LAPORAN PENERAPAN ALGORITMA BRUTE FORCE DAN BRANCH AND BOUND PADA KNAPSACK

DISUSUN UNTUK MEMENUHI TUGAS BESAR MATA KULIAH

STRATEGI ALGORITMA

Disusun Oleh :

RAYHAN ADRIAN FADLILLAH - 1301210331
R. ADICONDRO YUSUF HENDRATMO - 1301213152
MUHAMMAD FATIH YUMNA LAJUWIRDI LIRRAHMAN - 1301213389

IF-45-05

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM BANDUNG

2023

ABSTRAK

Knapsack merupakan permasalahan optimasi klasik yang memiliki aplikasi dalam kehidupan sehari-hari. Dalam penyelesaian permasalahan knapsack, terdapat banyak algoritma yang dapat menghasilkan solusi yang optimal. Beberapa diantaranya adalah algoritma brute force dan branch and bound. Pada pendekatan Brute Force, semua kombinasi kemungkinan item akan dievaluasi secara berurutan untuk mencari solusi optimal. Namun, metode ini tidak efisien ketika jumlah item yang dimiliki sangat besar. Oleh karena itu, dilakukan penerapan metode Branch and Bound untuk meningkatkan efisiensi algoritma.

Metode Branch and Bound memanfaatkan teknik pemotongan dan pembatasan ruang pencarian yang tidak relevan, dengan mempertimbangkan batasan kapasitas knapsack dan estimasi nilai maksimal yang dapat dicapai. Algoritma ini secara rekursif membagi masalah menjadi submasalah yang lebih kecil dan mengeliminasi cabang pencarian yang tidak mungkin menghasilkan solusi optimal.Dalam laporan ini, akan dijelaskan langkah-langkah penerapan kedua algoritma tersebut pada masalah Knapsack. Dilakukan analisis perbandingan kinerja antara Brute Force dan Branch and Bound dalam hal waktu eksekusi dan efisiensi solusi yang dihasilkan.

Penelitian ini akan menggunakan contoh kasus dengan ukuran masalah yang bervariasi untuk menguji dan membandingkan kinerja kedua algoritma. Metrik yang akan digunakan dalam evaluasi adalah waktu eksekusi dan kualitas solusi yang dihasilkan. Data hasil eksperimen akan dianalisis dan dibandingkan secara kuantitatif.Diharapkan bahwa laporan ini akan memberikan pemahaman yang lebih baik tentang penerapan algoritma Brute Force dan Branch and Bound pada masalah Knapsack. Selain itu, hasil penelitian ini juga dapat memberikan wawasan tentang kinerja dan keefektifan masing-masing algoritma dalam menyelesaikan masalah optimisasi kombinatorial. Penelitian ini memiliki potensi untuk memberikan kontribusi pada pengembangan algoritma penyelesaian masalah Knapsack yang lebih efisien dan efektif di masa depan.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Knapsack, juga dikenal sebagai Ransel, adalah sebuah masalah optimisasi kombinatorial yang sering dihadapi dalam ilmu komputer dan matematika. Dalam masalah ini, kita memiliki sebuah knapsack (tas) dengan kapasitas terbatas dan kumpulan item dengan nilai dan bobot tertentu. Tujuannya adalah memilih kombinasi item yang akan dimasukkan ke dalam knapsack sehingga total nilai item yang dipilih adalah maksimum, tanpa melampaui kapasitas knapsack.

Dalam masalah Knapsack, setiap item memiliki dua atribut utama: nilai dan bobot. Nilai mewakili manfaat atau keuntungan yang diberikan oleh item tersebut, sedangkan bobot mencerminkan penggunaan sumber daya yang diperlukan. Knapsack mewakili sumber daya atau batasan yang perlu dipertimbangkan, seperti kapasitas berat maksimum yang dapat ditampung.

Untuk menyelesaikan masalah Knapsack, terdapat beberapa pendekatan algoritma yang dapat digunakan, termasuk algoritma brute force, algoritma greedy, dynamic programming dan branch and bound. Setiap pendekatan memiliki kelebihan dan kelemahan masing-masing tergantung pada karakteristik masalah yang dihadapi.

Masalah Knapsack memiliki banyak aplikasi praktis, seperti perencanaan produksi, pemotongan bahan, perencanaan investasi, dan pengaturan jadwal. Dalam konteks ini, Knapsack dapat membantu dalam pengambilan keputusan yang efisien untuk mengoptimalkan penggunaan sumber daya terbatas dan mencapai hasil maksimum.

Dalam penelitian dan pengembangan algoritma, Knapsack sering digunakan sebagai contoh masalah kompleksitas yang dapat diukur. Dengan memahami dan mempelajari Knapsack, kita dapat mengembangkan algoritma optimasi yang lebih canggih dan memperluas pemahaman kita tentang masalah optimisasi kombinatorial secara umum.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dituliskan diatas, maka rumusan masalah :

1. Bagaimana penerapan algoritma Brute Force dan Branch and Bound sebagai pembelajaran dalam pengambilan solusi dan langkah yang optimal pada Knapsack?
2. Algoritma mana yang lebih efektif dalam mencapai hasil optimal dalam Knapsack?

1.2 Tujuan Penelitian

Berdasarkan latar belakang yang telah dituliskan diatas, maka rumusan masalah :

1. Penelitian ini bertujuan untuk mengimplementasikan algoritma Brute Force dan Branch and Bound untuk mencari solusi optimal dari masalah Knapsack. Dengan menerapkan algoritma ini, kita dapat memahami cara kerjanya dan mendapatkan pemahaman yang lebih baik tentang kompleksitas waktu yang terlibat dalam pendekatan Brute Force dan Branch and Bound.
2. Membandingkan kinerja algoritma Brute Force dan Branch and Bound dalam menyelesaikan masalah Knapsack. Dalam hal ini, kita akan menganalisis waktu eksekusi dari kedua algoritma serta kualitas solusi yang dihasilkan oleh masing-masing pendekatan.

BAB II
DASAR TEORI

2. 1 Knapsack

Knapsack merupakan varian populer dari permasalahan knapsack dimana dalam 0/1 knapsack, tiap barang dalam himpunan barang hanya dapat diambil sebanyak satu buah (1) atau tidak diambil sama sekali (0). Secara lebih formal, permasalah 0/1 knapsack adalah sebagai berikut: diberikan n buah barang yang dinomori dari 1 sampai n, dimana masing masing barang memiliki berat sebesar wi dan nilai vi, dengan berat maksimum yang dapat diambil sebesar W maka akan dicari himpunan solusi X = (x1, x2, …, xn) dimana xi akan bernilai 1 jika barang ke-i diambil dan 0 jika tidak diambil sama sekali. Solusi yang dicari merupakan solusi yang memaksimalkan total nilai objektif yaitu

$$\sum_{i=1}^{n} v_i x_i$$

Dengan syarat memiliki total bobot kurang dari W, atau dinotasikan sebagai berikut.

$$\sum_{i=1}^{n} w_i x_i \leq W$$

2.2 Brute Force

Algoritma brute force atau sering kali disebut sebagai exhaustive search merupakan salah satu pendekatan dalam penyelesaian masalah yang paling sederhana. Algoritma brute force secara sistematis mencoba seluruh kemungkinan pada himpunan kandidat solusi dan melakukan pengecekan apakah kandidat tersebut memenuhi kriteria solusi permasalahan. Contohnya adalah dalam pencarian nilai dari a^b, algoritma brute force akan mengalikan a sebanyak b kali untuk mendapatkan hasil, padahal terdapat banyak cara lainnya yang dapat mendapatkan hasil dengan lebih efisien.

Algoritma brute force sering kali dipakai karena sifatnya yang sangat sederhana dan mudah diimplementasikan sehingga dapat digunakan sebagai implementasi awal untuk

mengecek apakah solusi dapat ditemukan. Selain itu, algoritma brute force juga sering digunakan untuk keperluan benchmarking algoritma sebagai base solution untuk dibandingkan dengan algoritma lainnya.

Dalam permasalahan 0/1 knapsack algoritma brute force memiliki langkah penyelesaian sebagai berikut.

1. Enumerasi himpunan kandidat solusi, yaitu semua kemungkinan pengambilan barang
2. Hitung keuntungan dari setiap kandidat solusi, dimana jika total bobot dari kandidat solusi melebihi batas bobot, solusi untuk kandidat tersebut tidak diperhitungkan
3. Pilih kandidat solusi yang menghasilkan total keuntungan atau nilai terbesar

Untuk memudahkan pemahaman, berikut adalah visualisasi search tree dalam penyelesaian 0/1 knapsack pada 3 barang menggunakan algoritma brute force

Untuk memudahkan pemahaman, berikut adalah visualisasi search tree dalam penyelesaian 0/1 knapsack pada 3 barang menggunakan algoritma brute force. Dapat dilihat bahwa algoritma brute force tidak melakukan langkah optimasi dalam pemilihan kandidat solusi. Bisa jadi bobot dari dua barang pertama yang dipilih sudah melebihi total bobot maksimal namun algoritma brute force tetap melanjutkan enumerasi solusi ke barang ke-3.

2.3 Branch and Bound

Algoritma branch and bound merupakan salah satu varian algoritma untuk menyelesaikan permasalahan optimasi, yaitu upaya dalam meminimalkan atau memaksimalkan sebuah fungsi objektif dengan suatu batasan persoalan. Algoritma branch and bound merupakan algoritma yang lebih mangkus secara kompleksitas waktu dibandingkan dengan metode brute force karena pada branch and bound, setiap kandidat solusi akan dievaluasi solusi sementara dimana jika solusi sementara tidak mengarah ke solusi optimal maka komponen lanjutan dari solusi sementara tersebut tidak dilanjutkan dan kandidat solusi itu dimatikan.

Secara urut, algoritma global dari branch and bound adalah sebagai berikut.

1. Masukkan simpul akar ke antrian, jika simpul akar merupakan solusi maka solusi telah didapatkan dan algoritma dihentikan
2. Jika antrian kosong, maka solusi tidak ditemukan dan algoritma dihentikan

3. Jika antrian tidak kosong, maka akan dipilih simpul dengan nilai beban terkecil dari antrian dan jika terdapat beberapa simpul dengan nilai minimum, maka dapat dipilih simpul sembarang yang memiliki nilai minimum
4. Jika simpul yang dipilih merupakan simpul solusi, maka solusi sudah ditemukan dan algoritma dihentikan. Jika tidak, maka seluruh anak simpul akan dibangkitkan dan jika simpul tersebut tidak memiliki anak, maka kembali ke langkah ke-2
5. Untuk setiap anak dari simpul yang dipilih, hitung nilai beban dan masukkan simpul tersebut ke dalam antrian
6. Kembali ke langkah 2

Dalam permasalahan 0/1 knapsack, setiap barang akan dihitung bound value-nya, yaitu nilai maksimum dari keuntungan yang dapat diraih. Jika bound value dari barang tersebut kurang dari nilai keuntungan maksimal sementara, maka pencarian solusi yang mengarah pada barang tersebut akan dihentikan karena sudah dapat dipastikan tidak mengarah ke solusi optimal.

BAB III
IMPLEMENTASI DAN ANALISIS

3.1 Brute Force *(Exhaustive Search)*

3.1.1 Permasalahan

Anda memiliki sebuah knapsack dengan kapasitas maksimum 15 kg. Anda memiliki kumpulan item berikut dengan nilai dan bobotnya:

Item	Value	Bobot
A	$10	2 kg
B	$12	4 kg
C	$8	6 kg
D	$15	9 kg

3.1.2 Penyelesaian

Himpunan Bagian	Total Bobot	Total Keuntungan
{}	0	$0
{1}	2	$10
{2}	4	$12
{3}	6	$8
{4}	9	$9
{1,2}	6	$22
{1,3}	8	$18

{1,4}	11	$25
{2,3}	10	$20
{2,4}	13	$27
{3,4}	15	$23
{1,2,3}	12	$30
{1,2,4}	15	$37
{1,3,4}	17	$33 (Tidak Layak)
{2,3,4}	19	$35 (Tidak Layak)
{1,2,3,4}	21	$45 (Tidak Layak)

Himpunan bagian yang memberikan keuntungan maksimum adalah {1,2,4} dengan total keuntungan $37.

3.1.3 Algoritma

```python
import time

def knapsack_brute_force(values, weights, capacity):
    num_items = len(values)
    max_value = 0
    best_combination = []

    for i in range(2**num_items):
        combination = []
        total_value = 0
        total_weight = 0

        for j in range(num_items):
            if  i >> j) & 1:
              ( combination.append(j)
                total_value += values[j]
                total_weight += weights[j]

        if total_weight <= capacity and total_value > max_value:
            max_value = total_value
            best_combination = combination

    return max_value, best_combination


# Example usage
values =  10, 12, 8, 15]
weights ∉  2, 4, 6, 9]
capacity ∉ 15

start_time = time.time()
max_value, best_combination = knapsack_brute_force(values, weights, capacity)
end_time = time.time()

execution_time = end_time - start_time

print("Best combination of items:", best_combination)
print("Total value:", max_value)
print("Execution time:", execution_time, "seconds")
```

3.1.4 Kompleksitas

Kompleksitas waktu algoritma Brute Force pada Knapsack adalah n untuk objek, banyaknya himpunan bagian yang harus dievaluasi adalah 2^n.Perhitungan total bobot setiap himpunan bagian dapat dikerjakan dalam O(n). Maka kompleksitas algoritma exhaustive search untuk permasalahan knapsack adalah O(n.2^n)

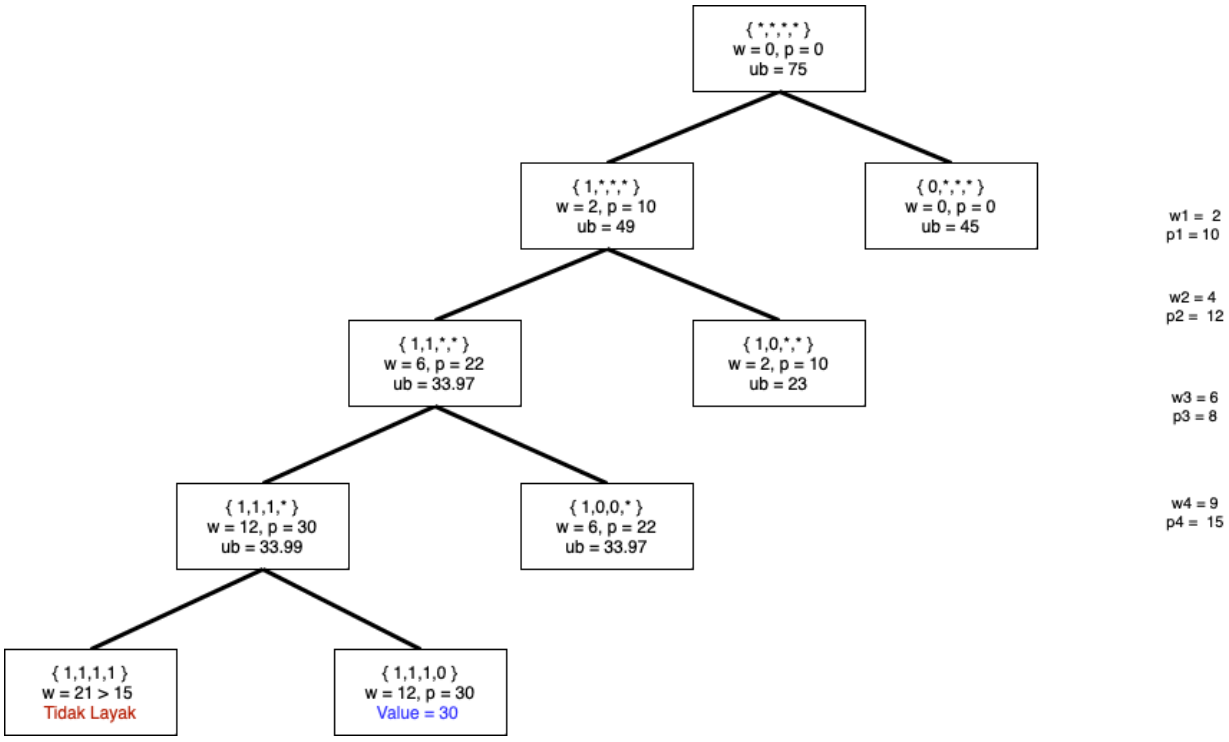
3.2 Branch and Bound

3.2.1 Permasalahan

Anda memiliki sebuah knapsack dengan kapasitas maksimum 15 kg. Anda memiliki kumpulan item berikut dengan nilai dan bobotnya:

Item	Value	Bobot
A	$10	2 kg
B	$12	4 kg
C	$8	6 kg
D	$15	9 kg

3.2.2 Penyelesaian



3.2.3 Algoritma

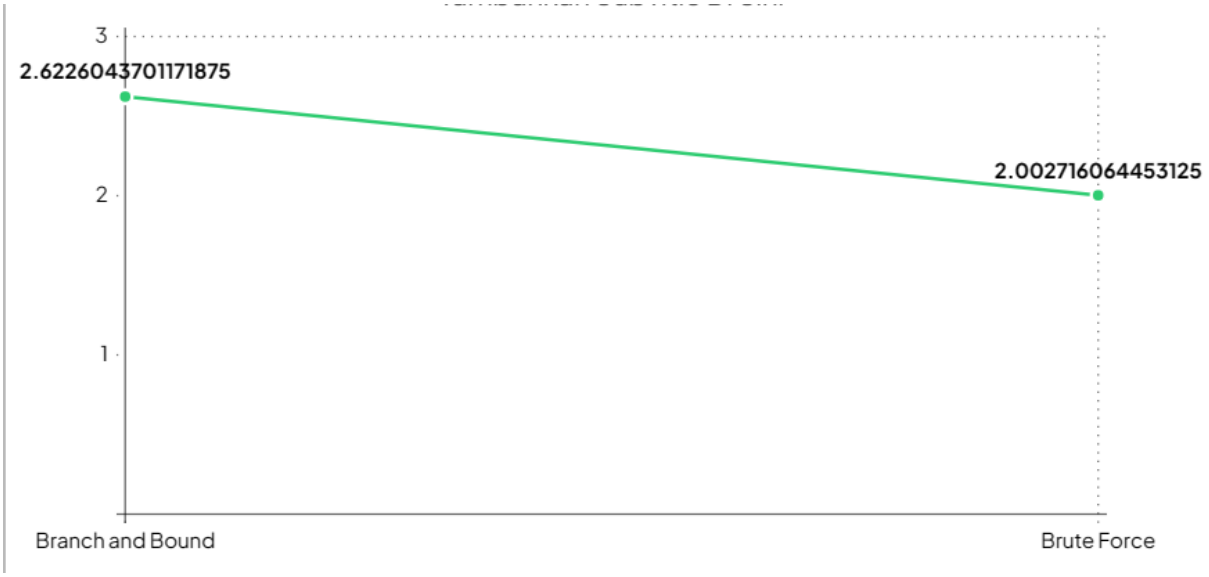
```python
import time

class Item:
    def __init__(self, value, weight):
        self.value = value
        self.weight = weight
        self.ratio = value / weight


def knapsack_branch_and_bound(values, weights, capacity):
    num_items = len(values)
    items = []
    for i in range(num_items):
        items.append(Item(values[i], weights[i]))

    items.sort(key=lambda x: x.ratio, reverse=True)

    max_value = 0
    best_combination = []

    def branch_and_bound(node, current_value, current_weight):
        nonlocal max_value, best_combination

        if node < num_items and current_weight + items[node].weight <= capacity:
            current_value += items[node].value
            current_weight += items[node].weight

            if current_value > max_value:
                max_value = current_value
                best_combination =  i for i in range(node + 1)]
                                    [
            branch_and_bound(node + 1, current_value, current_weight)

            current_value -= items[node].value
            current_weight -= items[node].weight

            branch_and_bound(node + 1, current_value, current_weight)

    branch_and_bound(0, 0, 0)

    return max_value, best_combination


# Example usage
values =  10, 12, 8, 15]
weights ∉  2, 4, 6, 9]
capacity ∉ 15

start_time = time.time()
max_value, best_combination = knapsack_branch_and_bound(values, weights, capacity)
end_time = time.time()

execution_time = end_time - start_time

print("Best combination of items:", best_combination)
print("Total value:", max_value)
print("Execution time:", execution_time, "seconds")
```

3.2.4 Kompleksitas

Kompleksitas algoritma Branch and Bound pada Knapsack bervariasi tergantung pada berbagai faktor, termasuk ukuran masalah Knapsack dan struktur instance masalah. Namun, secara umum, kompleksitas waktu algoritma Branch and Bound pada Knapsack dapat dianggap sebagai O(2^N), dimana N adalah jumlah item yang tersedia dalam knapsack.

Selain itu, algoritma Branch and Bound juga melibatkan operasi pengurutan item berdasarkan rasio nilai-bobot. Kompleksitas pengurutan ini biasanya adalah O(N log N), dimana N adalah jumlah item.Dalam prakteknya, kompleksitas waktu algoritma Branch and Bound pada Knapsack dapat berbeda-beda tergantung pada faktor-faktor tersebut. Namun, secara umum, kompleksitas waktu algoritma Branch and Bound dapat dianggap sebagai O(2^N).

3.3 HASIL DAN ANALISIS



Dari hasil pengujian, didapatkan beberapa informasi penting. Pada kasus uji , yaitu kasus uji berukuran kecil, algoritma brute force meraih waktu eksekusi paling cepat. Hal ini sesuai dengan ekspektasi karena untuk jumlah barang yang sedikit, banyak komputasi yang dilakukan tidak terlalu banyak dan juga algoritma brute force tidak memiliki banyak pemrosesan tambahan seperti pada algoritma lain.

BAB IV

KESIMPULAN

Berdasarkan hasil running time, kesimpulan yang dapat ditarik adalah bahwa brute force memiliki running time yang lebih cepat daripada branch and bound dalam penyelesaian masalah knapsack. Namun, perlu diingat bahwa kesimpulan ini hanya berlaku dalam konteks spesifik dari eksperimen atau analisis yang dilakukan.

Dalam algoritma brute force, semua kemungkinan solusi dieksplorasi secara langsung. Ini berarti bahwa kompleksitas waktu brute force bergantung pada jumlah item yang ada dalam knapsack, dan kompleksitasnya adalah O(2^N), dimana N adalah jumlah item. Oleh karena itu, pada kasus-kasus dengan jumlah item yang sedikit atau ukuran masalah yang kecil, brute force dapat memberikan hasil lebih cepat karena tidak ada upaya pemotongan atau optimisasi yang dilakukan.

Di sisi lain, branch and bound menggunakan pendekatan yang lebih cerdas dengan melakukan pemotongan pada cabang-cabang pohon pencarian yang tidak mungkin menghasilkan solusi optimal. Dengan melakukan pemotongan ini, algoritma branch and bound dapat mengurangi jumlah simpul yang harus dieksplorasi, dan dalam beberapa kasus, ini dapat menghasilkan running time yang lebih baik daripada brute force. Namun, implementasi yang buruk atau kasus masalah yang khusus dapat mempengaruhi kinerja algoritma branch and bound, sehingga pada eksperimen tertentu, brute force dapat memberikan hasil yang lebih cepat.

Kesimpulan bahwa brute force memiliki running time tercepat dalam kasus yang diamati bukan berarti bahwa brute force selalu lebih baik daripada branch and bound. Dalam kasus knapsack dengan ukuran masalah yang lebih besar atau kompleksitas yang lebih tinggi, branch and bound cenderung lebih efisien karena mengurangi jumlah simpul yang harus dieksplorasi. Oleh karena itu, pemilihan algoritma tergantung pada karakteristik spesifik dari masalah knapsack yang dihadapi.

DAFTAR PUSTAKA

[1].http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Brute-Force-(2016).pdf

[2]http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Branch-&-Bound-(2018).pdf

[3]http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/20172018/Program-Dinamis-(2018).pdf

[4]http://standardwisdom.com/softwarejournal/2010/03/bangfor-the-buck-knapsacks-in-real-life/