LAPORAN HASIL PROYEK
DESAIN DAN ANALISIS ALGORITMA
PENCARIAN PROFIT TERBESAR PENJUALAN BATIK KE BERBAGAI NEGARA DENGAN MENGGUNAKAN ALGORITMA GREEDY DAN DIVIDE AND CONQUER

OLEH KELOMPOK 5:

Shinta Usaila Farachin	**: 23031554160**
Ikhrima Atusifah	**: 23031554181**
Arina Tri Yuni W. T.	**: 23031554203**

Kelas 2023A

UNIVERSITAS NEGERI SURABAYA

2024

PENDAHULUAN

Latar Belakang

Penyebaran produk batik Indonesia ke seluruh dunia terus mengalami perkembangan pesat, sejalan dengan meningkatnya minat pasar internasional terhadap produk tradisional yang sarat dengan nilai seni dan budaya. Untuk memperluas jangkauan pasar dan meningkatkan keuntungan, para pengusaha batik perlu mengoptimalkan strategi penjualannya sehingga dapat mencapai profit yang maksimal. Salah satu tantangan utama yang dihadapi adalah menetapkan harga jual yang tepat serta menentukan jumlah produksi yang sesuai untuk setiap negara yang menjadi sasaran pasar. Dalam hal ini, diperlukan pendekatan yang sistematis dalam merencanakan distribusi batik agar profit dapat dimaksimalkan.

Dalam upaya ini, penerapan algoritma komputasi seperti Greedy dan Divide and Conquer dapat menjadi solusi efektif untuk menemukan strategi penjualan yang paling optimal. Algoritma Greedy memfokuskan pada pengambilan keputusan lokal yang terbaik dengan harapan dapat menghasilkan solusi global yang memuaskan. Sementara itu, algoritma Divide and Conquer membagi masalah menjadi sub-masalah yang lebih kecil, menyelesaikannya secara terpisah, dan kemudian menggabungkan hasilnya untuk mendapatkan solusi menyeluruh. Kedua algoritma ini dapat digunakan dalam pencarian profit terbesar, dengan mempertimbangkan berbagai faktor seperti biaya, permintaan, dan harga jual di setiap negara.

Implementasi kedua algoritma ini diharapkan dapat memberikan pengusaha batik strategi distribusi yang efisien dan menguntungkan. Melalui simulasi dan analisis yang mendalam, diharapkan dapat dirumuskan solusi terbaik yang berpotensi meningkatkan profit penjualan batik di pasar internasional. Oleh karena itu, penelitian ini bertujuan untuk mengembangkan sebuah sistem yang memanfaatkan algoritma Greedy dan Divide and Conquer dalam mencari solusi optimal untuk mencapai profit tertinggi dalam penjualan batik di berbagai negara.

Tujuan

1. Mengetahui kode mencari profit terbesar penjualan batik ke berbagai negara menggunakan algoritma greedy
2. Mengetahui kode mencari profit terbesar penjualan batik ke berbagai negara menggunakan algoritma divide and conquer
3. Mengetahui perbandingan penggunaan algoritma greedy dan divide and conquer dalam mencari profit terbesar penjualan batik ke berbagai negara.

Manfaat

1. Optimalisasi keuntungan penjualan
2. Efisiensi pengelolaan sumber daya
3. Peningkatan daya saing di pasar internasional

Rumusan Masalah

1. Bagaimana code mencari profit terbesar penjualan batik ke berbagai negara menggunakan algoritma greedy?
2. Bagaimana code mencari profit terbesar penjualan batik ke berbagai negara menggunakan algoritma divide and conquer?
3. Bagaimana perbandingan penggunaan algoritma greedy dan divide and conquer dalam mencari profit terbesar penjualan batik ke berbagai negara?

METODE

Dalam projek ini, algoritma greedy dan divide and conquer diambil sebagai metode perbandingan untuk menyelesaikan masalah pencarian profit terbesar dari penjualan batik ke berbagai negara. Kedua algoritma ini diterapkan pada permasalahan yang identik, bertujuan untuk mengidentifikasi keunggulan dan kelemahan masing-masing dalam hal efisiensi waktu dan hasil akhir yang dicapai.

Algoritma greedy memanfaatkan prinsip pengambilan keputusan lokal yang dianggap optimal pada setiap langkah, tanpa mempertimbangkan implikasinya terhadap keputusan di tahap berikutnya. Dalam konteks ini, metode greedy akan menentukan strategi penjualan dengan memilih opsi yang memberikan keuntungan terbesar secara langsung pada setiap tahap. Diharapkan metode ini mampu menghasilkan solusi yang cepat, meskipun tidak selalu menjamin optimalitas pada tingkat global.

Di sisi lain, algoritma divide and conquer menyelesaikan masalah dengan cara membaginya menjadi sub-masalah yang lebih kecil, menyelesaikannya secara mandiri, dan kemudian menggabungkan hasilnya untuk mendapatkan solusi akhir. Metode ini memungkinkan analisis yang lebih mendalam terhadap seluruh data penjualan dan distribusi, sehingga dapat menghasilkan solusi yang lebih akurat, meskipun memerlukan waktu komputasi yang cenderung lebih lama dibandingkan dengan metode greedy. Dengan membandingkan hasil dari kedua algoritma ini, kita dapat memperoleh wawasan mengenai efektivitas dan efisiensi masing-masing metode dalam konteks penjualan batik internasional.

HASIL DAN DISKUSI

Penjualan batik ke berbagai negara memerlukan strategi yang efektif untuk memaksimalkan keuntungan. Dalam projek ini, penerapan algoritma komputasi Greedy dan Divide and Conquer adalah pendekatan yang digunakan untuk menyelesaikan masalah

optimasi. Algoritma Greedy beroperasi berdasarkan prinsip pemilihan solusi lokal terbaik di setiap langkah, sementara Divide and Conquer membagi masalah menjadi bagian-bagian kecil yang dapat dikelola secara mandiri, sehingga menghasilkan solusi global yang lebih optimal. Sebelum menjelaskan code inti dari projek ini ada beberapa code yang dibuat untuk mendukung pencarian profit terbesar penjualan batik ke berbagai negara menggunakan algoritma greedy dan devide and conquer. Berikut adalah codenya

```
import pandas as pd
from tabulate import tabulate

data = pd.read_csv('/content/EksporBatik_2010_2021.csv')
data = data.fillna(0)
```

Pada code ini, dataset yang telah disiapkan sebelumnya akan dibaca. Dataset yang akan digunakan untuk projek ini adalah data ekspor batik dari mulai tahun 2010 sampai tahun 2021 yang diambil dari kaggle. Kemudian untuk fungsi code ‘data = data.fillna(0)’ adalah untuk mengganti data yang kosong menjadi angka nol sehingga nantinya tidak ada missing value di dalamnya.

```
years = range(2010, 2022)
for year in years:
    data[f'B-{str(year)[-2:]} (Kg)'] = data[f'B-{str(year)[-2:]} (Kg)'].astype(str).str.replace(',', '.').astype(float)
    data[f'N-{str(year)[-2:]} (USD)'] = data[f'N-{str(year)[-2:]} (USD)'].astype(str).str.replace(',', '.').astype(float)
```

Kode di atas melakukan pra-pemrosesan data untuk memastikan konsistensi format angka pada kolom volume (B-YY (Kg)) dan nilai ekspor (N-YY (USD)) dalam dataset. Dengan menggunakan loop yang mencakup setiap tahun dari 2010 hingga 2021, data pada kolom tersebut diubah dari format string—yang mungkin mengandung tanda koma sebagai pemisah desimal—menjadi format numerik tipe float. Langkah ini sangat penting untuk menghindari kesalahan dalam analisis atau perhitungan matematis, terutama karena format angka pada data ekspor seringkali bervariasi antar wilayah. Melalui proses ini, semua data numerik dipastikan siap untuk diolah oleh algoritma yang akan diterapkan selanjutnya.

```
# # Mengonversi nilai ke format numerik
years = [2011, 2016, 2021]
for year in years:
    data[f'B-{str(year)[-2:]} (Kg)'] = data[f'B-{str(year)[-2:]} (Kg)'].astype(str).str.replace(',', '.').astype(float)
    data[f'N-{str(year)[-2:]} (USD)'] = data[f'N-{str(year)[-2:]} (USD)'].astype(str).str.replace(',', '.').astype(float)

# Memproses data untuk setiap tahun yang dipilih
for year in years:
    filtered_data = data[['Negara', f'B-{str(year)[-2:]} (Kg)', f'N-{str(year)[-2:]} (USD)']].copy()
    filtered_data.columns = ['Negara', 'Berat (kg)', 'Harga (USD)']
    filtered_data['Profit'] = filtered_data['Berat (kg)'] * filtered_data['Harga (USD)']

    # Menyimpan hasil ke file CSV
    filtered_data.to_csv(f"profit_{year}.csv", index=False)

    # Mencetak tabel hasil
    print(f"Hasil untuk tahun {year}:")
    print(tabulate(filtered_data, headers='keys', tablefmt='grid'))
    print("\n")
```

Kode di atas melakukan konversi dan analisis data untuk beberapa tahun yang dipilih sebagai sampel, yaitu 2011, 2016, dan 2021. Langkah pertama melibatkan konversi kolom yang berisi data berat ekspor (B-YY (Kg)) dan nilai ekspor (N-YY (USD)) menjadi tipe data numerik float. Ini dilakukan untuk memastikan keseragaman format yang diperlukan dalam analisis. Selanjutnya, untuk setiap tahun, data di filter untuk menyertakan kolom negara, berat, dan harga. Kemudian, ditambahkan kolom baru bernama "Profit" yang menghitung keuntungan dengan mengalikan berat dengan harga. Hasil analisis ini disimpan dalam file CSV yang dinamai sesuai dengan tahun yang dianalisis. Selain itu, data juga ditampilkan dalam format tabel menggunakan pustaka Tabulate, memudahkan interpretasi hasil untuk setiap tahun.

Code Mencari Profit Terbesar Menggunakan Greedy

```
years = [2011, 2016, 2021]
for year in years:
    data[f'B-{str(year)[-2:]} (Kg)'] = data[f'B-{str(year)[-2:]} (Kg)'].astype(str).str.replace(',', '.').astype(float)
    data[f'N-{str(year)[-2:]} (USD)'] = data[f'N-{str(year)[-2:]} (USD)'].astype(str).str.replace(',', '.').astype(float)

# Memproses data dan menghitung solusi greedy untuk setiap tahun yang dipilih
for year in years:
    filtered_data = data[['Negara', f'B-{str(year)[-2:]} (Kg)', f'N-{str(year)[-2:]} (USD)']].copy()
    filtered_data.columns = ['Negara', 'Berat', 'Nilai']
    filtered_data['Profit'] = filtered_data['Berat'] * filtered_data['Nilai']
    # filtered_data = filtered_data[filtered_data['Profit'] !=0]

    # Menghitung solusi greedy
    greedy_solution = filtered_data.sort_values(by='Profit', ascending=False)
    total_greedy_profit = greedy_solution['Profit'].sum()

    greedy_table = greedy_solution[['Negara', 'Profit']].reset_index(drop=True)
    print(f"Hasil Greedy untuk tahun {year}:")
    print(tabulate(greedy_table, headers='keys', tablefmt='grid'))
    print(f"\nTotal profit dengan algoritma Greedy untuk tahun {year}: {total_greedy_profit}\n")
```

Kode yang di atas menerapkan algoritma Greedy untuk menghitung keuntungan maksimum dari ekspor batik pada tahun-tahun yang dibuat sampel, yaitu 2011, 2016, dan 2021. Langkah pertama adalah memastikan bahwa data berat (B-YY (Kg)) dan nilai ekspor (N-YY (USD)) memiliki format numerik yang konsisten untuk memudahkan perhitungan. Selanjutnya, data di filter agar hanya mencakup kolom negara, berat, dan nilai ekspor. Dari situ, ditambahkan kolom baru yang berisi "Profit", hasil dari perkalian antara berat dan nilai ekspor. Data kemudian diurutkan berdasarkan kolom Profit secara menurun, sesuai dengan prinsip algoritma Greedy, yang memilih keuntungan tertinggi pada setiap langkahnya. Total keuntungan dihitung dengan menjumlahkan semua nilai Profit, dan hasil analisis disajikan dalam bentuk tabel menggunakan pustaka Tabulate, yang juga menampilkan total keuntungan untuk setiap tahun yang dianalisis.

Code Mencari Profit Terbesar Menggunakan Divide and Conquer

```python
def calculate_profit_divide_and_conquer(data, start, end):
    # Basis: jika hanya ada satu elemen
    if start == end:
        return [(data.iloc[start]['Negara'], data.iloc[start]['Profit'])], data.iloc[start]['Profit']

    # Memecah data menjadi dua bagian
    mid = (start + end) // 2
    left_table, left_profit = calculate_profit_divide_and_conquer(data, start, mid)
    right_table, right_profit = calculate_profit_divide_and_conquer(data, mid + 1, end)

    # Menggabungkan hasil dari bagian kiri dan kanan
    combined_table = left_table + right_table
    combined_profit = left_profit + right_profit

    return combined_table, combined_profit

    total_dc_profit = calculate_profit_divide_and_conquer(filtered_data)
```

Kode di atas menerapkan algoritma Divide and Conquer untuk menghitung total profit dari data ekspor batik. Fungsi `calculate_profit_divide_and_conquer` membagi data menjadi dua bagian sehingga mencapai kondisi dasar, yaitu ketika hanya ada satu elemen dalam subset data. Pada kondisi dasar ini, fungsi akan mengembalikan informasi mengenai negara dan profit dari elemen tersebut. Setelah data dipecah, profit masing-masing bagian kiri dan kanan dihitung secara rekursif, dan hasilnya kemudian digabungkan, baik dalam bentuk tabel profit maupun dalam total profit. Hasil akhir berupa penggabungan tabel profit dari semua elemen dan total profit keseluruhan. Namun, terdapat kesalahan kecil pada kode tersebut: variabel `total_dc_profit` telah dideklarasikan, tetapi parameter `start` dan `end` belum ditambahkan saat memanggil fungsi rekursif, sehingga perlu diperbaiki agar berfungsi sesuai dengan tujuannya.

```python
# Memproses data dan menghitung solusi Divide and Conquer untuk setiap tahun yang dipilih
for year in years:
    filtered_data = data[['Negara', f'B-{str(year)[-2:]} (Kg)', f'N-{str(year)[-2:]} (USD)']].copy()
    filtered_data.columns = ['Negara', 'Berat', 'Nilai']
    filtered_data['Profit'] = filtered_data['Berat'] * filtered_data['Nilai']
    # filtered_data = filtered_data[filtered_data['Profit'] !=0]

    # Memastikan data terurut untuk mempermudah proses (opsional, terga Loading... lah)
    filtered_data = filtered_data.sort_values(by='Profit', ascending=False).reset_index(drop=True)

    # Menggunakan Divide and Conquer
    results, total_profit = calculate_profit_divide_and_conquer(filtered_data, 0, len(filtered_data) - 1)

    # Mengonversi hasil ke dalam tabel
    result_table = [{'Negara': negara, 'Profit': profit} for negara, profit in results]

    print(f"Hasil Divide and Conquer untuk tahun {year}:")
    print(tabulate(result_table, headers='keys', tablefmt='grid'))
    print(f"\nTotal profit dengan algoritma Divide and Conquer untuk tahun {year}: {total_profit}\n")
```

Kode di atas mengimplementasikan algoritma Divide and Conquer untuk menghitung dan menampilkan total profit dari penjualan batik pada sampel tahun, yakni 2011, 2016, dan 2021. Proses dimulai dengan memfilter data untuk memilih kolom-kolom yang relevan, yaitu negara, berat, dan nilai ekspor. Selanjutnya, kolom baru yang disebut "Profit" dibuat dengan mengalikan berat dan nilai. Data yang sudah terproses kemudian diurutkan berdasarkan profit

secara menurun, dengan tujuan agar proses penggabungan hasil dari algoritma Divide and Conquer menjadi lebih terstruktur. Fungsi `calculate_profit_divide_and_conquer` digunakan untuk membagi data menjadi bagian-bagian yang lebih kecil, melakukan perhitungan profit secara rekursif, dan menggabungkan hasilnya. Akhirnya, hasil tersebut ditampilkan dalam bentuk tabel yang menunjukkan negara dan profit menggunakan pustaka Tabulate, sementara total profit dihitung dengan menjumlahkan profit dari seluruh data yang telah diolah.

Code Membandingkan Hasil 2 Algoritma Greedy dan Divide and Conquer

```
total_dc_profit = total_profit
total_greedy_profit = greedy_solution['Profit'].sum()

total_profit = filtered_data['Profit'].sum()
greedy_fraction = total_greedy_profit / total_profit
dc_fraction = total_dc_profit / total_profit

fraction_result = pd.DataFrame([
    {"Metode": "Greedy", "Fraction": greedy_fraction},
    {"Metode": "Divide and Conquer", "Fraction": dc_fraction}
])
print(tabulate(fraction_result, headers='keys', tablefmt='grid'))
```

Kode ini dirancang untuk membandingkan kontribusi profit dari dua algoritma, yaitu Greedy dan Divide and Conquer, terhadap total profit penjualan batik. Proses dimulai dengan menghitung total profit yang diperoleh dari masing-masing algoritma, serta profit keseluruhan berdasarkan data yang telah difilter. Selanjutnya, kontribusi tiap algoritma terhadap total profit dihitung dalam bentuk fraksi, dengan membandingkan profit yang dihasilkan oleh masing-masing metode terhadap total profit keseluruhan. Hasil perbandingan ini disajikan dalam sebuah tabel yang mengilustrasikan kontribusi algoritma Greedy dan Divide and Conquer, sehingga memungkinkan analisis mengenai seberapa efektif kedua metode tersebut dalam memaksimalkan keuntungan.

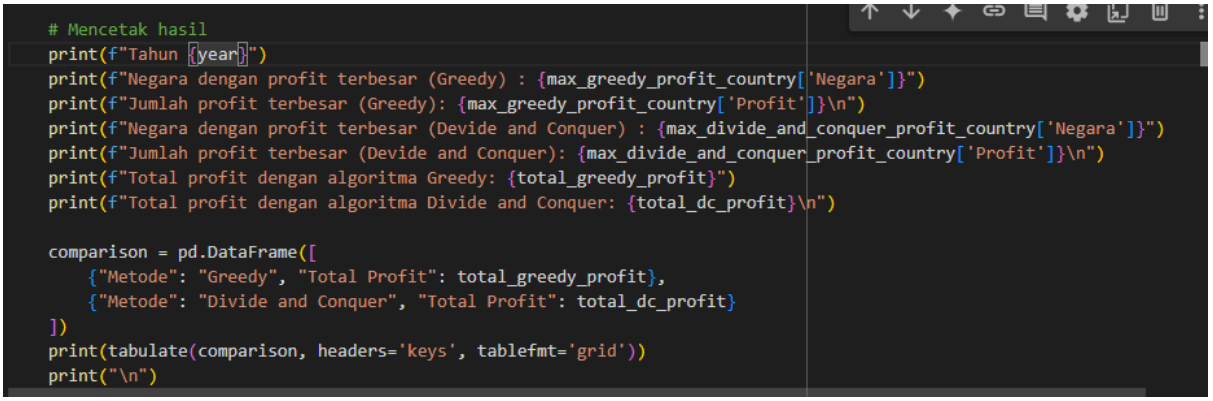
```
years = [2011, 2016, 2021]
for year in years:
    data[f'B-{str(year)[-2:]} (Kg)'] = data[f'B-{str(year)[-2:]} (Kg)'].astype(str).str.replace(',', '.').astype(float)
    data[f'N-{str(year)[-2:]} (USD)'] = data[f'N-{str(year)[-2:]} (USD)'].astype(str).str.replace(',', '.').astype(float)

    filtered_data = data[['Negara', f'B-{str(year)[-2:]} (Kg)', f'N-{str(year)[-2:]} (USD)']].copy()
    filtered_data.columns = ['Negara', 'Berat', 'Nilai']
    filtered_data['Profit'] = filtered_data['Berat'] * filtered_data['Nilai']

    # Menghitung solusi Greedy
    greedy_solution = filtered_data.sort_values(by='Profit', ascending=False)
    total_greedy_profit = greedy_solution['Profit'].sum()
    max_greedy_profit_country = greedy_solution.iloc[0]

    # Menghitung solusi Divide and Conquer
    filtered_data = filtered_data.sort_values(by='Profit', ascending=False).reset_index(drop=True)
    results, total_dc_profit = calculate_profit_divide_and_conquer(filtered_data, 0, len(filtered_data) -1)
    max_divide_and_conquer_profit_country = filtered_data.iloc[0]
```

Kode ini mengimplementasikan perhitungan dan perbandingan antara solusi yang dihasilkan oleh algoritma Greedy dan Divide and Conquer untuk sampel tahun 2011, 2016, dan 2021. Pertama-tama, data untuk setiap tahun diubah menjadi format numerik yang konsisten, sehingga nilai ekspor dan berat disajikan dalam bentuk desimal. Selanjutnya, untuk setiap tahun, dilakukan pemfilteran data dan perhitungan kolom "Profit," yang dihasilkan dari perkalian antara berat dan nilai ekspor. Algoritma Greedy menghitung total profit dengan mengurutkan data berdasarkan profit tertinggi, sekaligus mengidentifikasi negara yang memberikan profit terbesar. Sementara itu, dalam pendekatan Divide and Conquer, data juga diurutkan berdasarkan profit dan diolah dengan fungsi rekursif untuk menghitung total profit, dengan cara yang sama mengidentifikasi negara penyumbang profit tertinggi. Kedua algoritma ini memberikan informasi tentang negara dengan keuntungan tertinggi dan total profit yang diperoleh, sehingga memungkinkan perbandingan efektivitas kedua metode dalam optimasi keuntungan.



```
# Mencetak hasil
print(f"Tahun {year}")
print(f"Negara dengan profit terbesar (Greedy) : {max_greedy_profit_country['Negara']}")
print(f"Jumlah profit terbesar (Greedy): {max_greedy_profit_country['Profit']}\n")
print(f"Negara dengan profit terbesar (Devide and Conquer) : {max_divide_and_conquer_profit_country['Negara']}")
print(f"Jumlah profit terbesar (Devide and Conquer): {max_divide_and_conquer_profit_country['Profit']}\n")
print(f"Total profit dengan algoritma Greedy: {total_greedy_profit}")
print(f"Total profit dengan algoritma Divide and Conquer: {total_dc_profit}\n")

comparison = pd.DataFrame([
    {"Metode": "Greedy", "Total Profit": total_greedy_profit},
    {"Metode": "Divide and Conquer", "Total Profit": total_dc_profit}
])
print(tabulate(comparison, headers='keys', tablefmt='grid'))
print("\n")
```

Kode ini menyajikan perbandingan hasil antara algoritma Greedy dan Divide and Conquer untuk tahun-tahun yang diambil sampel, yaitu 2011, 2016, dan 2021. Pertama, untuk setiap tahun, akan ditampilkan negara dengan profit tertinggi beserta jumlah profit tersebut untuk kedua algoritma. Setelah itu, total profit yang dihasilkan oleh masing-masing algoritma juga akan dicetak. Selanjutnya, sebuah DataFrame dibentuk untuk memvisualisasikan perbandingan total profit dari algoritma Greedy dan Divide and Conquer dalam format tabel yang rapi, menggunakan pustaka Tabulate. Pendekatan ini memberikan gambaran yang jelas mengenai seberapa efektif masing-masing metode dalam memaksimalkan keuntungan, serta memudahkan analisis perbandingan antara kedua algoritma dalam konteks optimasi profit penjualan batik.

```
!pip install big-O-calculator
!pip install memory_profiler
```

Perintah `! pip install big-O-calculator` dan `! pip install memory_profiler` berfungsi untuk menginstal dua pustaka Python yang sangat berguna dalam analisis kinerja program. Pustaka big-O-calculator membantu menghitung kompleksitas waktu algoritma, yang menunjukkan bagaimana waktu eksekusi berubah seiring dengan peningkatan ukuran input, atau yang dikenal sebagai notasi Big O. Ini memberi wawasan tentang efisiensi algoritma

dalam hal kecepatan. Di sisi lain, memory_profiler digunakan untuk memantau penggunaan memori dalam program Python, memungkinkan pengguna untuk mengidentifikasi berapa banyak memori yang digunakan oleh berbagai bagian kode. Kedua pustaka ini sangat penting untuk menganalisis dan mengoptimalkan kinerja program, baik dari segi waktu eksekusi maupun penggunaan memori.

```
import time
import memory_profiler

years = [2011, 2016, 2021]
for year in years:
    # Data type conversion and replacement
    data[f'B-{str(year)[-2:]} (Kg)'] = data[f'B-{str(year)[-2:]} (Kg)'].astype(str).str.replace(',', '.').astype(float)
    data[f'N-{str(year)[-2:]} (USD)'] = data[f'N-{str(year)[-2:]} (USD)'].astype(str).str.replace(',', '.').astype(float)

    # Create filtered data
    filtered_data = data[['Negara', f'B-{str(year)[-2:]} (Kg)', f'N-{str(year)[-2:]} (USD)']].copy()
    filtered_data.columns = ['Negara', 'Berat', 'Nilai']
    filtered_data['Profit'] = filtered_data['Berat'] * filtered_data['Nilai']

    # Measure Greedy solution time and memory
    start_time = time.time()
    mem_before = memory_profiler.memory_usage()[0]
    greedy_solution = filtered_data.sort_values(by='Profit', ascending=False)
    total_greedy_profit = greedy_solution['Profit'].sum()
    max_greedy_profit_country = greedy_solution.iloc[0]
    mem_after = memory_profiler.memory_usage()[0]
    end_time = time.time()
    time_greedy = end_time - start_time
    memory_greedy = mem_after - mem_before
```


Kode diatas dirancang untuk mengukur waktu eksekusi dan penggunaan memori saat menerapkan algoritma Greedy pada data penjualan batik dari tahun 2011, 2016, dan 2021. Proses dimulai dengan mengubah data ke dalam format numerik yang konsisten, diikuti dengan penghitungan kolom "Profit. " Selanjutnya, kode ini mengukur waktu eksekusi dan penggunaan memori sebelum dan setelah penerapan solusi Greedy, yang melibatkan penyortiran data berdasarkan profit serta perhitungan total profit dan negara dengan profit tertinggi. Untuk mengukur waktu eksekusi, digunakan pustaka time, sedangkan penggunaan memori dipantau melalui memory_profiler. Hasil analisis ini memberikan wawasan berharga mengenai efisiensi dari segi waktu dan penggunaan memori dalam menjalankan algoritma Greedy pada dataset tersebut.

```python
# Measure Divide and Conquer time and memory
start_time = time.time()
mem_before = memory_profiler.memory_usage()[0]
filtered_data = filtered_data.sort_values(by='Profit', ascending=False).reset_index(drop=True)
results, total_dc_profit = calculate_profit_divide_and_conquer(filtered_data, 0, len(filtered_data) - 1)
max_divide_and_conquer_profit_country = filtered_data.iloc[0]
mem_after = memory_profiler.memory_usage()[0]
end_time = time.time()
time_dc = end_time - start_time
memory_dc = mem_after - mem_before

# Print results
print(f"Tahun {year}")

print(f"Kompleksitas Algoritma Greedy:")
print(f"  Waktu (s): {time_greedy}")
print(f"  Memori (MiB): {memory_greedy}\n")

print(f"Kompleksitas Algoritma Divide and Conquer:")
print(f"  Waktu (s): {time_dc}")
print(f"  Memori (MiB): {memory_dc}\n")

comparison = pd.DataFrame([
    {"Metode": "Greedy", "Total Profit": total_greedy_profit, "Time (s)": time_greedy, "Memory (MiB)": memory_greedy},
    {"Metode": "Divide and Conquer", "Total Profit": total_dc_profit, "Time (s)": time_dc, "Memory (MiB)": memory_dc}
])
print(tabulate(comparison, headers='keys', tablefmt='grid'))
print("\n")
```

Kode ini dirancang untuk mengukur dan membandingkan kompleksitas waktu serta penggunaan memori antara algoritma Greedy dan Divide and Conquer pada sampel tahun 2011, 2016, dan 2021. Setelah menghitung profit masing-masing metode, kode ini mengevaluasi waktu eksekusi dan penggunaan memori kedua algoritma dengan memanfaatkan pustaka time dan memory_profiler. Waktu eksekusi ditentukan dari selisih antara waktu mulai dan waktu selesai, sedangkan penggunaan memori diukur dengan membandingkan jumlah memori sebelum dan setelah proses eksekusi. Hasil pengukuran, mencakup waktu eksekusi dan penggunaan memori untuk masing-masing algoritma, disajikan dalam format tabel yang memudahkan perbandingan. Dengan cara ini, kode ini memberikan wawasan yang jelas mengenai efisiensi kedua algoritma dalam hal waktu dan penggunaan memori, memungkinkan analisis perbandingan yang bermanfaat untuk optimasi profit penjualan batik.

Analisis Hasil

Berdasarkan semua code yang telah diberikan diatas dapat diambil beberapa hasil analisis yaitu sebagai berikut:

1. Penggunaan algoritma greedy dan divide and conquer memberikan hasil yang sama untuk menentukan negara dengan profit terbesar untuk setiap tahun 2011, 2016, dan 2021. Hal ini dapat terjadi karena kedua algoritma tersebut memilih item berdasarkan rasio keuntungan terhadap berat (profit/berat), yang merupakan inti dari metode Knapsack Fractional. Dalam pendekatan ini, kita diperbolehkan untuk mengambil sebagian dari setiap item, sehingga baik Greedy maupun Divide and Conquer akan terlebih dahulu memilih item yang memiliki rasio profit/berat tertinggi. Karena metode pemilihan ini mirip, terutama jika data sudah terurut atau tidak terlalu kompleks sehingga, kedua algoritma tersebut dapat menghasilkan output yang sama

```
Tahun 2011
Negara dengan profit terbesar (Greedy) : Amerika Serikat
Jumlah profit terbesar (Greedy): 1.8373453923328776e+16

Negara dengan profit terbesar (Devide and Conquer) : Amerika Serikat
Jumlah profit terbesar (Devide and Conquer): 1.8373453923328776e+16

Total profit semua negara dengan algoritma Greedy: 1.9501696301283556e+16
Total profit semua negara dengan algoritma Divide and Conquer: 1.950169630128355e+16

+----+--------------------+----------------+
|    | Metode             |   Total Profit |
+====+====================+================+
|  0 | Greedy             |    1.95017e+16 |
+----+--------------------+----------------+
|  1 | Divide and Conquer |    1.95017e+16 |
+----+--------------------+----------------+
```

```
Tahun 2016
Negara dengan profit terbesar (Greedy) : Amerika Serikat
Jumlah profit terbesar (Greedy): 683468065761853.8

Negara dengan profit terbesar (Devide and Conquer) : Amerika Serikat
Jumlah profit terbesar (Devide and Conquer): 683468065761853.8

Total profit semua negara dengan algoritma Greedy: 687010113661443.1
Total profit semua negara dengan algoritma Divide and Conquer: 687010113661443.1

+----+--------------------+----------------+
|    | Metode             |   Total Profit |
+====+====================+================+
|  0 | Greedy             |     6.8701e+14 |
+----+--------------------+----------------+
|  1 | Divide and Conquer |     6.8701e+14 |
+----+--------------------+----------------+
```

```
Tahun 2021
Negara dengan profit terbesar (Greedy) : Amerika Serikat
Jumlah profit terbesar (Greedy): 29935750240506.727

Negara dengan profit terbesar (Devide and Conquer) : Amerika Serikat
Jumlah profit terbesar (Devide and Conquer): 29935750240506.727

Total profit semua negara dengan algoritma Greedy: 30493981069995.07
Total profit semua negara dengan algoritma Divide and Conquer: 30493981069995.08

+----+--------------------+----------------+
|    | Metode             |   Total Profit |
+====+====================+================+
|  0 | Greedy             |     3.0494e+13 |
+----+--------------------+----------------+
|  1 | Divide and Conquer |     3.0494e+13 |
+----+--------------------+----------------+
```

2. Kompleksitas waktu penggunaan greedy lebih cepat daripada divide and conquer. Hal ini disebabkan karena karena metode kerja divide and conquer yang lebih rumit. Dalam pendekatan Greedy, setiap langkah hanya memerlukan pemilihan elemen terbaik yang tersedia saat itu, menjadikannya lebih cepat dan efisien. Sebagai contoh, algoritma Greedy yang memilih negara dengan profit tertinggi hanya perlu melakukan satu kali pengurutan dan pemilihan. Di sisi lain, algoritma Divide and Conquer membagi masalah menjadi sub-masalah yang lebih kecil, lalu menyelesaikan masing-masing sub-masalah tersebut secara terpisah sebelum menggabungkan hasilnya. Proses ini melibatkan pengulangan penghilangan data menjadi bagian-bagian yang lebih kecil, sehingga jumlah operasi yang dilakukan meningkat secara eksponensial. Meskipun Divide and Conquer memiliki keunggulan dalam menangani sub-masalah yang lebih kompleks, dalam konteks ini, proses pemecahan dan penggabungan data membuat waktu eksekusinya lebih panjang dibandingkan dengan metode Greedy, yang hanya memerlukan pengurutan dan pemilihan yang sederhana.

```
Tahun 2011
Kompleksitas Algoritma Greedy:
  Waktu (s): 0.20179080963134766
  Memori (MiB): 0.0

Kompleksitas Algoritma Divide and Conquer:
  Waktu (s): 0.2874128818511963
  Memori (MiB): 0.0

+----+--------------------+----------------+------------+----------------+
|    | Metode             |   Total Profit |   Time (s) |   Memory (MiB) |
+====+====================+================+============+================+
|  0 | Greedy             |    1.95017e+16 |   0.201791 |              0 |
+----+--------------------+----------------+------------+----------------+
|  1 | Divide and Conquer |    1.95017e+16 |   0.287413 |              0 |
+----+--------------------+----------------+------------+----------------+
```

```
Tahun 2016
Kompleksitas Algoritma Greedy:
  Waktu (s): 0.20187640190124512
  Memori (MiB): 0.0

Kompleksitas Algoritma Divide and Conquer:
  Waktu (s): 0.27260756492614746
  Memori (MiB): 0.0

+----+--------------------+----------------+------------+----------------+
|    | Metode             |   Total Profit |   Time (s) |   Memory (MiB) |
+====+====================+================+============+================+
|  0 | Greedy             |     6.8701e+14 |   0.201876 |              0 |
+----+--------------------+----------------+------------+----------------+
|  1 | Divide and Conquer |     6.8701e+14 |   0.272608 |              0 |
+----+--------------------+----------------+------------+----------------+
```

```
Tahun 2021
Kompleksitas Algoritma Greedy:
  Waktu (s): 0.20212745666503906
  Memori (MiB): 0.0

Kompleksitas Algoritma Divide and Conquer:
  Waktu (s): 0.2642641067504883
  Memori (MiB): 0.0
```

	Metode	Total Profit	Time (s)	Memory (MiB)
0	Greedy	3.0494e+13	0.202127	0
1	Divide and Conquer	3.0494e+13	0.264264	0

KESIMPULAN

Kesimpulan dari projek ini adalah meskipun algoritma Greedy dan Divide and Conquer menghasilkan hasil yang identik dalam menentukan negara dengan profit terbesar, perbedaan utama terletak pada kompleksitas waktu eksekusinya. Keduanya memilih item berdasarkan rasio profit terhadap berat, yang menghasilkan output yang sama, terutama ketika data yang dianalisis tidak terlalu kompleks. Namun, algoritma Greedy terbukti lebih efisien, hanya memerlukan satu langkah pengurutan dan pemilihan, sementara Divide and Conquer memerlukan waktu lebih lama karena melibatkan proses pemecahan masalah menjadi sub-masalah yang kemudian digabungkan kembali. Dengan demikian, meskipun kedua algoritma menawarkan solusi yang serupa, Greedy lebih cepat dan lebih mudah diimplementasikan.

REFERENSI

Kan, A. R., Stougie, L., & Vercellis, C. (1993). A class of generalized greedy algorithms for the multi-knapsack problem. *Discrete applied mathematics*, *42*(2-3), 279-290.

link https://www.sciencedirect.com/science/article/pii/0166218X9390051O

Morales, F. A., & Martínez, J. A. (2019). On the implementation and assessment of several divide & conquer matheuristic strategies for the solution of the knapsack problem. *arXiv preprint arXiv:1901.01215*.

link https://www.researchgate.net/profile/Fernando-Morales-2/publication/330221152_Analysis_of_Divide_Conquer_strategies_for_the_0-1_Minimization_Knapsack_Problem/links/5c3dc104a6fdccd6b5adcac0/Analysis-of-Divide-Conquer-strategies-for-the-0-1-Minimization-Knapsack-Problem.pdf