LAPORAN TUGAS

GRAFIKA KOMPUTER

(Dosen : *Rio Priantama S.T., M.T.I.*)

Tugas DDA dan Bresenham

Nama : Muhammad Rizal Nurfirdaus

NIM : 20230810088

Kelas : TINFC-2023-04

TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS KUNINGAN

1. **Algoritma DDA**

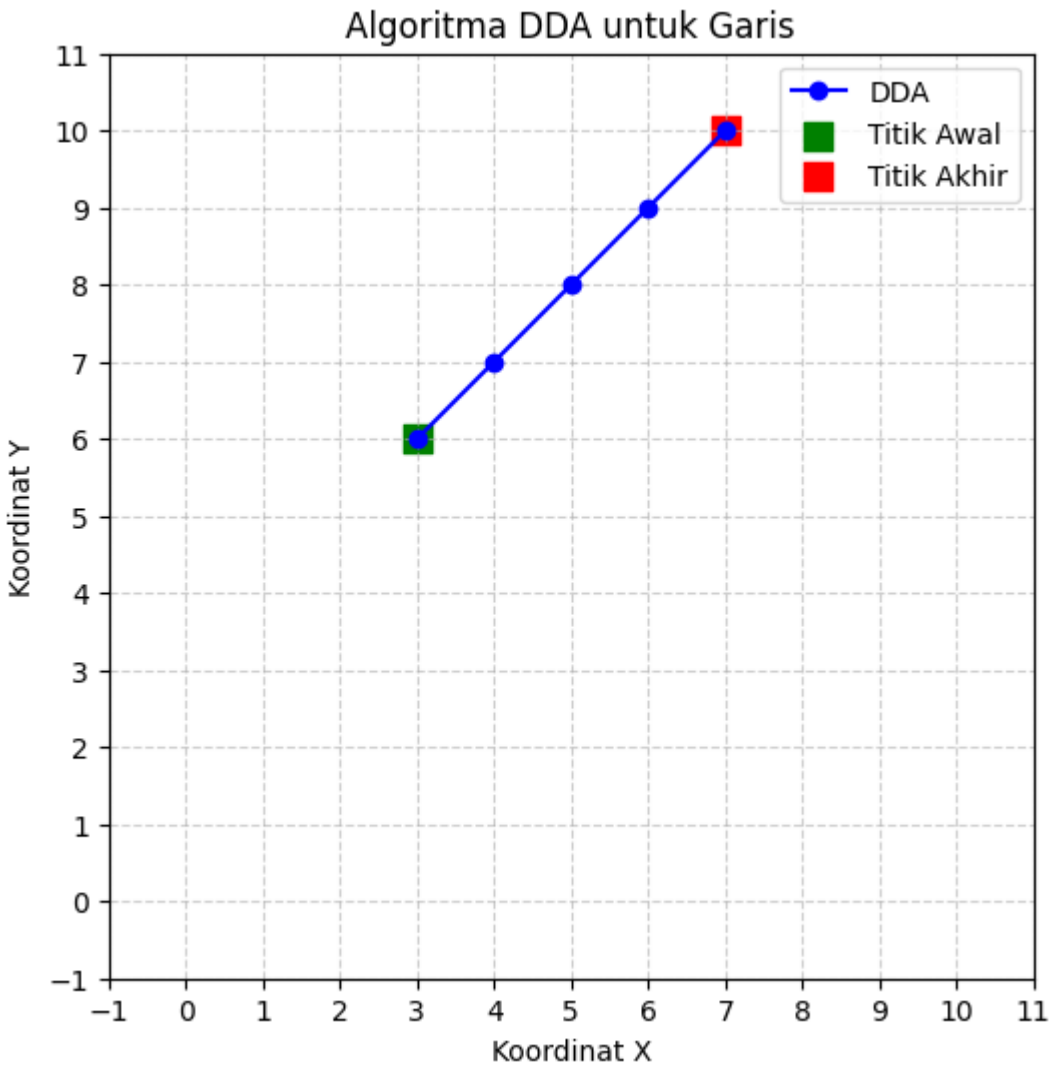
Source Code :

```
# Tugas DDA
import matplotlib.pyplot as plt
import numpy as np

# 1. Preparing
def dda_algorithm(x_start, y_start, x_end, y_end):
    # Hitung selisih
    dx = x_end - x_start
    dy = y_end - y_start

    # Tentukan jumlah langkah (steps)
    steps = int(max(abs(dx), abs(dy)))

    # Hitung increment
    if steps == 0:
        x_increment = 0
        y_increment = 0
    else:
        x_increment = dx / steps
        y_increment = dy / steps

    x = x_start
    y = y_start

    x_coords = []
    y_coords = []

    # Iterasi dan hitung koordinat
    for i in range(steps + 1):
        x_coords.append(round(x))
        y_coords.append(round(y))

        x += x_increment
        y += y_increment

    return x_coords, y_coords

def draw_line(x_coords, y_coords, algorithm_name):
    # 5. Gambar Garis & 6. Tampilkan Grafik
    plt.figure(figsize=(8, 6))

    # Plot koordinat titik-titik
    plt.plot(x_coords, y_coords, marker='o', linestyle='-',
color='blue', label=algorithm_name)
```

```
    # Plot titik awal dan akhir
    plt.scatter([x_coords[0]], [y_coords[0]], color='green',
marker='s', s=100, label='Titik Awal')
    plt.scatter([x_coords[-1]], [y_coords[-1]], color='red',
marker='s', s=100, label='Titik Akhir')

    # Atur sumbu agar terlihat seperti grid pixel
    max_coord = max(max(x_coords), max(y_coords), 10) # Sesuaikan batas
    min_coord = min(min(x_coords), min(y_coords), 0) # Sesuaikan batas

    plt.xlim(min_coord - 1, max_coord + 1)
    plt.ylim(min_coord - 1, max_coord + 1)

    plt.xticks(np.arange(min_coord - 1, max_coord + 2, 1))
    plt.yticks(np.arange(min_coord - 1, max_coord + 2, 1))

    plt.grid(True, linestyle='--', alpha=0.6)
    plt.gca().set_aspect('equal', adjustable='box') # Penting untuk
grafik komputer

    plt.title(f'Algoritma {algorithm_name} untuk Garis')
    plt.xlabel('Koordinat X')
    plt.ylabel('Koordinat Y')
    plt.legend()
    plt.show()

# --- Main Program DDA ---
print("--- Algoritma DDA ---")

# 2. Input titik
try:
    x_a = float(input("Masukkan koordinat x untuk Titik A: "))
    y_a = float(input("Masukkan koordinat y untuk Titik A: "))
    x_b = float(input("Masukkan koordinat x untuk Titik B: "))
    y_b = float(input("Masukkan koordinat y untuk Titik B: "))
except ValueError:
    print("Input harus berupa angka.")
    exit()

# 3. Pilih Arah
direction = input("Pilih arah penggambaran (A ke B atau B ke A): ")

# Tentukan titik awal dan akhir
if direction.lower() == "a ke b":
    x_start, y_start = x_a, y_a
    x_end, y_end = x_b, y_b
elif direction.lower() == "b ke a":
    x_start, y_start = x_b, y_b
```

```
    x_end, y_end = x_a, y_a
else:
    print("Arah tidak valid. Menggunakan 'A ke B' secara default.")
    x_start, y_start = x_a, y_a
    x_end, y_end = x_b, y_b

# 4. Algoritma DDA
x_coords_dda, y_coords_dda = dda_algorithm(x_start, y_start, x_end,
y_end)

print("\nKoordinat titik-titik yang dihitung (DDA):")
for i in range(len(x_coords_dda)):
    print(f"({x_coords_dda[i]}, {y_coords_dda[i]})")

# 5. Gambar Garis & 6. Tampilkan Grafik
draw_line(x_coords_dda, y_coords_dda, "DDA")

# 7. Finish task
print("--- Tugas DDA Selesai ---")
```

Hasil Run :



Analisis : Program di atas merupakan implementasi algoritma **DDA (Digital Differential Analyzer)** untuk menggambar garis antara dua titik dalam bidang koordinat. Program dimulai dengan meminta input dari pengguna berupa koordinat titik **A** dan **B**, serta arah penggambaran garis (A ke B atau B ke A). Berdasarkan input tersebut, program menghitung selisih koordinat (dx dan dy) dan menentukan jumlah langkah (steps) berdasarkan nilai maksimum di antara keduanya. Selanjutnya, program menghitung nilai **inkrementasi** untuk sumbu X dan Y (x_increment dan y_increment) yang digunakan untuk menghasilkan koordinat setiap titik pada garis.

Setiap titik hasil perhitungan disimpan dalam daftar, dibulatkan agar sesuai dengan posisi piksel, kemudian digambar menggunakan **Matplotlib**. Garis utama divisualisasikan dengan warna biru, titik awal ditandai warna hijau, dan titik akhir warna merah, lengkap dengan grid dan skala koordinat agar tampak seperti tampilan raster grafika komputer.

Secara keseluruhan, program ini **benar dan rapi**, serta sudah modular karena fungsi perhitungan dan fungsi gambar dipisahkan (dda_algorithm() dan draw_line()). Algoritma DDA ini bekerja dengan perhitungan *floating point*, sehingga mudah dipahami dan cocok untuk pembelajaran dasar grafika komputer. Namun, dibanding algoritma **Bresenham**, metode DDA sedikit kurang efisien karena melibatkan pembulatan dan operasi pecahan.

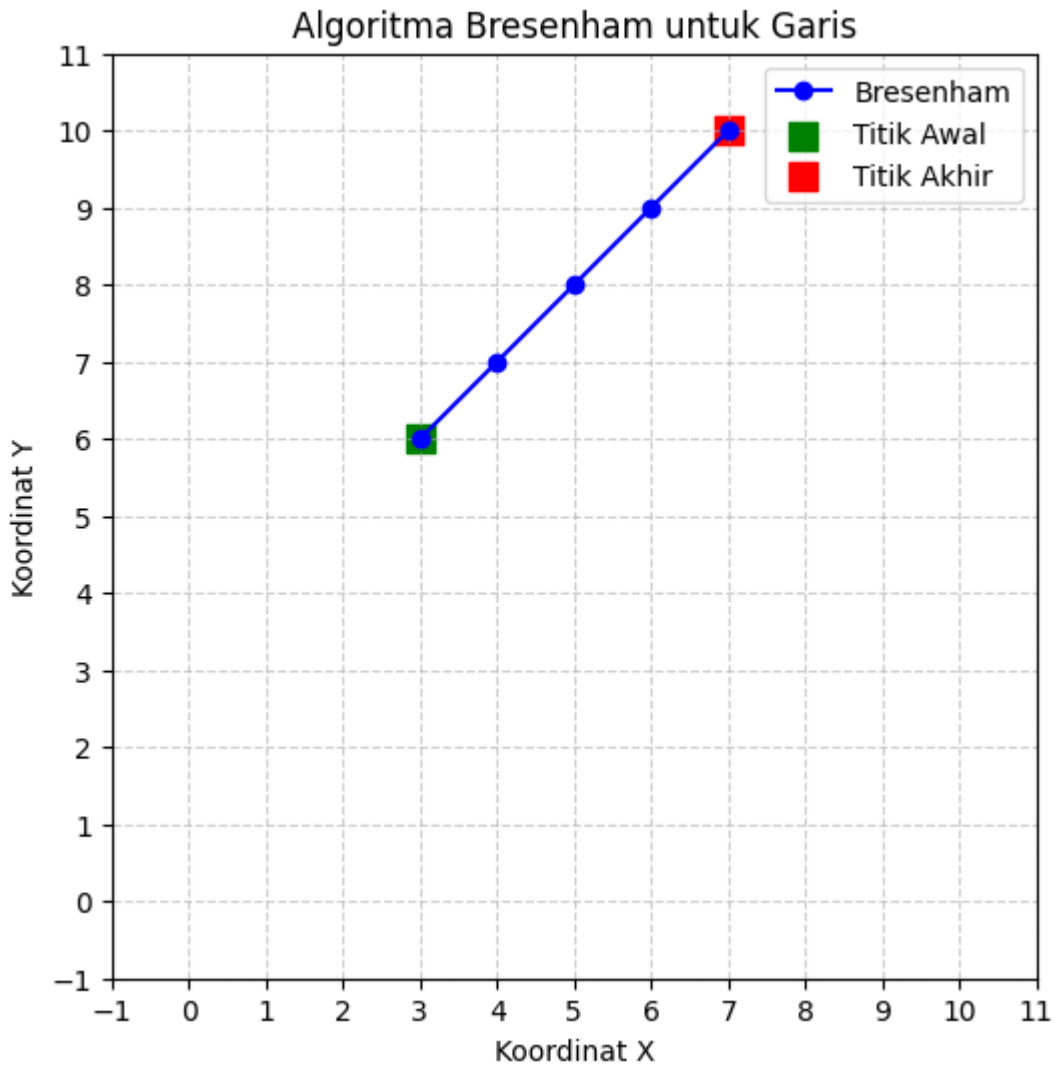
2. Tugas Bressenhem

Source Code :

```
# Tugas Bresenham
import matplotlib.pyplot as plt
import numpy as np

# 1. Preparing
def bresenham_algorithm(x1, y1, x2, y2):
    x_coords = []
    y_coords = []

    dx = abs(x2 - x1)
    dy = abs(y2 - y1)

    # Tentukan arah langkah (step)
    sx = 1 if x1 < x2 else -1
    sy = 1 if y1 < y2 else -1

    err = dx - dy

    current_x, current_y = x1, y1

    while True:
        x_coords.append(current_x)
        y_coords.append(current_y)

        if current_x == x2 and current_y == y2:
            break

        e2 = 2 * err

        # Perbarui eror dan koordinat x
        if e2 > -dy:
            err -= dy
            current_x += sx

        # Perbarui eror dan koordinat y
        if e2 < dx:
            err += dx
            current_y += sy

    return x_coords, y_coords

# --- Main Program Bresenham (Lanjutkan dari Input DDA) ---

# Tentukan titik awal dan akhir (gunakan yang sudah di-input untuk DDA)
```

```
# Catatan: Algoritma Bresenham biasanya tidak memerlukan penentuan arah
(A ke B/B ke A)
# karena ia menangani semua oktan, tetapi kita menggunakan input DDA
untuk konsistensi.
x_start_b, y_start_b = int(x_a), int(y_a)
x_end_b, y_end_b = int(x_b), int(y_b)

# 4. Algoritma Bresenham
print("\n--- Algoritma Bresenham ---")
x_coords_bres, y_coords_bres = bresenham_algorithm(x_start_b,
y_start_b, x_end_b, y_end_b)

print("\nKoordinat titik-titik yang dihitung (Bresenham):")
for i in range(len(x_coords_bres)):
    print(f"({x_coords_bres[i]}, {y_coords_bres[i]})")

# 5. Gambar Garis & 6. Tampilkan Grafik
draw_line(x_coords_bres, y_coords_bres, "Bresenham")

# 7. Finish task
print("--- Tugas Bresenham Selesai ---")
```

Hasil Run :



Analisis : Program di atas merupakan implementasi **algoritma Bresenham** untuk menggambar garis antara dua titik koordinat pada bidang kartesius. Program ini dimulai dengan mendefinisikan fungsi bresenham_algorithm() yang menghitung titik-titik garis secara **integer murni**, tanpa menggunakan operasi pecahan (*floating point*), sehingga efisien dan cocok untuk penggambaran pada layar piksel.

Algoritma bekerja dengan membandingkan nilai perbedaan dx (jarak horizontal) dan dy (jarak vertikal), kemudian menggunakan variabel kesalahan err untuk menentukan kapan sumbu Y harus dinaikkan atau diturunkan saat sumbu X bertambah. Setiap koordinat hasil perhitungan disimpan ke dalam dua daftar (x_coords dan y_coords).

Setelah semua titik diperoleh, hasilnya digambar menggunakan fungsi draw_line(), menampilkan garis biru yang terbentuk dari titik-titik hasil algoritma, dengan titik awal berwarna hijau dan titik akhir berwarna merah.

Secara keseluruhan, program ini **benar dan efisien**, menampilkan hasil yang akurat sesuai prinsip **algoritma Bresenham**, yaitu menggambar garis dengan perhitungan cepat dan presisi tinggi menggunakan operasi bilangan bulat.