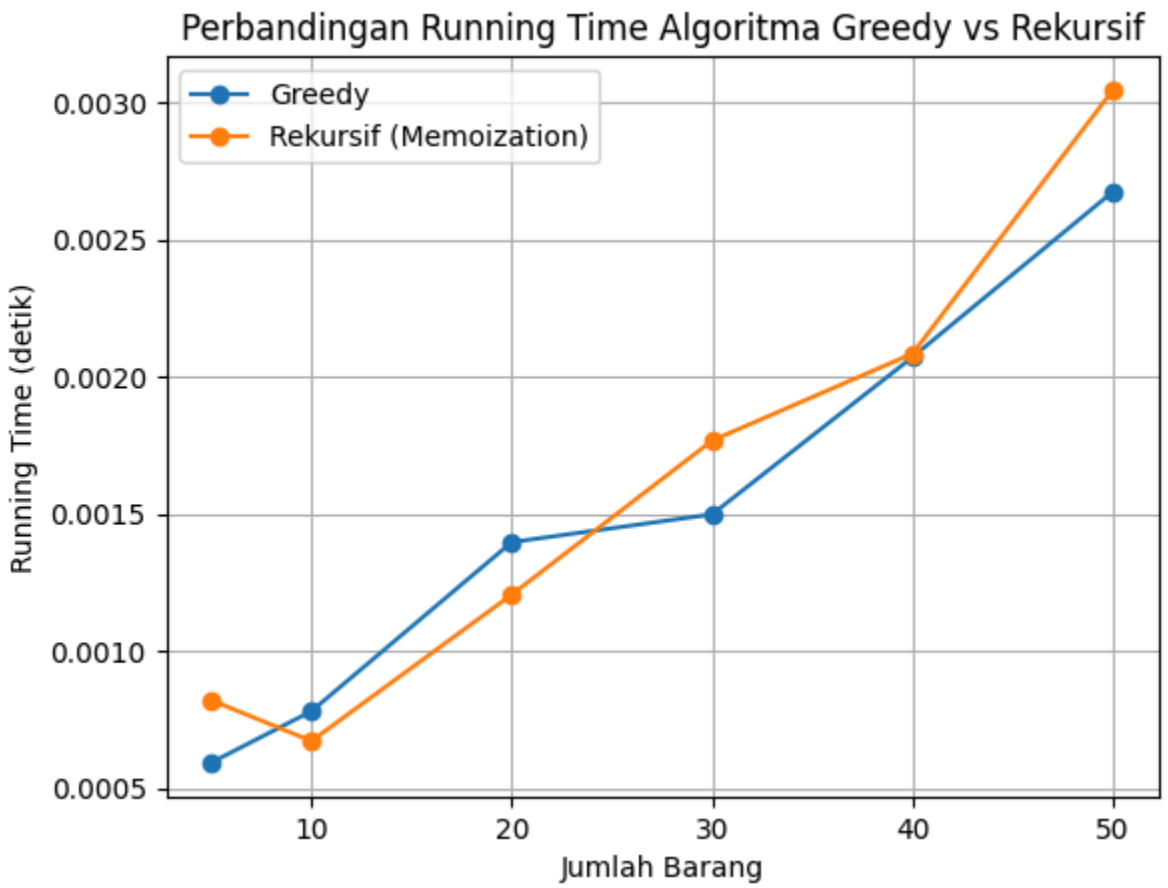
Algoritma ini menyelesaikan masalah Knapsack menggunakan pendekatan **Rekursif** yang diperkuat dengan **memoization**. Tujuannya adalah untuk memaksimalkan nilai total barang yang dapat dimasukkan ke dalam knapsack dengan batasan kapasitas ukuran dan berat, dengan cara membagi masalah menjadi sub-masalah lebih kecil dan menyimpan hasil perhitungan sub-masalah tersebut agar tidak dihitung ulang.

a. Kodingan

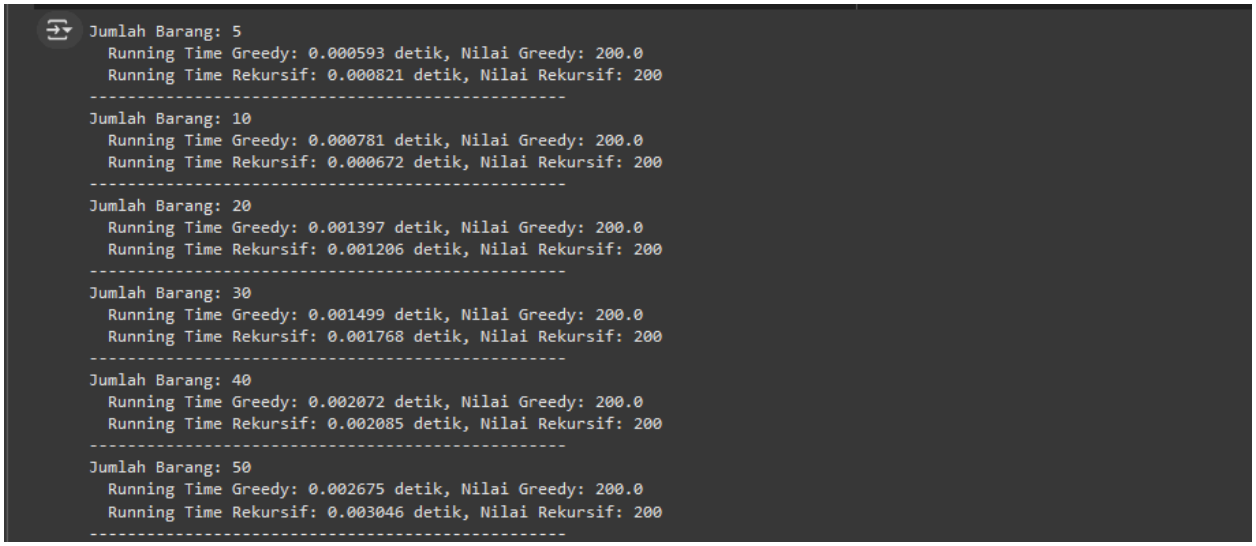
```python
def FungsiRansel2DRekursif(daftarBarang, i, u, b):
    if i < 0 or u <= 0 or b <= 0:
        return 0

    if memo[i][u][b] != UNDEF:
        return memo[i][u][b]

    opsiTidakAmbil = FungsiRansel2DRekursif(daftarBarang, i - 1, u, b)

    opsiAmbil = 0
    if daftarBarang[i].ukuran <= u and daftarBarang[i].berat <= b:
        opsiAmbil = FungsiRansel2DRekursif(daftarBarang, i - 1, u - daftarBarang[i].ukuran, b - daftarBarang[i].berat) + daftarBarang[i].nilai

    hasilTerbaik = max(opsiTidakAmbil, opsiAmbil)

    memo[i][u][b] = hasilTerbaik

    return hasilTerbaik
```

3. Perbandingan Algoritma

Grafik:



Output:



4. Analisis perbandingan kedua algoritma

a. Running Time
   i. Greedy(Iteratif)

      Algoritma Iteratif (Greedy) memiliki running time yang cenderung lebih cepat dibandingkan dengan Algoritma Rekursif, bahkan untuk jumlah barang yang lebih banyak. Ini karena algoritma greedy hanya melakukan

beberapa operasi sederhana, seperti perhitungan rasio nilai/berat dan pengurutan barang, yang tidak memerlukan banyak waktu.

ii. Rekursif

Algoritma Rekursif dengan memoization masih efisien, namun dibandingkan dengan algoritma Greedy, ia cenderung lebih lambat. Meskipun memoization mengurangi penghitungan berulang dengan menyimpan hasil dari submasalah yang sudah dihitung sebelumnya, running time tetap meningkat seiring bertambahnya jumlah barang. Hal ini disebabkan oleh kompleksitas rekursi yang tinggi dan banyaknya submasalah yang harus dihitung dan disimpan dalam memoization

b. Kompleksitas Waktu

i. Iteratif (Greedy)

Kompleksitas waktu dari algoritma greedy adalah **O(nlogn)** karena pengurutan barang berdasarkan rasio nilai/berat.

ii. Rekursif

Kompleksitas waktu dari algoritma rekursif adalah **O(n*kapasitas_ukuran*kapasitas_berat)** karena algoritma memecah masalah menjadi sub-masalah berdasarkan barang dan kapasitas tas (ukuran dan berat), dan memoization menghindari perhitungan ulang.

c. Nilai Maksimum yang Diperoleh

Algoritma Iteratif (Greedy) memilih barang berdasarkan rasio nilai/berat tertinggi dan menambahkannya ke dalam tas secara iteratif, tanpa mempertimbangkan apakah kombinasi barang yang dipilih akan menghasilkan solusi yang optimal untuk kapasitas tas yang tersedia. Dalam beberapa kasus, meskipun algoritma ini cepat dan efisien (dengan waktu eksekusi yang lebih singkat), ia mungkin tidak menghasilkan solusi terbaik, karena tidak mempertimbangkan semua kemungkinan kombinasi barang yang dapat dimasukkan.

Di sisi lain, algoritma Rekursif dengan memoization mempertimbangkan semua kemungkinan kombinasi barang yang dapat dimasukkan ke dalam tas, baik dari sisi kapasitas ukuran maupun kapasitas berat, untuk menemukan solusi yang optimal. Meskipun lebih lambat dibandingkan

Algoritma Iteratif (Greedy) (dengan waktu eksekusi yang lebih lama, terutama untuk jumlah barang yang lebih banyak), algoritma ini mampu memberikan solusi terbaik karena mempertimbangkan semua sub-masalah dengan menyimpan hasil perhitungan yang telah dilakukan sebelumnya (memoization), sehingga menghindari penghitungan berulang.

5. Kesimpulan

Berdasarkan hasil pengujian terhadap kedua algoritma, dapat disimpulkan bahwa Algoritma Iteratif (Greedy) memiliki running time yang lebih cepat dibandingkan dengan Algoritma Rekursif (Memoization). Algoritma Iteratif bekerja dengan cara memilih barang berdasarkan rasio nilai/berat tertinggi terlebih dahulu, yang membuatnya lebih efisien dalam waktu eksekusi. Namun, algoritma ini tidak selalu menghasilkan solusi optimal, karena tidak mengevaluasi semua kemungkinan kombinasi barang berdasarkan kapasitas yang tersedia.

Sementara, Algoritma Rekursif (Memoization) memastikan solusi yang optimal dengan mempertimbangkan setiap kombinasi barang yang dapat dimasukkan ke dalam tas, tanpa melampaui kapasitas ukuran dan berat. Meskipun lebih lambat akibat melibatkan lebih banyak perhitungan dan penyimpanan sub-masalah, algoritma ini menjamin hasil yang optimal dalam setiap kasus.

Secara keseluruhan, Algoritma Iteratif lebih cocok digunakan ketika kecepatan eksekusi lebih diutamakan, meskipun tidak selalu memberikan solusi optimal. Sementara itu, Algoritma Rekursif (Memoization) lebih tepat digunakan ketika kualitas solusi yang optimal lebih penting, meskipun dengan sedikit pengorbanan dalam waktu eksekusi.

6. Referensi

https://www.researchgate.net/figure/Gambar-1-Ilustrasi-Knapsack-Knapsack-problem-secara-secara-matematis-dapat-ditulis_fig1_324078835

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Makalah/Makalah-IF2211-2018-039.pdf

7. Poster

https://drive.google.com/drive/folders/1S9qlcHK7d4W8qYYMvHdaXjw3VK8IAIH0?usp=sharing