Aplikasi Algoritma Decrease and Conquer dan Dynamic Programming dalam Melakukan Impedance Tuning pada Rangkaian Resistor Sederhana

Application of the Decrease and Conquer Algorithm and Dynamic Programming in Performing Impedance Tuning on Simple Resistor Circuits

Daniel Egiant Sitanggang - 13521056
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: 13521056@std.stei.itb.ac.id

***Abstract*—Kompleksitas rangkaian elektronik berkembang seiring berkembanganya peradaban manusia. Meningkatnya kompleksitas rangkaian elektronik membuat rekayasawan tidak dapat lagi melakukan *impedance tuning* secara *trial and error*. Pada makalah ini, akan dipecahkan persoalan *impedance tuning* dengan menggunakan algoritma *Decrease and Conquer* dan algoritma program dinamis. Pada makalah ini diperoleh kesimpulan bahwa kedua algoritma tersebut efektif dalam menyelesaikan persoalan *impedance tuning*, implementasi *Decrease and Conquer* memiliki kompleksitas eksponensial dan implementasi algoritma program dinamis memiliki kompleksitas polinomial.**

Keywords—resistor; impedance tuning; decrease and conquer; dynamic programming

I. PENDAHULUAN

Peradaban manusia telah berkembang pesat selama 100 tahun terakhir ditandai dengan majunya teknologi pada perangkat elektronik yang beredar di era modern ini. Perangkat-perangkat tersebut telah berkembang jauh secara kompleksitas rangkaian namun sebenarnya mayoritas komponen-komponen penyusunnya masih sama dengan awal perangkat/ teknologi tersebut ditemukan seperti resistor, kapasitor, induktor, transistor, dan lain sebagainya.

Resistor berperan sangat penting sebagai bagian integral dari rangkaian elektronik. Peran utama resistor adalah untuk memastikan stabilitas perangkat elektronik agar perangkat tersebut bekerja dengan optimal. Hal ini dicapai resistor dengan membatasi dan/atau mengatur tegangan yang berjalan pada rangkaian atau pada sirkuit tertentu.

Agar perangkat elektronik dapat berjalan dengan optimal, rangkaian resistor yang tepat harus dirancang dalam perangkat tersebut. Kegiatan Teknologi disusun oleh komponen komponen listrik dan resistor adalah salah satunya. Peranan resistor sangat penting. Kegiatan merangkai resistor sedemikian rupa sehingga mendapatkan resistansi yang diinginkan sehingga perangkat dapat berjalan dengan optimal ini disebut *resistance tuning/ impedance tuning*.

Impedance tuning memiliki berbagai aplikasi dalam teknologi yang ada pada kehidupan sehari-hari seperti optimasi impedansi antena pada sistem komunikasi nirkabel guna meningkatkan efisiensi transmisi dan jangkauan sinyal, pencocokan impedansi pada perangkat audio yang saling terhubung guna memaksimalkan kualitas suara yang dihasilkan, dan pengaturan impedansi pada teknologi pencitraan medis agar memperoleh kualitas citra yang baik serta aman bagi subjek citra.

Impedance tuning merupakan rutin yang sering dilakukan oleh rekayasawan, namun penulis menemukan masih jarang solusi terkait persoalan tersebut dipublikasikan. Bahkan, tak jarang rutin *impedance tuning* dilakukan dengan *trial and error* yaitu rekayasawan mencoba-coba merangkai rangkaian resistor sehingga diperoleh hambatan ekivalen yang diinginkan.

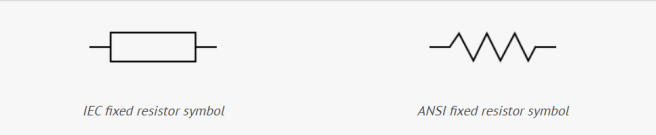
Oleh karena itu, pada makalah ini penulis membagikan alternatif solusi pemecahan masalah *impedance tuning* yaitu persoalan untuk mencari susunan rangkaian resistor berdasarkan masukan sejumlah resistor dan besar hambatan ekivalen. Pemecahan masalah ini menggunakan pendekatan *Decrease and Conquer* dan program dinamis.

II. LANDASAN TEORI

A. Resistor

Resistor adalah sebuah komponen elektronika pasif yang menciptakan hambatan terhadap arus listrik. Resistor mempunyai hambatan yang memiliki satuan ohm Ω . Resistor memiliki beberapa fungsi antara lain: mengatur arus yang masuk ke sirkuit, membagi tegangan, membangkitkan panas, dan menjadi *timer* analog. Terdapat beberapa jenis resistor sesuai dengan aplikasinya seperti *fixed resistors, variable resistors, thermistors, varistors, light dependent resistors* dan

magneto-resistors. Pada makalah kali ini, hanya *fixed resistor* yang akan dibahas dan seluruh kata resistor yang mengacu pada *fixed resistor*.



Gambar 2.1 Representasi Resistor pada Diagram Sirkuit diambil dari https://eepower.com/resistor-guide/https://www.electronics-notes.com/articles/basic_concepts/resistance/resistors-in-series-parallel.php

B. Hukum Ohm

Hukum Ohm adalah hukum fisika yang menyatakan hubungan resistor dengan tegangan dan arus listrik yang mengalir pada rangkaian tersebut. Hukum Ohm menyatakan bahwa besar resistansi suatu resistor adalah konstan sesuai dengan persamaan berikut:

$$R = \frac{V}{I}$$

Gambar 2.2 Persamaan Resistansi Ohm diambil dari galeri pribadi

Hal tersebut mengakibatkan resistor-resistor yang dirangkai secara seri memiliki besar resistansi ekivalen sesuai dengan ekspresi berikut:

$$R_{total} = R_1 + R_2 + R_3 \ldots$$

Gambar 2.3 Persamaan Resistansi Ekivalen pada Rangkaian Resistor Seri diambil dari https://www.electronics-notes.com/articles/basic_concepts/resistance/resistors-in-series-parallel.php

dan resistor-resistor yang dirangkai secara paralel memiliki besar resistansi ekivalen sesuai dengan ekspresi berikut:

$$\frac{1}{R_{total}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \cdots$$

Gambar 2.4 Persamaan Resistansi Ekivalen pada Rangkaian Resistor Paralel diambil dari https://www.electronics-notes.com/articles/basic_concepts/resistance/resistors-in-series-parallel.php

Persamaan-persamaan tersebut yang akan menjadi acuan untuk mengevaluasi besarnya hambatan ekivalen pada suatu sirkuit/ rangkaian resistor.

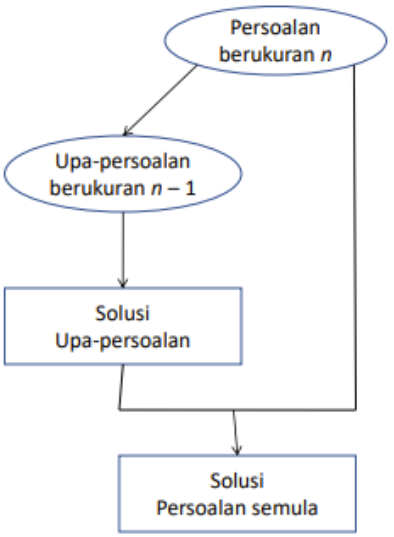
C. Decrease and Conquer

Algoritma *Decrease and Conquer* diartikan sebagai algoritma dengan mereduksi persoalan menjadi dua upa-persoalan (sub-problem) yang lebih kecil, tetapi selanjutnya hanya memproses satu sub-persoalan saja (Rinaldi Munir, 2021). Algoritma *Decrease and Conquer* berbeda dengan algoritma *Divide and Conquer* yang memecah persoalan menjadi banyak upa-persoalan, kemudian memecahkan dan menggabungkan solusi dari tiap upa-persoalan sehingga menjadi solusi dari persoalan utama. Pada *Decrease and Conquer,* tiap iterasi telah mereduksi persoalan utama sehingga tidak perlu dilakukan penyatuan solusi kembali.

Algoritma *Decrease and Conquer* memiliki tiga buah varian yaitu:

- Decrease by a constant

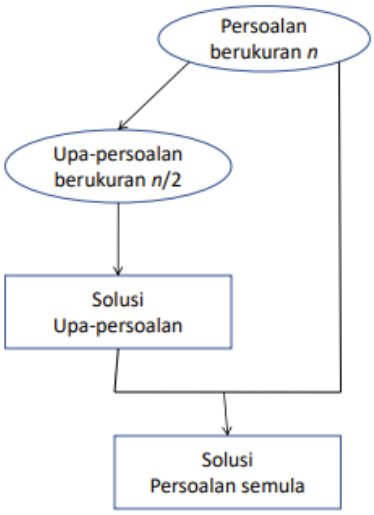
  yaitu melakukan reduksi persoalan sebesar konstanta yang sama pada setiap iterasi algoritma. Beberapa persoalan yang dapat dipecahkan dengan metode ini adalah perpangkatan a^n, *selection sort*, dan *insertion sort.*



Gambar 2.5 Skema Umum algoritma Decrease and Conquer by a Constant diambil dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Decrease-and-Conquer-2021-Bagian1.pdf

- Decrease by a constant factor

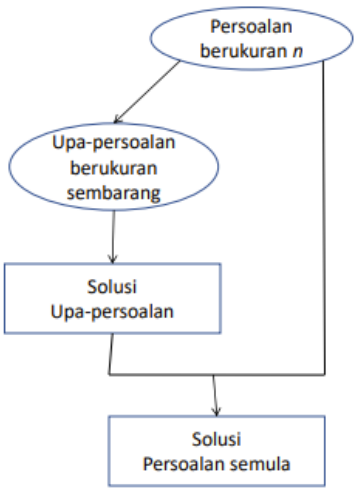
  yaitu melakukan reduksi persoalan sebesar faktor konstanta yang sama setiap iterasi algoritma. Beberapa persoalan yang dapat dipecahkan dengan metode ini adalah *binary search* dan mencari koin palsu.



Gambar 2.5 Skema Umum algoritma Decrease and Conquer by a Factor diambil dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Decrease-and-Conquer-2021-Bagian1.pdf

- Decrease by a variable size

  yaitu melakukan reduksi persoalan dengan besaran yang bervariasi pada setiap iterasi algoritma. Beberapa persoalan yang dapat dipecahkan dengan metode ini adalah *interpolation search* dan *selection problem.*



Gambar 2.5 Skema Umum algoritma Decrease and Conquer by a Constant diambil dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Decrease-and-Conquer-2021-Bagian2.pdf

D. Dynamic Programming dan Memoization

Program dinamis/*dynamic programming* adalah metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan tahapan sehingga persoalan dapat dipandang sebagai serangkaian keputusan yang saling berkaitan (Rinaldi Munir, 2023). Kata dinamis pada program dinamis mengacu pada penggunaan tabel pada proses pencarian solusi yang berkembang seiring langkah pencarian solusi.

Algoritma program dinamis memiliki dua pendekatan yaitu program dinamis maju (top-down) dan program dinamis mundur (bottom-up).

Pendekatan top-down adalah pendekatan yang dimulai dari permasalah yang besar, lalu dipecah menjadi permasalahan yang lebih kecil. Pada proses pemecahan masalah, solusi dari sub masalah yang telah dipecahkan disimpan dalam tabel untuk dapat digunakan kembali. Pendekatan top-down ini biasanya menggunakan teknik rekursi.

Pendekatan bottom-up adalah pendekatan yang dimulai dengan memecahkan submasalah terkecil, kemudian menyimpan solusi tersebut dalam tabel. Solusi dari sub masalah tersebut kemudian digunakan untuk memecahkan masalah yang lebih besar hingga masalah utama. Pendekatan bottom-up ini biasanya menggunakan teknik iteratif.

Memoization adalah salah satu teknik dalam program dinamis guna mengoptimalkan perhitungan dari sub masalah yang dipanggil kembali. Prinsip dari memoization adalah menyimpan solusi dari setiap sub masalah sehingga solusi tersebut dapat diambil kembali ketika diperlukan. Hal ini dilakukan untuk menghindari perulangan perhitungan yang tidak perlu. Dengan melakukan memoization, kompleksitas waktu dari suatu algoritma dapat menurun secara signifikan. Implementasi dari memoization biasanya dengan menggunakan *hash table, array,* atau *cache.*

E. Impedance Tuning Problem

Impedance Tuning Problem merupakan persoalan untuk menyusun rangkaian resistor sehingga mendapatkan hambatan total yang diinginkan. Pada persoalan ini, akan diberikan sebuah list resistor dengan n buah elemen, dan sebuah nilai hambatan target sebesar k. Masing-masing resistor mempunyai besar hambatan secara acak berupa bilangan positif. Berdasarkan resistor-resistor pada list tersebut, akan dirangkai sebuah rangkaian listrik sedemikian rupa sehingga rangkaian tersebut memiliki hambatan ekivalen sebesar k.

Misalnya yaitu n = 5 dengan list_resistor = {10, 10, 20, 30, 40} dan k = 55, maka solusi persoalan tersebut adalah (S 20 30 (P 10 10)), artinya resistor dengan hambatan 10 dirangkai secara paralel dengan resistor dengan hambatan 10 lainnya, kemudian rangkaian tersebut dirangkai secara seri dengan resistor dengan hambatan 20 dan 30.

Impedance Tuning Problem sebenarnya dapat diselesaikan algoritma *exhaustive search*, namun pemecahan masalah dengan pendekatan tersebut menghasilkan algoritma dengan kompleksitas waktu yang tinggi. Pendekatan *exhaustive search* dalam menyelesaikan

persoalan *impedance tuning problem* memiliki kompleksitas waktu $O(2^n . n!. 2^n)$ yaitu banyaknya subset dari resistor yang hendak digunakan dikalikan permutasi tiap subset, dikalikan dengan kombinasi operasi/*parenthesis* resistor-resistor yang digabung menjadi satu rangkaian. Oleh karena itu, pada makalah ini akan digunakan pendekatan *decrease and conquer* untuk melakukan pencarian solusi. kemudian algoritma tersebut dioptimasi kembali dengan menggunakan tabel *Memoization* untuk meminimalisir pemanggilan fungsi pada persoalan yang sama secara berulang.

III. IMPLEMENTASI DAN PEMBAHASAN

Pada bab ini akan dipecahkan permasalahan *impedance tuning*. Terdapat dua pendekatan yang cukup identik yang digunakan untuk menyelesaikan persoalan *impedance tuning* pada makalah ini yaitu dengan menggunakan algoritma *Decrease and Conquer* dan algoritma *dynamic programming.* Selanjutnya, kedua solusi penyelesaian masalah tersebut akan dibandingkan dan dianalisis lebih lanjut.

A. Representasi Komponen Listrik dan Sirkuit Listrik

Sebelum mengimplementasikan algoritma, perlu dibuat representasi dari sirkuit dan komponen listrik pada komputer. Implementasi dari representasi komponen listrik dan sirkuit listrik pada program yang dibuat pada makalah ini serta menjadi acuan untuk algoritma penyelesaian dibuat menggunakan pendekatan OOP. Berikut adalah implementasi dari komponen listrik (resistor) serta sirkuit dalam bahasa pemrograman *Python.*

```
class Component(ABC):
    @abstractmethod
    def calculate_magnitude(self):
        pass

class Resistor(Component):
    def __init__(self, magnitude):
        self.magnitude = magnitude

    def calculate_magnitude(self):
        return self.magnitude

    def __str__(self):
        return str(self.magnitude)

class SeriesCircuit(Component):
    def __init__(self, components):
        self.components = components

    def calculate_magnitude(self):
        return
sum(component.calculate_magnitude() for
component in self.components)

    def __str__(self):
        return "(S" + ' '.join(str(c) for c in
self.components) + ")"
```

```
class ParalelCircuit(Component):
    def __init__(self, components):
        self.components = components

    def calculate_magnitude(self):
        return
1/sum(1/component.calculate_magnitude() for
component in self.components)

    def __str__(self):
        return "(P" + ' '.join(str(c) for c in
self.components) + ")"
```

B. Penyelesaian Persoalan dengan Decrease and Conquer

Pada makalah ini akan dipecahkan persoalan *impedance tuning* dengan menggunakan pendekatan *Decrease and Conquer* by constant. Penyelesaian persoalan *impedance tuning problem* dengan pendekatan *Decrease and Conquer* adalah dengan mencoba membangun rangkaian resistor dari rangkaian paling sederhana, yaitu satu buah resistor. Kemudian rangkai resistor tersebut sehingga persoalan menjadi persoalan yang sama dengan n_baru = n_lama-1 dan impedansi_target_baru <= impedansi_target_lama. Lebih jelasnya, berikut adalah langkah-langkah penyelesaiannya:

1. Ambil satu buah resistor dari list_resistor
2. Resistor tersebut mempunyai alternatif apakah digunakan dalam rangkaian (hidup) atau tidak (mati). Jika resistor tersebut hidup, maka terdapat dua buah alternatif apakah resistor tersebut dirangkai secara seri atau secara paralel pada rangkaian yang sudah ada.

   – Jika dirangkai secara seri, maka persoalan menjadi persoalan yang sama untuk n baru = n lama - 1 dan target resistance menjadi resistance_lama - resistance resistor yang dipilih (sesuai dengan hukum ohm pada rangkaian seri).

   – Jika dirangkai secara paralel, maka persoalan menjadi persoalan yang sama untuk n baru = n lama - 1 dan target resistance menjadi (resistance_lama * resistance resistor yang dipilih) / (resistance_lama - resistance resistor yang dipilih) (sesuai dengan hukum ohm pada rangkaian paralel).

   Terapkan kembali algoritma serupa pada masing-masing upa persoalan. Jika upa persoalan berhasil dipecahkan, maka persoalan utama berhasil dipecahkan pula.
3. Lakukan hal tersebut pada setiap resistor dalam list resistor.

Berikut adalah implementasinya dalam bahasa pemrograman *Python.*

```
def arrange_circuit(list_resistor,
resistance):
```

```
    num_resistors = len(list_resistor)
    if (num_resistors == 0):
        return None

    if num_resistors == 1:
        current_resistor = list_resistor[0]
        if
current_resistor.calculate_magnitude() ==
resistance:
            return current_resistor
        else:
            return None

    for i in range(len(list_resistor)):
        current_resistor = list_resistor[i]
        current_resistor_magnitude =
current_resistor.calculate_magnitude()
        tail_resistor = list_resistor[0:i] +
list_resistor[i+1:]
        # if current resistor connected
series with the rest
        tail_target_series = resistance -
current_resistor_magnitude
        if not tail_target_series < 0 :
            tail_result =
arrange_circuit(tail_resistor,
tail_target_series)
            if tail_result is not None:
                return
SeriesCircuit([current_resistor,
tail_result])

        # if current resistor connected
paralel with the rest
        divisor = current_resistor_magnitude
- resistance
        if (divisor > 0):
            tail_target_paralel =
resistance*current_resistor_magnitude/divisor
            tail_result =
arrange_circuit(tail_resistor,
tail_target_paralel)
            if tail_result is not None:
                return
ParalelCircuit([current_resistor,
tail_result])

        # if current resistor excluded
        tail_result =
arrange_circuit(tail_resistor, resistance)
        if tail_result is not None:
            return tail_result
```

C. Penyelesaian Persoalan dengan Dynamic Programming

Sebelumnya telah diimplementasikan solusi persoalan *impedance tuning* dengan menggunakan pendekatan *Decrease and Conquer*. Namun pendekatan tersebut memiliki kekurangan yaitu fungsi rekursif dengan argumen yang sama dipanggil berulang kali. Oleh karena itu algoritma *Decrease and Conquer* sebelumnya akan dimodifikasi dengan menggunakan teknik *Memoization* untuk menyimpan solusi dari fungsi rekursif yang telah dipanggil pada sebuah tabel yang diimplementasikan dengan *Python's Dictionary*. Hal tersebut menjadikan algoritma ini bukan lagi *Decrease and Conquer* melainkan algoritma *Dynamic Programming*. Lebih jelasnya, berikut adalah *source-code program* yang telah dimodifikasi dalam bahasa pemrograman *Python*:

```
def arrange_circuit_with_memo(list_resistor,
resistance, memo={}):
    key = (tuple(list_resistor), resistance)
    if key in memo:
        return memo[key]

    num_resistors = len(list_resistor)
    if (num_resistors == 0):
        return None

    if num_resistors == 1:
        current_resistor = list_resistor[0]
        if
current_resistor.calculate_magnitude() ==
resistance:
            memo[key] = current_resistor
            return current_resistor
        else:
            return None

    for i in range(len(list_resistor)):
        current_resistor = list_resistor[i]
        current_resistor_magnitude =
current_resistor.calculate_magnitude()
        tail_resistor = list_resistor[0:i] +
list_resistor[i+1:]
        # if current resistor connected
series with the rest
        tail_target_series = resistance -
current_resistor_magnitude
        if not tail_target_series < 0 :
            tail_result =
arrange_circuit_with_memo(tail_resistor,
tail_target_series, memo)
            if tail_result is not None:
                result =
SeriesCircuit([current_resistor,
tail_result])
                memo[key] = result
                return result
        # if current resistor connected
paralel with the rest
        divisor = current_resistor_magnitude
- resistance
        if (divisor > 0):
            tail_target_paralel =
resistance*current_resistor_magnitude/diviso
r
            tail_result =
```

```
arrange_circuit_with_memo(tail_resistor,
tail_target_paralel, memo)
            if tail_result is not None:
                result =
ParalelCircuit([current_resistor,
tail_result])
                memo[key] = result
                return result
        # if current resistor excluded
        tail_result =
arrange_circuit_with_memo(tail_resistor,
resistance, memo)
        if tail_result is not None:
            memo[key] = tail_result
            return tail_result

    memo[key] = None
    return None
```

IV. PENGUJIAN DAN ANALISIS

A. Pengujian

Kasus Uji 1, n = 5, ada solusi

Masukan:
r1 = Resistor(10)
r2 = Resistor(20)
r3 = Resistor(30)
r4 = Resistor(40)
r5 = Resistor(50)
list_resistor = [r1, r2, r3, r4, r5]
target = 45

Keluaran:

Algoritma Decrease and Conquer:
Solution: (S20 (P50 (S10 40)))
Execution time: 0.0ms

Algoritma Dynamic Programmin:
Solution: (S20 (P50 (S10 40)))
Execution time: 0.0010306835174560547ms

Kasus Uji 2, n = 5, tidak ada solusi

Masukan:
r1 = Resistor(10)
r2 = Resistor(20)
r3 = Resistor(30)
r4 = Resistor(40)
r5 = Resistor(50)
list_resistor = [r1, r2, r3, r4, r5]
target = 77

Keluaran:

Algoritma Decrease and Conquer:
Solution: None
Execution time: 0.003995180130004883ms

Algoritma Dynamic Programming:
Solution: None
Execution time: 0.001996278762817383ms

Kasus Uji 3, n = 7, ada solusi

Masukan:
r1 = Resistor(10)
r2 = Resistor(20)
r3 = Resistor(30)
r4 = Resistor(40)
r5 = Resistor(50)
r6 = Resistor(60)
r7 = Resistor(70)

list_resistor = [r1, r2, r3, r4, r5, r6, r7]
target = 55

Keluaran:

Algoritma Decrease and Conquer:
Solution: (S10 (S20 (P30 (S40 (S50 60)))))
Execution time: 0.0ms

Algoritma Dynamic Programming:
Solution: (S10 (S20 (P30 (S40 (S50 60)))))
Execution time: 0.0ms

Kasus Uji 4, n = 7, tidak ada solusi

Masukan:
r1 = Resistor(10)
r2 = Resistor(20)
r3 = Resistor(30)
r4 = Resistor(40)
r5 = Resistor(50)
r6 = Resistor(60)
r7 = Resistor(70)

list_resistor = [r1, r2, r3, r4, r5, r6, r7]
target = 77

Keluaran:

Algoritma Decrease and Conquer:
Solution: None
Execution time: 0.8064179420471191ms

Algoritma Dynamic Programming:
Solution: None
Execution time: 0.07766032218933105ms

Jumlah operasi (evaluasi resistor) pada masing-masing algoritma

```
D:\Kuliah\Semester 4\STIMA\Makalah STIMA>python -u "d:\Ku
  Ukuran masukan (n)    Comp w/o memo    Comp w/memo
--------------------  ---------------  ------------
                   1                1             1
                   2                6           5.7
                   3               39            39
                   4            292.8           187
                   5             2958           908.1
                   6          31095.1        4774.3
                   7           429924         17167
```

B. Analisis

Baik algoritma *Decrease and Conquer* dan algoritma program dinamis efektif untuk menyelesaikan persoalan *Impedance Tuning*. Hal ini terlihat dari kedua algoritma tersebut berhasil menemukan solusi pada persoalan yang memang memiliki solusi, dan tidak mengembalikan solusi pada persoalan yang tidak memiliki solusi.
Berdasarkan jumlah operasinya, efisiensi algoritma program dinamis dapat dihampiri dengan fungsi polinomial dengan orde 6, $O(n^6)$. Sedangkan algoritma *Decrease and Conquer* tidak. Hal ini berarti kompleksitas algoritma *Decrease and Conquer* bukan merupakan polinomial, melainkan eksponensial.
Terdapat juga temuan berupa algoritma Decrease and Conquer konsisten lebih lambat daripada algoritma program dinamis. Berikut adalah perbandingan operasi dari masing-masing algoritma.

Gambar 4.1 Perbandingan Operasi algoritma Decrease and Conquer (tanpa memoisasi) dan Dynamic Programming (dengan memoisasi) diambil dari galeri pribadi.

Berdasarkan grafik tersebut dapat dilihat bahwa algoritma *Decrease and Conquer* memiliki kompleksitas eksponensial dan pengaplikasian *memoization* efektif memangkuskan algoritma *Decrease and Conquer* yang telah dibuat.

V. KESIMPULAN

- Baik algoritma *Decrease and Conquer* dan algoritma program dinamis efektif menemukan solusi persoalan *Impedance Tuning*.
- Algoritma *Decrease and Conquer* memiliki kompleksitas eksponensial sedangkan algoritma *dynamic programming* memiliki kompleksitas $O(n^6)$.
- Teknik *memoization* pada program dinamis efektif untuk memangkuskan algortima yang banyak memakai rekursi seperti *Decrease and Conquer*.

VIDEO LINK AT YOUTUBE

https://www.youtube.com/watch?v=giTxKKnVdVQ

PRANALA GITHUB

https://github.com/egijago/resistor_arranger

ACKNOWLEDGMENT

Puji syukur kehadirat Tuhan Yang Maha Esa atas segala karunia-Nya yang melimpah dalam perjalanan penulisan makalah ini. Penulis ingin mengucapkan terima kasih sebesar-besarnya kepada semua pihak yang telah memberikan dukungan dan kontribusi dalam penyelesaian makalah ini yang berjudul "Aplikasi Algoritma Decrease and Conquer dan Dynamic Programming dalam Melakukan Impedance Tuning pada Rangkaian Resistor Sederhana".

Penulis ingin mengucapkan terima kasih kepada:

Ibu Dr. Nur Ulfa Maulidevi serta Bapak Dr. Ir. Rinaldi Munir, M.T, selaku dosen pengampu yang telah memberikan bimbingan, arahan, serta ilmu yang digunakan dalam menuliskan makalah ini

Seluruh dosen dan staf pengajar di jurusan Teknik Informatika yang telah berperan penting dalam memberikan pengetahuan dan wawasan yang sangat berarti bagi penulis.

Teman-teman seangkatan dan keluarga yang telah memberikan dukungan moril dan semangat dalam menghadapi tantangan penulisan makalah ini.

Penulis juga ingin menyampaikan terima kasih kepada semua pihak yang namanya tidak dapat disebutkan satu persatu namun telah memberikan kontribusi baik secara langsung maupun tidak langsung dalam penulisan makalah ini.

Akhir kata, penulis menyampaikan permohonan maaf jika terdapat kekurangan atau kesalahan dalam penulisan makalah ini. Semoga makalah ini dapat bermanfaat dan memberikan kontribusi positif dalam bidang ilmu yang penulis pelajari.

References

[1] https://eepower.com/resistor-guide/resistor-fundamentals/what-is-a-resistor/ (diakses pada 22 Mei 2023)

[2] https://eepower.com/resistor-guide/ (diakses pada 22 Mei 2023)

[3] https://www.electronics-notes.com/articles/basic_concepts/resistance/resistors-in-series-parallel.php (diakses pada 22 Mei 2023)

[4] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Decrease-and-Conquer-2021-Bagian1.pdf\ (diakses pada 22 Mei 2023)

[5] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Decrease-and-Conquer-2021-Bagian2.pdf\ (diakses pada 22 Mei 2023)

[6] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian1.pdf (diakses pada 22 Mei 2023)

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 22 Mei 2023