

# Tugas Kecil 2 IF2211 Strategi Algoritma

## Semester II tahun 2023/2024

Membangun Kurva Bézier dengan Algoritma Titik Tengah berbasis Divide and Conquer

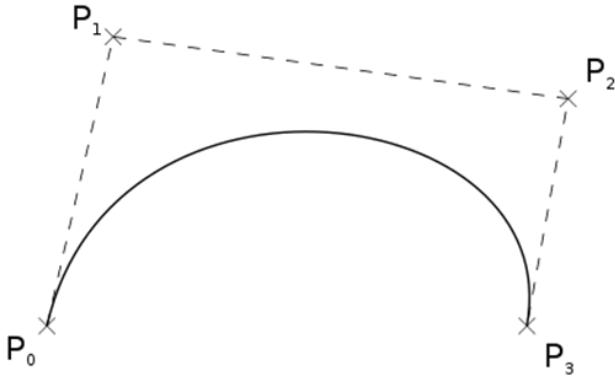


**Disusun Oleh :**  
**13522001 - Mohammad Nugraha Eka Prawira**

**INSTITUT TEKNOLOGI BANDUNG  
BANDUNG  
2024**

## BAB I

### Deskripsi Tugas



**Gambar 1.1. Kurva Bézier Kubik**  
(Sumber: [https://id.wikipedia.org/wiki/Kurva\\_B%C3%A9zier](https://id.wikipedia.org/wiki/Kurva_B%C3%A9zier))

Kurva Bézier adalah kurva halus yang sering digunakan dalam desain grafis, animasi, dan manufaktur. Kurva ini dibuat dengan menghubungkan beberapa titik kontrol, yang menentukan bentuk dan arah kurva. Cara membuatnya cukup mudah, yaitu dengan menentukan titik-titik kontrol dan menghubungkannya dengan kurva. Kurva Bézier memiliki banyak kegunaan dalam kehidupan nyata, seperti *pen tool*, animasi yang halus dan realistik, membuat desain produk yang kompleks dan presisi, dan membuat font yang indah dan unik. Keuntungan menggunakan kurva Bézier adalah kurva ini mudah diubah dan dimanipulasi, sehingga dapat menghasilkan desain yang presisi dan sesuai dengan kebutuhan.

Sebuah kurva Bézier didefinisikan oleh satu set titik kontrol  $P_0$  sampai  $P_n$ , dengan  $n$  disebut order ( $n = 1$  untuk linier,  $n = 2$  untuk kuadrat, dan seterusnya). Titik kontrol pertama dan terakhir selalu menjadi ujung dari kurva, tetapi titik kontrol antara (jika ada) umumnya tidak terletak pada kurva. Pada gambar 1 diatas, titik kontrol pertama adalah  $P_0$ , sedangkan titik kontrol terakhir adalah  $P_3$ . Titik kontrol  $P_1$  dan  $P_2$  disebut sebagai titik kontrol antara yang tidak terletak dalam kurva yang terbentuk.

Mengulas lebih jauh mengenai bagaimana sebuah kurva Bézier bisa terbentuk, misalkan diberikan dua buah titik  $P_0$  dan  $P_1$  yang menjadi titik kontrol, maka kurva Bézier yang terbentuk adalah sebuah garis lurus antara dua titik. Kurva ini disebut dengan **kurva Bézier linier**. Misalkan terdapat sebuah titik  $Q_0$  yang berada pada garis yang dibentuk oleh  $P_0$  dan  $P_1$ , maka posisinya dapat dinyatakan dengan persamaan parametrik berikut.

$$Q_0 = B(t) = (1 - t)P_0 + tP_1, \quad t \in [0, 1]$$

dengan  $t$  dalam fungsi kurva Bézier linier menggambarkan seberapa jauh  $B(t)$  dari  $P_0$  ke  $P_1$ . Misalnya ketika  $t = 0.25$ , maka  $B(t)$  adalah seperempat jalan dari titik  $P_0$  ke  $P_1$ . sehingga seluruh rentang variasi nilai  $t$  dari 0 hingga 1 akan membuat persamaan  $B(t)$  membentuk sebuah garis lurus dari  $P_0$  ke  $P_1$ .

Misalkan selain dua titik sebelumnya ditambahkan sebuah titik baru, sebut saja  $P_2$ , dengan  $P_0$  dan  $P_1$  sebagai titik kontrol awal dan akhir, dan  $P_1$  menjadi titik kontrol antara. Dengan menyatakan titik  $Q_1$  terletak diantara garis yang menghubungkan  $P_1$  dan  $P_2$ , dan membentuk kurva Bézier linier yang berbeda dengan kurva letak  $Q_0$  berada, maka dapat dinyatakan sebuah titik baru,  $R_0$  yang berada diantara garis yang menghubungkan  $Q_0$  dan  $Q_1$  yang bergerak membentuk **kurva Bézier kuadratik** terhadap titik  $P_0$  dan  $P_2$ . Berikut adalah uraian persamaannya.

$$Q_0 = B(t) = (1 - t)P_0 + tP_1, \quad t \in [0, 1]$$

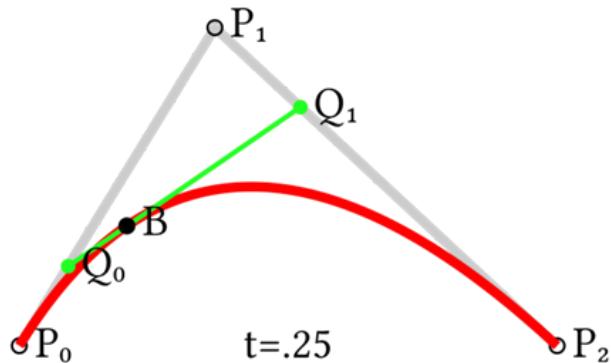
$$Q_1 = B(t) = (1 - t)P_1 + tP_2, \quad t \in [0, 1]$$

$$R_0 = B(t) = (1 - t)Q_0 + tQ_1, \quad t \in [0, 1]$$

dengan melakukan substitusi nilai  $Q_0$  dan  $Q_1$ , maka diperoleh persamaan sebagai berikut.

$$R_0 = B(t) = (1 - t)^2 P_0 + (1 - t)tP_1 + t^2 P_2, \quad t \in [0, 1]$$

Berikut adalah ilustrasi dari kasus diatas.



**Gambar 1.2.** Pembentukan Kurva Bézier Kuadratik.

(Sumber: <https://simonhalliday.com/2017/02/15/quadratic-bezier-curve-demo/>)

## BAB II

### ANALISIS DAN IMPLEMENTASI ALGORITMA BRUTEFORCE

Pada bagian bruteforce algoritma yang diimplementasikan cukup simple, cukup mengimplementasikan rumus yang ada disesuaikan dengan *syntax code programming* yang digunakan yaitu Python. Parameter yang diterima adalah Point yaitu titik koordinat dan jumlah iterasi yang akan digunakan.

$$Q_0 = B(t) = (1 - t)P_0 + tP_1, \quad t \in [0, 1]$$

Fungsi ini merupakan inti dari implementasi algoritma brute force. Berikut adalah penjelasan dari bagian-bagian fungsi tersebut:

1. **Argumen Fungsi:** Fungsi ini menerima empat argumen, yaitu P0, P1, P2, dan iterations. P0, P1, dan P2 adalah titik kontrol kurva Bézier, sedangkan *iterations* adalah jumlah iterasi yang akan dilakukan untuk menghasilkan titik-titik pada kurva Bézier.
2. **Inisialisasi Kurva:** Kurva dimulai dengan titik awal P0.
3. **Perhitungan Titik-titik Kurva:** Selanjutnya, dilakukan iterasi sebanyak *iterations*. Pada setiap iterasi, nilai parameter t dihitung berdasarkan iterasi tersebut, lalu dilakukan perhitungan koordinat x dan y untuk titik kurva pada parameter t menggunakan rumus umum kurva Bézier.
4. **Penyimpanan Titik:** Setiap titik yang dihasilkan ditambahkan ke dalam daftar *curve*.
5. **Penambahan Titik Akhir:** Setelah iterasi selesai, titik terakhir P2 ditambahkan ke dalam daftar *curve*.
6. **Pengembalian Kurva:** Fungsi mengembalikan daftar *curve* yang berisi semua titik pada kurva Bézier

Jumlah iterasi yang dilakukan oleh algoritma ditentukan oleh argumen *iterations*. Semakin besar nilai *iterations*, semakin halus kurva Bézier yang dihasilkan. Parameter *t* digunakan untuk menentukan posisi relatif titik pada kurva Bézier, dengan nilai *t* yang bervariasi dari 0 hingga 1.

Dengan demikian, implementasi ini secara iteratif menghitung koordinat titik pada kurva Bézier menggunakan rumus umumnya, dan menyimpan hasilnya dalam daftar *curve*. Ini memungkinkan kita untuk memperoleh titik-titik pada kurva Bézier dengan menggunakan metode brute force.

## BAB III

### ANALISIS DAN IMPLEMENTASI ALGORITMA DIVIDE AND CONQUER

Implementasi yang disediakan adalah fungsi rekursif untuk membagi kurva Bézier menjadi dua bagian menggunakan pendekatan divide and conquer.

#### 1. Argumen Fungsi:

Fungsi *divideBezier* menerima tiga titik kontrol kurva Bézier, yaitu P0, P1, dan P2, serta jumlah iterasi *iterations*.

#### 2. Kondisional:

Pertama-tama, fungsi memeriksa apakah nilai *iterations* sama dengan 0. Jika ya, itu berarti kita telah mencapai kondisi dasar rekursi, dan fungsi akan mengembalikan daftar yang berisi P0 dan P2, yang merupakan titik awal dan akhir dari segmen kurva Bézier yang telah dibagi.

#### 3. Pembagian Kurva:

Jika nilai *iterations* tidak sama dengan 0, fungsi akan membagi kurva Bézier menjadi dua segmen. Ini dilakukan dengan cara:

- Menemukan titik-titik tengah dari garis yang menghubungkan P0 dan P1 (*q0*) serta P1 dan P2 (*q1*) menggunakan fungsi *find\_midpoint*.
- Kemudian, menemukan titik tengah antara *q0* dan *q1* (*r0*).
- Kurva Bézier awal (P0, P1, P2) dibagi menjadi dua segmen: P0, *q0*, dan *r0* untuk segmen kiri dan *r0*, *q1*, dan P2 untuk segmen kanan.
- Rekursi dilakukan pada kedua segmen tersebut untuk membagi mereka lebih lanjut.

#### 4. Pengembalian Hasil:

Setelah rekursi selesai, hasil dari kedua panggilan rekursif dijadikan satu dengan menambahkan titik-titik segmen kiri dengan titik-titik segmen kanan, kecuali titik yang terduplicat (*curve\_right[1:]*). Ini dilakukan karena titik tengah (*r0*) telah dimasukkan dua kali, sekali di akhir segmen kiri dan sekali di awal segmen kanan.

Implementasi ini menggunakan pendekatan rekursif untuk membagi kurva Bézier menjadi segmen-segmen yang lebih kecil, yang pada akhirnya akan menghasilkan daftar titik-titik yang membentuk kurva Bézier. Pendekatan ini memanfaatkan prinsip divide and conquer untuk memecahkan masalah pembagian kurva Bézier menjadi sub-masalah yang lebih sederhana dan kemudian menggabungkan hasilnya.

## BAB IV

## SOURCE CODE

Berikut adalah source code untuk fungsi fungsi yang dibutuhkam :

Berikut adalah code untuk file main yang memanggil fungsi tersebut:

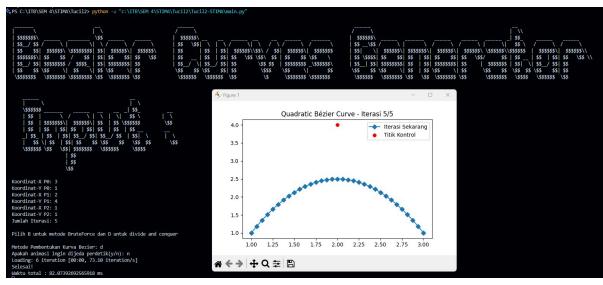
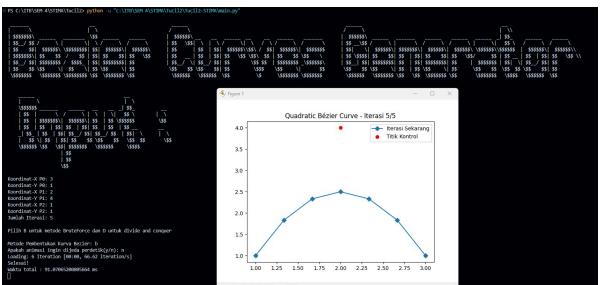
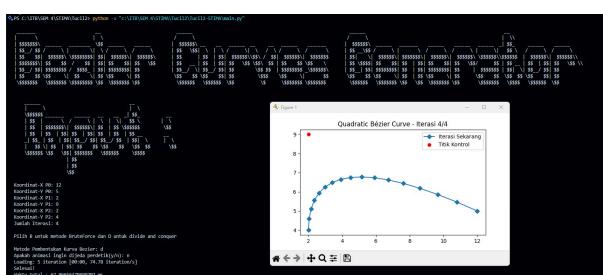
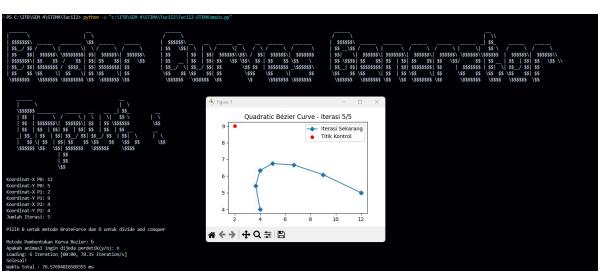
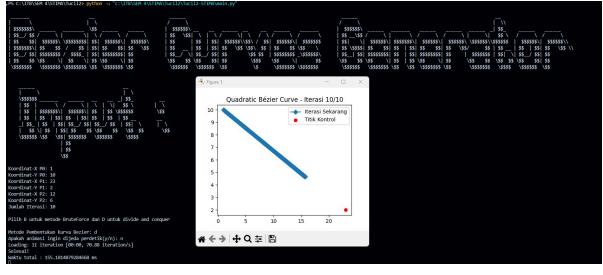
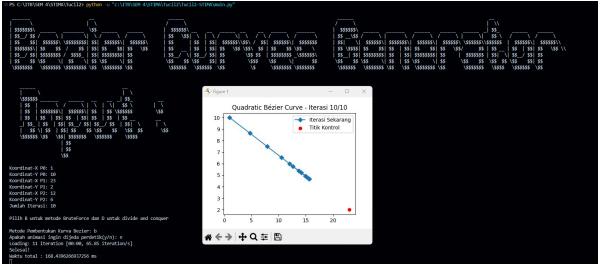
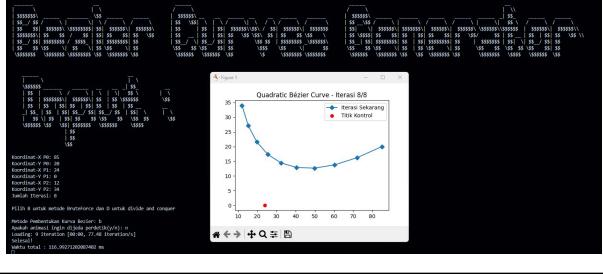
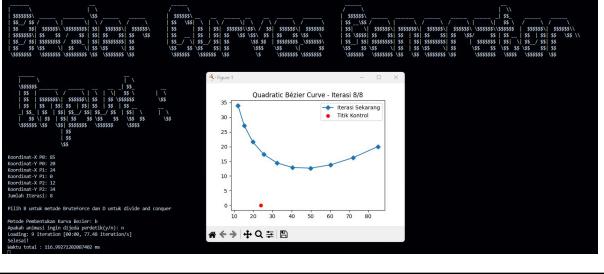


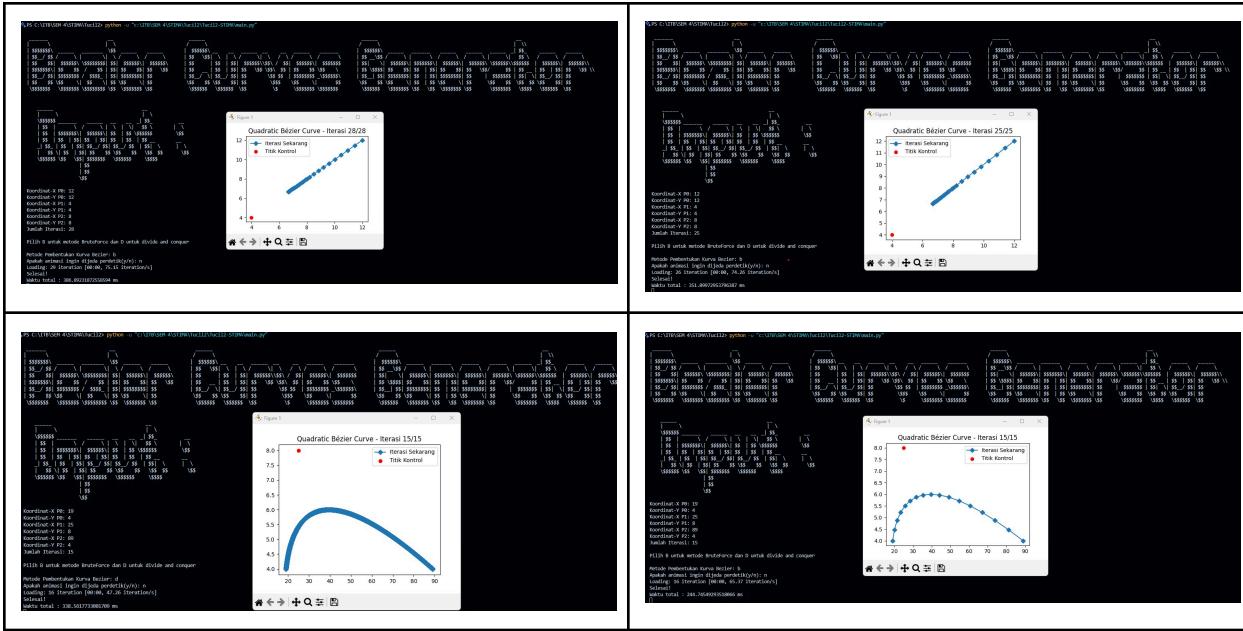
Tucil2 - main.py

```
1 from source import *
2 from tqdm import tqdm
3 import time
4 from tqdm import tqdm
5
6 printWellcome()
7
8 P0,P1,P2,iterations,option,pauses = callInput()
9 plt.ion()
10 fig, ax = plt.subplots()
11
12 progress_bar = tqdm(total=iterations, desc="Loading", unit=" iteration")
13 startTime=time.time()
14 for i in range(iterations + 1):
15     update(i,P0,P1,P2,iterations,option,pauses)
16     progress_bar.update(1)
17 progress_bar.close()
18 print("Selesai!")
19 print(f'Waktu total : {(time.time()-startTime)*1000} ms')
20 plt.ioff()
21 plt.show()
22
```

## BAB V

### HASIL PERCOBAAN

Metode Divide And Conquer	Metode BruteForce
 <p>Pada gambar ini terdapat tampilan terminal dan plot. Terminal menunjukkan proses iterasi dan nilai koordinat titik kontrol. Plot menunjukkan kurva Bezier dengan titik kontrol merah dan titik sekarang biru.</p>	 <p>Pada gambar ini terdapat tampilan terminal dan plot. Terminal menunjukkan proses iterasi dan nilai koordinat titik kontrol. Plot menunjukkan kurva Bezier dengan titik kontrol merah dan titik sekarang biru.</p>
 <p>Pada gambar ini terdapat tampilan terminal dan plot. Terminal menunjukkan proses iterasi dan nilai koordinat titik kontrol. Plot menunjukkan kurva Bezier dengan titik kontrol merah dan titik sekarang biru.</p>	 <p>Pada gambar ini terdapat tampilan terminal dan plot. Terminal menunjukkan proses iterasi dan nilai koordinat titik kontrol. Plot menunjukkan kurva Bezier dengan titik kontrol merah dan titik sekarang biru.</p>
 <p>Pada gambar ini terdapat tampilan terminal dan plot. Terminal menunjukkan proses iterasi dan nilai koordinat titik kontrol. Plot menunjukkan kurva Bezier dengan titik kontrol merah dan titik sekarang biru.</p>	 <p>Pada gambar ini terdapat tampilan terminal dan plot. Terminal menunjukkan proses iterasi dan nilai koordinat titik kontrol. Plot menunjukkan kurva Bezier dengan titik kontrol merah dan titik sekarang biru.</p>
 <p>Pada gambar ini terdapat tampilan terminal dan plot. Terminal menunjukkan proses iterasi dan nilai koordinat titik kontrol. Plot menunjukkan kurva Bezier dengan titik kontrol merah dan titik sekarang biru.</p>	 <p>Pada gambar ini terdapat tampilan terminal dan plot. Terminal menunjukkan proses iterasi dan nilai koordinat titik kontrol. Plot menunjukkan kurva Bezier dengan titik kontrol merah dan titik sekarang biru.</p>



## BAB VI

### HASIL ANALISIS

Analisis perbandingan solusi brute force dengan solusi divide and conquer pada pembagian kurva Bézier dapat memberikan wawasan tentang kelebihan dan kekurangan dari kedua pendekatan tersebut. Pertama-tama, mari kita lihat dari segi kompleksitas algoritma:

#### 1. Solusi Brute Force:

Kompleksitas Waktu: Solusi brute force akan menghasilkan kurva Bézier dengan menghitung setiap titik pada kurva secara langsung menggunakan rumus umum kurva Bézier. Dalam hal ini, kompleksitas waktu adalah  $O(n * m)$ , di mana  $n$  adalah jumlah iterasi dan  $m$  adalah jumlah titik kontrol kurva Bézier.

Kompleksitas Ruang: Solusi ini tidak memerlukan alokasi memori tambahan yang signifikan, karena hanya menyimpan daftar titik-titik yang dihasilkan.

#### 2. Solusi Divide and Conquer:

Kompleksitas Waktu: Solusi divide and conquer membagi kurva Bézier menjadi dua bagian, menyelesaikan setiap bagian secara rekursif, dan kemudian menggabungkan hasilnya. Kompleksitas waktu rekursifnya adalah  $O(\log n)$ , di mana  $n$  adalah jumlah iterasi. Namun, perlu diperhatikan bahwa setiap panggilan rekursif akan membagi kurva menjadi dua bagian, sehingga kompleksitas keseluruhannya masih  $O(n * \log n)$ .

Kompleksitas Ruang: Solusi ini memerlukan alokasi memori tambahan untuk menyimpan titik-titik yang dihasilkan selama proses pembagian dan penggabungan kurva. Namun, kompleksitas ruangnya masih dapat dianggap sebagai  $O(n)$  karena setiap panggilan rekursif hanya menyimpan titik-titik yang diperlukan untuk submasalah tertentu.

Dari segi kompleksitas, solusi divide and conquer memiliki keunggulan dalam hal waktu karena memiliki kompleksitas waktu yang lebih baik dalam kasus terburuk. Namun, solusi ini memerlukan alokasi memori tambahan untuk menyimpan titik-titik yang dihasilkan selama proses pembagian dan penggabungan, sedangkan solusi brute force tidak memerlukan alokasi memori tambahan yang signifikan.

Selain itu, perlu juga dipertimbangkan dalam konteks implementasi dan skala masalah yang dihadapi. Solusi brute force mungkin lebih mudah untuk diimplementasikan dan cukup efisien untuk kasus dengan jumlah titik kontrol yang kecil atau jumlah iterasi yang terbatas. Namun, solusi divide and conquer akan lebih sesuai untuk kasus dengan jumlah titik kontrol yang besar atau ketika waktu eksekusi yang optimal menjadi prioritas utama.

Dengan demikian, pemilihan antara solusi brute force dan solusi divide and conquer tergantung pada karakteristik masalah yang dihadapi, persyaratan performa, dan preferensi implementasi. Walaupun dalam implementasi yang dibuat, algoritma bruteforce nya belum maksimal karena pada hasil iterasi yang sama titik yang dihasilkan lebih sedikit daripada algoritma Divide and Conquer. Hasil titik yang dihasilkan baru akan seimbang jika jumlah iterasi bruteforce lebih banyak sedikit.

## BAB VII

### LAMPIRAN DAN REPOSITORY

Poin	Ya	Tidak
1. Program berhasil dijalankan.	✓	
2. Program dapat melakukan visualisasi kurva Bézier..	✓	
3. Solusi yang diberikan program optimal.	✓	
4. <b>[Bonus]</b> Program dapat membuat kurva untuk n titik kontrol.		✓
5. <b>[Bonus]</b> Program dapat melakukan visualisasi proses pembuatan kurva.		✓

**Github :** <https://github.com/EkaaPrawiraa/Tucil2-STIMA>