

LAPORAN
PRAKTIKUM JOBSHEET 4
(Brute Force dan Divide Conquer)



Disusun Oleh :

VALENTINA SANTI GREHASTA

NIM. 2341720016

TI-1E/27

JURUSAN TEKNOLOGI INFORMASI
PRODI D-IV TEKNIK INFORMATIKA
POLITEKNIK NEGERI MALANG

2024

JOBSHEET IV

BRUTE FORCE DAN DIVIDE CONQUER

4.1 Tujuan Praktikum

Setelah melakukan materi praktikum ini, mahasiswa mampu:

1. Mahasiswa mampu membuat algoritma brute force dan divide-conquer
2. Mahasiswa mampu menerapkan penggunaan algoritma brute force dan divide-conquer

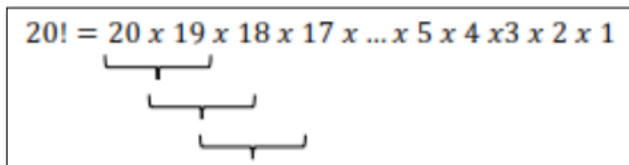
4.2 Menghitung Nilai Faktorial dengan Algoritma Brute Force dan Divide and Conquer

Perhatikan Diagram Class berikut ini :

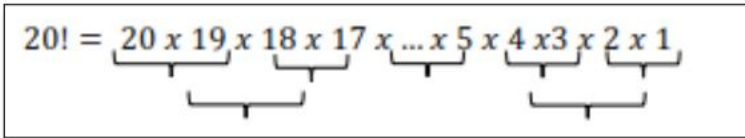
Faktorial
nilai: int
faktorialBF(): int
faktorialDC(): int

Berdasarkan diagram class di atas, akan dibuat program class dalam Java. Untuk menghitung nilai faktorial suatu angka menggunakan 2 jenis algoritma, Brute Force dan Divide and Conquer. Jika digambarkan terdapat perbedaan proses perhitungan 2 jenis algoritma tersebut sebagai berikut :

Tahapan pencarian nilai faktorial dengan algoritma Brute Force :



Tahapan pencarian nilai faktorial dengan algoritma Divide and Conquer :



4.2.1 Langkah-langkah Percobaan

Kode Program :

- Class Faktorial

```
package minggu5;
public class Faktorial {
    int nilai;
    int faktorialBF(int n) {
        int fakto = 1;
        for(int i = 1; i <= n; i++) {
            fakto = fakto * i;
        }
        return fakto;
    }
    int faktorialDC(int n) {
        if(n==1) {
            return 1;
        } else {
            int fakto = n * faktorialDC(n-1);
            return fakto;
        }
    }
}
```

- Class MainFaktorial

```
package minggu5;

import java.util.Scanner;

public class MainFaktorial {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("_____");
        System.out.println("Masukkan jumlah elemen: ");
        int iJml = sc.nextInt();

        Faktorial[] fk = new Faktorial[10];
        for (int i=0; i < iJml; i++) {
            fk[i] = new Faktorial();
            System.out.println("Masukkan nilai data ke-" +(i+1)+":");
            fk[i].nilai = sc.nextInt();
        }

        System.out.println("HASIL - BRUTE FORCE");
        for (int i=0; i < iJml; i++) {
            System.out.println("Hasil penghitungan faktorial
menggunakan Brute Force adalah "
                + fk[i].faktorialBF(fk[i].nilai));
        }

        System.out.println("HASIL - DIVIDE AND CONQUER:");
        for (int i=0; i < iJml; i++) {
            System.out.println("Hasil penghitungan faktorial
menggunakan Divide and Conquer adalah "
                + fk[i].faktorialDC(fk[i].nilai));
        }
    }
}
```

4.2.2 Verifikasi Hasil Percobaan

```
-----
Masukkan jumlah elemen:
3
Masukkan nilai data ke-1:
5
Masukkan nilai data ke-2:
8
Masukkan nilai data ke-3:
3
HASIL - BRUTE FORCE
Hasil penghitungan faktorial menggunakan Brute Force adalah 120
Hasil penghitungan faktorial menggunakan Brute Force adalah 40320
Hasil penghitungan faktorial menggunakan Brute Force adalah 6
HASIL - DIVIDE AND CONQUER:
Hasil penghitungan faktorial menggunakan Divide and Conquer adalah 120
Hasil penghitungan faktorial menggunakan Divide and Conquer adalah 40320
Hasil penghitungan faktorial menggunakan Divide and Conquer adalah 6
PS D:\Struktur Data dan Algoritma\BruteForceDivideConquer> █
```

4.2.3 Pertanyaan

1. Pada base line Algoritma Divide Conquer untuk melakukan pencarian nilai faktorial, jelaskan perbedaan bagian kode pada penggunaan if dan else!

Jawaban:

Pada algoritma Divide and Conquer untuk menghitung faktorial, bagian kode **if** digunakan untuk menentukan kondisi dasar di mana perhitungan rekursif berhenti, yaitu ketika parameter **n** sudah mencapai 1, sehingga nilai faktorial yang dihasilkan adalah 1. Sedangkan bagian kode **else** akan dieksekusi ketika kondisi dasar tidak terpenuhi, yang berarti perhitungan rekursif dilanjutkan dengan mengalikan nilai **n** dengan hasil dari panggilan rekursif untuk **n-1**, menerapkan pendekatan rekursif dalam menyelesaikan masalah yang lebih besar dengan memecahnya menjadi masalah yang lebih kecil.

2. Apakah memungkinkan perulangan pada method faktorialBF() diubah selain menggunakan for? Buktikan!

Jawaban:

Ya, memungkinkan. Bisa menggunakan rekursi untuk mengimplementasikan method faktorialBF().

Bukti Kode Program :

```
package minggu5;

public class Faktorial {
    int nilai;

    // int faktorialBF(int n) {
    //     int fakto = 1;
    //     for(int i = 1; i <= n; i++) {
    //         fakto = fakto * i;
    //     }
    //     return fakto;
    // }

    int faktorialBF(int n) {
        if (n==0 || n==1) {
            return 1;
        } else {
            return n * faktorialBF(n - 1);
        }
    }

    int faktorialDC(int n) {
        if(n==1) {
            return 1;
        } else {
            int fakto = n * faktorialDC(n-1);
            return fakto;
        }
    }
}
```

Output :

```

-----
Masukkan jumlah elemen:
3
Masukkan nilai data ke-1:
5
Masukkan nilai data ke-2:
8
Masukkan nilai data ke-3:
3
HASIL - BRUTE FORCE
Hasil penghitungan faktorial menggunakan Brute Force adalah 120
Hasil penghitungan faktorial menggunakan Brute Force adalah 40320
Hasil penghitungan faktorial menggunakan Brute Force adalah 6
HASIL - DIVIDE AND CONQUER:
Hasil penghitungan faktorial menggunakan Divide and Conquer adalah 120
Hasil penghitungan faktorial menggunakan Divide and Conquer adalah 40320
Hasil penghitungan faktorial menggunakan Divide and Conquer adalah 6
PS D:\Struktur Data dan Algoritma\BruteForceDivideConquer>

```

3. Jelaskan perbedaan antara **fakto *= i;** dan **int fakto = n * faktorialDC(n-1);** !

Jawaban :

Perbedaan antara **fakto *= i;** dan **int fakto = n * faktorialDC(n-1);** terletak pada konteks penggunaannya. **fakto *= i;** digunakan dalam perulangan untuk mengakumulasi hasil perkalian dalam method **faktorialBF()**, sementara **int fakto = n * faktorialDC(n-1);** digunakan dalam rekursi untuk memecah perhitungan faktorial menjadi sub-perhitungan yang lebih kecil dalam method **faktorialDC()**. Proses rekursif ini mengalikan nilai **n** dengan hasil faktorial dari **n-1**, secara berulang hingga mencapai kondisi dasar (base case), sementara perulangan **fakto *= i;** mengalikan **fakto** dengan nilai **i** pada setiap iterasi untuk menghasilkan nilai faktorial dari **n**.

4.3 Menghitung Hasil Pangkat dengan Algoritma Brute Force dan Divide and Conquer

Pada praktikum ini kita akan membuat program class dalam Java. Untuk menghitung nilai pangkat suatu angka menggunakan 2 jenis algoritma, Brute Force dan Divide and Conquer.

4.3.1 Langkah-langkah Percobaan

Kode Program :

- Class Pangkat :

```

package minggu5;

public class Pangkat {
    public int nilai, pangkat;

    int pangkatBF(int a, int n) {
        int hasil = 1;
        for (int i = 0; i < n; i++) {
            hasil *= a;
        }
        return hasil;
    }
}

```

```

int pangkatDC(int a, int n) {
    if (n == 0) {
        return 1;
    } else {
        int hasil = pangkatDC(a, n / 2);
        if (n % 2 == 0)
        {
            return hasil * hasil;
        } else {
            return hasil * hasil * a;
        }
    }
}
}

```

- Class MainPangkat :

```

package minggu5;

import java.util.Scanner;

public class MainPangkat {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("=====");
        System.out.println("Masukkan jumlah elemen yang dihitung: ");
        int elemen = sc.nextInt();

        Pangkat[] png = new Pangkat[elemen];
        for (int i = 0; i < elemen; i++) {
            png[i] = new Pangkat();
            System.out.println("Masukkan nilai yang hendak
dipangkatkan:");
            png[i].nilai = sc.nextInt();
            System.out.println("Masukkan nilai pemangkat:");
            png[i].pangkat = sc.nextInt();
        }

        System.out.println("HASIL PANGKAT - BRUTE FORCE");
        for (int i = 0; i < elemen; i++) {
            System.out.println("Hasil dari " + png[i].nilai + " pangkat "
+ png[i].pangkat + " adalah "
+ png[i].pangkatBF(png[i].nilai, png[i].pangkat));
        }
        System.out.println("HASIL PANGKAT - DIVIDE AND CONQUER");
        for (int i = 0; i < elemen; i++) {
            System.out.println("Hasil dari " + png[i].nilai + " pangkat "
+ png[i].pangkat + " adalah "
+ png[i].pangkatDC(png[i].nilai, png[i].pangkat));
        }
    }
}

```

4.3.2 Verifikasi Hasil Percobaan

```

=====
Masukkan jumlah elemen yang dihitung:
2
Masukkan nilai yang hendak dipangkatkan:
6
Masukkan nilai pemangkat:
2
Masukkan nilai yang hendak dipangkatkan:
4
Masukkan nilai pemangkat:
3
HASIL PANGKAT - BRUTE FORCE
Hasil dari 6 pangkat 2 adalah 36
Hasil dari 4 pangkat 3 adalah 64
HASIL PANGKAT - DIVIDE AND CONQUER
Hasil dari 6 pangkat 2 adalah 36
Hasil dari 4 pangkat 3 adalah 64

```

4.3.3 Pertanyaan

1. Jelaskan mengenai perbedaan 2 method yang dibuat yaitu PangkatBF() dan PangkatDC()!

Jawaban :

Perbedaan utama antara method **pangkatBF()** dan **pangkatDC()** terletak pada pendekatan yang digunakan dalam menghitung pangkat suatu bilangan. Method **pangkatBF()** menggunakan pendekatan iteratif dengan perulangan **for**, sedangkan method **pangkatDC()** menggunakan pendekatan rekursif dengan memecah perhitungan pangkat menjadi perhitungan yang lebih kecil, memanfaatkan sifat simetris pangkat untuk mengurangi jumlah perkalian yang diperlukan.

2. Apakah tahap combine sudah termasuk dalam kode tersebut? Tunjukkan!

Jawaban :

Sudah.

```

if (n % 2 == 0)
{
    return hasil * hasil;
} else {
    return hasil * hasil * a;
}

```

3. Modifikasi kode program tersebut, anggap proses pengisian atribut dilakukan dengan konstruktor.

Kode Program Hasil Modifikasi:

- Class Pangkat :

```

package minggu5;

public class Pangkat {
    public int nilai, pangkat;

    // Konstruktor untuk inisialisasi nilai dan pangkat
    public Pangkat(int nilai, int pangkat) {
        this.nilai = nilai;
        this.pangkat = pangkat;
    }
}

```

```

int pangkatBF(int a, int n) {
    int hasil = 1;
    for (int i = 0; i < n; i++) {
        hasil *= a;
    }
    return hasil;
}

int pangkatDC(int a, int n) {
    if (n == 0) {
        return 1;
    } else {
        int hasil = pangkatDC(a, n / 2);
        if (n % 2 == 0)
        {
            return hasil * hasil;
        } else {
            return hasil * hasil * a;
        }
    }
}
}

```

- Class MainPangkat:

```

package minggu5;

import java.util.Scanner;

public class MainPangkat {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("=====");
        System.out.println("Masukkan jumlah elemen yang dihitung: ");
        int elemen = sc.nextInt();

        Pangkat[] png = new Pangkat[elemen];
        for (int i = 0; i < elemen; i++) {
            System.out.println("Masukkan nilai yang hendak
dipangkatkan:");
            int nilai = sc.nextInt();
            System.out.println("Masukkan nilai pemangkat:");
            int pangkat = sc.nextInt();
            png[i] = new Pangkat(nilai, pangkat); // Membuat objek
Pangkat dengan konstruktor
        }

        System.out.println("HASIL PANGKAT - BRUTE FORCE");
        for (int i = 0; i < elemen; i++) {
            System.out.println("Hasil dari " + png[i].nilai + " pangkat
" + png[i].pangkat + " adalah "
+ png[i].pangkatBF(png[i].nilai, png[i].pangkat));
        }
        System.out.println("HASIL PANGKAT - DIVIDE AND CONQUER");
        for (int i = 0; i < elemen; i++) {
            System.out.println("Hasil dari " + png[i].nilai + " pangkat
" + png[i].pangkat + " adalah "

```



```

        + png[i].pangkatDC(png[i].nilai, png[i].pangkat));
    }
}
}

```

- Output :

```

=====
Masukkan jumlah elemen yang dihitung:
2
Masukkan nilai yang hendak dipangkatkan:
6
Masukkan nilai pemangkat:
2
Masukkan nilai yang hendak dipangkatkan:
4
Masukkan nilai pemangkat:
3
HASIL PANGKAT - BRUTE FORCE
Hasil dari 6 pangkat 2 adalah 36
Hasil dari 4 pangkat 3 adalah 64
HASIL PANGKAT - DIVIDE AND CONQUER
Hasil dari 6 pangkat 2 adalah 36
Hasil dari 4 pangkat 3 adalah 64

```

4. Tambahkan menu agar salah satu method yang terpilih saja yang akan dijalankan menggunakan switch-case!

Kode Program :

```

package minggu5;

import java.util.Scanner;

public class MainPangkat {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("=====");
        System.out.println("Masukkan jumlah elemen yang dihitung:");

        int elemen = sc.nextInt();

        Pangkat[] png = new Pangkat[elemen];
        for (int i = 0; i < elemen; i++) {
            System.out.println("Masukkan nilai yang hendak dipangkatkan:");
            int nilai = sc.nextInt();
            System.out.println("Masukkan nilai pemangkat:");
            int pangkat = sc.nextInt();
            png[i] = new Pangkat(nilai, pangkat); // Membuat objek Pangkat dengan konstruktor
        }

        System.out.println("Pilih method yang ingin dijalankan:");
        System.out.println("1. Pangkat Brute Force");
        System.out.println("2. Pangkat Divide and Conquer");
        int pilihan = sc.nextInt();
    }
}

```

```

        switch (pilihan) {
            case 1:
                System.out.println("HASIL PANGKAT - BRUTE FORCE");
                for (int i = 0; i < elemen; i++) {
                    System.out.println("Hasil dari " + png[i].nilai
+ " pangkat " + png[i].pangkat + " adalah "
                                + png[i].pangkatBF(png[i].nilai,
png[i].pangkat));
                }
                break;
            case 2:
                System.out.println("HASIL PANGKAT - DIVIDE AND
CONQUER");
                for (int i = 0; i < elemen; i++) {
                    System.out.println("Hasil dari " + png[i].nilai
+ " pangkat " + png[i].pangkat + " adalah "
                                + png[i].pangkatDC(png[i].nilai,
png[i].pangkat));
                }
                break;
            default:
                System.out.println("Pilihan tidak valid");
                break;
        }
    }
}

```

Output :

```

=====
Masukkan jumlah elemen yang dihitung:
2
Masukkan nilai yang hendak dipangkatkan:
6
Masukkan nilai pemangkat:
2
Masukkan nilai yang hendak dipangkatkan:
4
Masukkan nilai pemangkat:
3
Pilih method yang ingin dijalankan:
1. Pangkat Brute Force
2. Pangkat Divide and Conquer
1
HASIL PANGKAT - BRUTE FORCE
Hasil dari 6 pangkat 2 adalah 36
Hasil dari 4 pangkat 3 adalah 64
PS D:\Struktur Data dan Algoritma\BruteF

```

4.4 Menghitung Sum Array dengan Algoritma Brute Force dan Divide and Conquer

Di dalam percobaan ini, kita akan mempraktekkan bagaimana proses *divide*, *conquer*, dan *combine* diterapkan pada studi kasus penjumlahan keuntungan suatu perusahaan dalam beberapa bulan.

4.4.1 Langkah-langkah Percobaan

Kode Program :

- Class Sum :

```

package minggu5;

public class Sum {
    public int elemen;
    public double keuntungan[];
    public double total;

    Sum(int elemen) {
        this.elemen = elemen;
        this.keuntungan = new double[elemen];
        this.total = 0;
    }

    double totalBF(double arr[]) {
        for (int i = 0; i < elemen; i++) {
            total = total + arr[i];
        }
        return total;
    }

    double totalDC(double arr[], int l, int r) {
        if (l == r) {
            return arr[l];
        } else if (l < r) {
            int mid = (l + r) / 2;
            double lsum = totalDC(arr, l, mid - 1);
            double rsum = totalDC(arr, mid + 1, r);
            return lsum + rsum + arr[mid];
        }
        return 0;
    }
}

```

- Class MainSum :

```

package minggu5;

import java.util.Scanner;

public class MainSum {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("=====
=====");
        System.out.println("Program Menghitung Keuntungan Total (Satuan
Juta. Misal 5.9)");
        System.out.println("Masukan jumlah bulan : ");
        int elm = sc.nextInt();

        Sum sm = new Sum(elm);
        System.out.println("=====
=====");
        for (int i = 0; i < sm.elemen; i++) {
            System.out.print("Masukan untung bulan ke - " + (i + 1) + "
= ");
            sm.keuntungan[i] = sc.nextDouble();
        }
    }
}

```

```

        System.out.println("=====
=====");
        System.out.println("Algoritma Brute Force");
        System.out.println("Total keuntungan perusahaan selama " +
sm.elemen + " bulan adalah = " + sm.totalBF(sm.keuntungan));
        System.out.println("=====
=====");
        System.out.println("Algoritma Divide Conquer");
        System.out.println("Total keuntungan perusahaan selama " +
sm.elemen + " bulan adalah = " + sm.totalDC(sm.keuntungan, 0, sm.elemen-
1));
    }
}

```

4.4.2 Verifikasi Hasil Percobaan

```

BruteForceDivideConquer_427ec\bin\minggus.MainSum
=====
Program Menghitung Keuntungan Total (Satuan Juta. Misal 5.9)
Masukan jumlah bulan :
5
=====
Masukan untung bulan ke - 1 = 8.5
Masukan untung bulan ke - 2 = 9.54
Masukan untung bulan ke - 3 = 7.2
Masukan untung bulan ke - 4 = 9.1
Masukan untung bulan ke - 5 = 6
=====
Algoritma Brute Force
Total keuntungan perusahaan selama 5 bulan adalah = 40.339999999999996
=====
Algoritma Divide Conquer
Total keuntungan perusahaan selama 5 bulan adalah = 39.04
PS D:\Struktur Data dan Algoritma\BruteForceDivideConquer>

```

4.4.3 Pertanyaan

1. Mengapa terdapat formulasi return value berikut? Jelaskan!

```
return lsum+rsum+arr[mid];
```

Jawaban :

Formulasi **return lsum + rsum + arr[mid]**; digunakan dalam algoritma Divide and Conquer untuk menggabungkan hasil dari dua submasalah dan menentukan hasil akhir. Yaitu menambahkan total keuntungan dari submasalah **lsum**, total keuntungan dari submasalah **rsum**, dan total keuntungan dari bulan tengah **arr[mid]** untuk mendapatkan total keuntungan dari seluruh periode bulan yang diproses.

2. Kenapa dibutuhkan variable mid pada method TotalDC()?

Jawaban :

Variabel mid digunakan dalam metode totalDC() untuk menandai posisi tengah dari rentang elemen yang sedang diproses.

3. Program perhitungan keuntungan suatu perusahaan ini hanya untuk satu perusahaan saja.

Bagaimana cara menghitung sekaligus keuntungan beberapa bulan untuk beberapa perusahaan.(Setiap perusahaan bisa saja memiliki jumlah bulan berbeda-beda)?
Buktikan dengan program!

Kode Program : MainSum2.java

```
package minggu5;
import java.util.Scanner;
public class MainSum2 {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("=====
=====");
        System.out.println("Program Menghitung Keuntungan Total
Perusahaan (Satuan Juta. Misal 5.9)");
        System.out.println("Masukkan jumlah perusahaan: ");
        int noperusahaan = sc.nextInt();

        Sum[] perusahaan = new Sum[noperusahaan];
        for (int i = 0; i < noperusahaan; i++) {
            System.out.println("Masukkan jumlah bulan untuk
perusahaan " + (i+1) + ": ");
            int bulan = sc.nextInt();
            perusahaan[i] = new Sum(bulan);

            System.out.println("Masukkan keuntungan per bulan untuk
perusahaan " + (i+1) + ": ");
            for (int j = 0; j < bulan; j++) {
                System.out.print("Masukkan untung bulan ke-" + (j +
1) + ": ");
                perusahaan[i].keuntungan[j] = sc.nextDouble();
            }
        }

        System.out.println("=====
=====");
        for (int i = 0; i < noperusahaan; i++) {
            System.out.println("Total keuntungan perusahaan " +
(i+1) + ": ");
            System.out.println("Algoritma Brute Force");
            System.out.println("Total keuntungan perusahaan selama "
+ perusahaan[i].elemen + " bulan adalah = " +
perusahaan[i].totalBF(perusahaan[i].keuntungan))
;
            System.out.println("Algoritma Divide Conquer");
            System.out.println("Total keuntungan perusahaan selama "
+ perusahaan[i].elemen + " bulan adalah = " +
perusahaan[i].totalDC(perusahaan[i].keuntungan,
0, perusahaan[i].elemen-1));
        }
    }
}
```

Output :

```

=====
Program Menghitung Keuntungan Total Perusahaan (Satuan Juta. Misal 5.9)
Masukkan jumlah perusahaan:
3
Masukkan jumlah bulan untuk perusahaan 1:
2
Masukkan keuntungan per bulan untuk perusahaan 1:
Masukkan untung bulan ke-1: 300000
Masukkan untung bulan ke-2: 200000
Masukkan jumlah bulan untuk perusahaan 2:
1
Masukkan keuntungan per bulan untuk perusahaan 2:
Masukkan untung bulan ke-1: 400000
Masukkan jumlah bulan untuk perusahaan 3:
3
Masukkan keuntungan per bulan untuk perusahaan 3:
Masukkan untung bulan ke-1: 150000
Masukkan untung bulan ke-2: 200000
Masukkan untung bulan ke-3: 100000
=====
Total keuntungan perusahaan 1:
Algoritma Brute Force
Total keuntungan perusahaan selama 2 bulan adalah = 500000.0
Algoritma Divide Conquer
Total keuntungan perusahaan selama 2 bulan adalah = 500000.0
Total keuntungan perusahaan 2:
Algoritma Brute Force
Total keuntungan perusahaan selama 1 bulan adalah = 400000.0
Algoritma Divide Conquer
Total keuntungan perusahaan selama 1 bulan adalah = 400000.0
Total keuntungan perusahaan 3:
Algoritma Brute Force
Total keuntungan perusahaan selama 3 bulan adalah = 450000.0
Algoritma Divide Conquer
Total keuntungan perusahaan selama 3 bulan adalah = 500000.0
PS D:\Struktur Data dan Algoritma\BruteForceDivideConquer>

```

4.5 Latihan Praktikum

1. Sebuah showroom memiliki daftar mobil dengan data sesuai tabel di bawah ini

merk	tipe	tahun	top_acceleration	top_power
BMW	M2 Coupe	2016	6816	728
Ford	Fiesta ST	2014	3921	575
Nissan	370Z	2009	4360	657
Subaru	BRZ	2014	4058	609
Subaru	Impreza WRX STI	2013	6255	703
Toyota	AE86 Trueno	1986	3700	553
Toyota	86/GT86	2014	4180	609
Volkswagen	Golf GTI	2014	4180	631

Tentukan:

- top_acceleration tertinggi menggunakan Divide and Conquer!
- top_acceleration terendah menggunakan Divide and Conquer!
- Rata-rata top_power dari seluruh mobil menggunakan Brute Force!

Kode Program :

- Showroom.java

```

package minggu5;

public class Showroom {
    Showroom[] mobils;
    String merk;
    String tipe;
    int tahun;
    int top_acceleration;
    int top_power;

    public Showroom(Showroom[] mobils) {
        this.mobils = mobils;
    }

    public Showroom(String merk, String tipe, int tahun, int
top_acceleration, int top_power) {
        this.merk = merk;
        this.tipe = tipe;
        this.tahun = tahun;
        this.top_acceleration = top_acceleration;
        this.top_power = top_power;
    }

    public int findMaxAcceleration() {
        return findMaxAcceleration(0, mobils.length - 1);
    }

    private int findMaxAcceleration(int left, int right) {
        if (left == right) {
            return mobils[left].top_acceleration;
        }

        int mid = (left + right) / 2;
        int maxLeft = findMaxAcceleration(left, mid);
        int maxRight = findMaxAcceleration(mid + 1, right);

        return Math.max(maxLeft, maxRight);
    }

    public int findMinAcceleration() {
        return findMinAcceleration(0, mobils.length - 1);
    }

    private int findMinAcceleration(int left, int right) {
        if (left == right) {
            return mobils[left].top_acceleration;
        }

        int mid = (left + right) / 2;
        int minLeft = findMinAcceleration(left, mid);
        int minRight = findMinAcceleration(mid + 1, right);

        return Math.min(minLeft, minRight);
    }

    public double calculateAveragePower() {
        double totalPower = 0;

```

```

        for (Showroom mobil : mobils) {
            totalPower += mobil.top_power;
        }
        return totalPower / mobils.length;
    }
}

```

- MainShowroom.java

```

package minggu5;

public class MainShowroom {
    public static void main(String[] args) {
        Showroom[] mobils = {
            new Showroom("BMW", "M2 Coupe", 2016, 6816, 728),
            new Showroom("Ford", "Fiesta ST", 2014, 3921, 575),
            new Showroom("Nissan", "370Z", 2009, 4360, 657),
            new Showroom("Subaru", "BRZ", 2014, 4058, 609),
            new Showroom("Subaru", "Impreza WRX STI", 2013, 6255,
703),
            new Showroom("Toyota", "AE86 Trueno", 1986, 3700, 553),
            new Showroom("Toyota", "86/GT86", 2014, 4180, 609),
            new Showroom("Volkswagen", "Golf GTI", 2014, 4180, 631)
        };

        Showroom showroom = new Showroom(mobils);

        // a) Mencari top_acceleration tertinggi menggunakan Divide and
Conquer
        int maxAcceleration = showroom.findMaxAcceleration();
        System.out.println("Top Acceleration Tertinggi: " +
maxAcceleration);

        // b) Mencari top_acceleration terendah menggunakan Divide and
Conquer
        int minAcceleration = showroom.findMinAcceleration();
        System.out.println("Top Acceleration Terendah: " +
minAcceleration);

        // c) Menghitung rata-rata top_power dari seluruh mobil
        double averagePower = showroom.calculateAveragePower();
        System.out.println("Rata-rata Top Power: " + averagePower);
    }
}

```

- Output :

```

Top Acceleration Tertinggi: 6816
Top Acceleration Terendah: 3700
Rata-rata Top Power: 633.125

```