**Laporan Tugas Kecil Strategi Algoritma 2**



**Divide And Conquer pada Kurva Bezier**

Oleh  
  
Jimly Nur Arif

13522123

Informatik ITB

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG

# Analisis dan implementasi dalam algoritma brute force sebagai algoritma pembanding

Source code sebagai berikut

import numpy as np

import matplotlib.pyplot as plt

import time

def calculateBezierPoint(control\_points, t):

    n = len(control\_points) - 1

    x = 0

    y = 0

    for i, point in enumerate(control\_points):

        coeff = np.math.comb(n, i) \* (1 - t)\*\*(n - i) \* t\*\*i

        x += coeff \* point[0]

        y += coeff \* point[1]

    return x, y

def bruteForceBezier(control\_points, num\_iterations):

    curve\_points = []

    for i in range(num\_iterations + 1):

        t = i / num\_iterations

        curve\_points.append(calculateBezierPoint(control\_points, t))

    return curve\_points

def plotBezierCurve(control\_points, curve\_points):

    control\_x = [point[0] for point in control\_points]

    control\_y = [point[1] for point in control\_points]

    curve\_x = [point[0] for point in curve\_points]

    curve\_y = [point[1] for point in curve\_points]

    plt.plot(curve\_x, curve\_y, label="Bezier Curve", color="blue")

    plt.scatter(control\_x, control\_y, color="red", label="Control Points")

    plt.xlabel("X")

    plt.ylabel("Y")

    plt.title("Bezier Curve (Brute Force)")

    plt.legend()

    plt.grid(True)

    plt.show()

def main():

    p0\_x, p0\_y = map(float, input("Masukkan koordinat titik awal (pisahkan dengan spasi): ").split())

    p1\_x, p1\_y = map(float, input("Masukkan koordinat titik kontrol (pisahkan dengan spasi): ").split())

    p2\_x, p2\_y = map(float, input("Masukkan koordinat titik akhir (pisahkan dengan spasi): ").split())

    num\_iterations = int(input("Masukkan jumlah iterasi: "))

    start\_time = time.time()

    control\_points = [(p0\_x, p0\_y), (p1\_x, p1\_y), (p2\_x, p2\_y)]

    curve\_points = bruteForceBezier(control\_points, num\_iterations)

    end\_time = time.time()

    runtime = end\_time - start\_time

    print(f"Waktu eksekusi: {runtime} detik")

    plotBezierCurve(control\_points, curve\_points)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Analisis BruteForce dengan Divide and Conquer

Brute Force:

Pendekatan Brute Force melibatkan penghitungan langsung titik-titik kurva Bézier dengan menggunakan persamaan umum kurva. Proses ini melibatkan evaluasi polinomial pada setiap titik parameter t dari 0 hingga 1 untuk menemukan koordinat (x, y) dari kurva pada setiap titik t. Pendekatan ini sederhana dan mudah dipahami, namun efisiensi waktu eksekusinya cenderung menurun secara eksponensial seiring dengan peningkatan jumlah titik yang harus dihitung.

Divide and Conquer:

Pendekatan Divide and Conquer, di sisi lain, membagi kurva Bézier menjadi segmen-segmen yang lebih kecil dan kemudian menghitung titik-titiknya secara terpisah. Misalnya, dalam kasus kurva Bézier kubik, kurva dibagi menjadi dua segmen kubik yang lebih kecil dengan dua titik kontrol di tengahnya. Kemudian, titik-titik kurva dihitung untuk setiap segmen secara terpisah. Pendekatan ini meningkatkan efisiensi karena mengurangi jumlah titik yang harus dievaluasi pada setiap iterasi, terutama untuk kurva dengan derajat yang tinggi.

Perbandingan:

Kompleksitas Waktu:

Brute Force: Kompleksitas waktu Brute Force meningkat secara eksponensial seiring dengan peningkatan derajat kurva atau jumlah titik yang dievaluasi.

Divide and Conquer: Kompleksitas waktu Divide and Conquer biasanya lebih baik daripada Brute Force karena membagi masalah menjadi sub-masalah yang lebih kecil, mengurangi jumlah operasi yang diperlukan.

Implementasi dan kemudahan:

Brute Force: Pendekatan yang mudah dipahami dan diimplementasikan, terutama untuk kurva Bézier dengan derajat rendah.

Divide and Conquer: Memerlukan pemahaman yang lebih mendalam tentang kurva Bézier dan kompleksitas algoritma bagi beberapa orang, tetapi bisa menjadi lebih efisien dalam situasi tertentu.

Efisiensi:

Brute Force: Biasanya kurang efisien daripada Divide and Conquer, terutama untuk kurva dengan jumlah titik yang besar.

Divide and Conquer: Lebih efisien dalam banyak kasus karena membagi masalah menjadi sub-masalah yang lebih kecil.

Penerapan:

Brute Force: Sederhana dan cocok untuk implementasi cepat pada kasus sederhana atau untuk keperluan prototyping.

Divide and Conquer: Lebih cocok untuk kasus-kasus di mana efisiensi waktu eksekusi menjadi faktor yang sangat penting, terutama untuk kurva dengan jumlah titik yang besar.

Dalam konteks kurva Bézier, khususnya untuk kurva dengan derajat yang tinggi atau jumlah titik yang besar, pendekatan Divide and Conquer cenderung menjadi pilihan yang lebih efisien. Namun, implementasi Brute Force masih dapat memberikan hasil yang memadai untuk kasus sederhana.

# Analisis dan implementasi dalam algoritma divide and conquer

Source Code

'''

Programmer = Jimly Nur Arif

NIM = 13519097

Objective: Divide and Conquer (DnC) for Quadratic Bezier Curve

This is an assignment in the algorithmic strategy course

IF2211 Bandung Institute of Technology

Spesification = bit.ly/tucil2stima24

revision = bit.ly/qnastima24

group = bit.ly/kelompoktucil2stima24

'''

import matplotlib.pyplot as plt

import time

def divide\_quadratic\_bezier(p0, p1, p2, num\_segments):

    if num\_segments == 1:

        mid\_point\_1 = ((p0[0] + p1[0]) / 2, (p0[1] + p1[1]) / 2)

        mid\_point\_2 = ((p1[0] + p2[0]) / 2, (p1[1] + p2[1]) / 2)

        mid\_point\_3 = ((mid\_point\_1[0] + mid\_point\_2[0]) / 2, (mid\_point\_1[1] + mid\_point\_2[1]) / 2)

        return [p0, mid\_point\_3, p2]

    else:

        # midpoint yang baru

        mid\_point\_1 = ((p0[0] + p1[0]) / 2, (p0[1] + p1[1]) / 2)

        mid\_point\_2 = ((p1[0] + p2[0]) / 2, (p1[1] + p2[1]) / 2)

        mid\_point\_3 = ((mid\_point\_1[0] + mid\_point\_2[0]) / 2, (mid\_point\_1[1] + mid\_point\_2[1]) / 2)

        left\_segments = divide\_quadratic\_bezier(p0, mid\_point\_1, mid\_point\_3, num\_segments - 1)

        right\_segments = divide\_quadratic\_bezier(mid\_point\_3, mid\_point\_2, p2, num\_segments - 1)

        return left\_segments + right\_segments[1:]

def draw\_bezier(p0, p1, p2, iterations):

    if iterations == 0:

        plt.plot(\*p0, 'ro')

        plt.plot(\*p1, 'ro')

        plt.plot(\*p2, 'ro')

        return

    q0 = ((p0[0] + p1[0]) / 2, (p0[1] + p1[1]) / 2)

    q1 = ((p1[0] + p2[0]) / 2, (p1[1] + p2[1]) / 2)

    r0 = ((q0[0] + q1[0]) / 2, (q0[1] + q1[1]) / 2)

    plt.plot(\*p0, 'ro')

    plt.plot(\*p1, 'ro')

    plt.plot(\*p2, 'ro')

    plt.plot([p0[0], p1[0]], [p0[1], p1[1]], 'r--')

    plt.pause(0.12)

    plt.plot([p1[0], p2[0]], [p1[1], p2[1]], 'r--')

    plt.pause(0.12)

    plt.plot(\*q0, 'gs')

    plt.pause(0.12)

    plt.plot(\*q1, 'gs')

    plt.pause(0.12)

    plt.plot([q0[0], q1[0]], [q0[1], q1[1]], 'g--')

    plt.pause(0.12)

    plt.plot(\*r0, 'bs')

    plt.pause(0.12)

    plt.plot([p0[0], r0[0]], [p0[1], r0[1]], 'b--')

    plt.pause(0.12)

    plt.plot([r0[0], p2[0]], [r0[1], p2[1]], 'b--')

    plt.pause(0.12)

    draw\_bezier(p0, q0, r0, iterations - 1)

    draw\_bezier(r0, q1, p2, iterations - 1)

x0, y0 = map(int, input("Masukkan koordinat titik awal (pisahkan dengan spasi): ").split())

x1, y1 = map(int, input("Masukkan koordinat titik kontrol (pisahkan dengan spasi): ").split())

x2, y2 = map(int, input("Masukkan koordinat titik akhir (pisahkan dengan spasi): ").split())

iterations = int(input("Masukkan jumlah iterasi: "))

p0 = (x0, y0)

p1 = (x1, y1)

p2 = (x2, y2)

start\_time = time.time()

curve\_points = divide\_quadratic\_bezier(p0, p1, p2, iterations)

end\_time = time.time()

runtime = end\_time - start\_time

print(f"Waktu eksekusi: {runtime} detik")

plt.figure(figsize=(8, 6))

for i in range(iterations + 1):

    # plt.clf()  # Clear the previous plot

    plt.title(f"Iterasi {i}")

    draw\_bezier((x0, y0), (x1, y1), (x2, y2), i)

    plt.pause(0.4)  # Pause for 1 second

    if i < iterations:

        plt.draw()  # Redraw the plot

plt.show()

Analisis

Proses dimulai dengan fungsi divide\_quadratic\_bezier, yang menerima tiga titik kontrol dari kurva Bézier quadratic (p0, p1, dan p2) serta jumlah segmentasi yang diinginkan (num\_segments). Pertama, fungsi memeriksa apakah jumlah segmen sudah mencapai satu. Jika ya, fungsi menghitung titik-titik kontrol untuk segmentasi terakhir dan mengembalikan titik-titik tersebut.

Jika jumlah segmen lebih besar dari satu, fungsi akan terus membagi kurva menjadi segmen-segmen yang lebih kecil dengan cara menemukan titik-titik tengah setiap garis yang menghubungkan titik kontrol. Kemudian, kurva dibagi menjadi dua bagian, yaitu bagian kiri dan bagian kanan, dengan menggunakan titik-titik tengah yang baru ditemukan sebagai titik kontrol. Proses ini dilakukan secara rekursif sampai mencapai jumlah segmen yang diinginkan.

Kemudian, dalam fungsi draw\_bezier, kurva Bézier digambar dengan menggunakan pendekatan yang sama seperti sebelumnya, yaitu dengan membagi kurva menjadi segmen-segmen yang lebih kecil dan menggambar garis-garis pendukung di antara titik-titik kontrol.

Secara keseluruhan, konsep "Divide and Conquer" dalam kode tersebut melibatkan membagi masalah besar (menggambar kurva Bézier) menjadi sub-masalah yang lebih kecil (menggambar segmen-segmen kurva Bézier) dan menyelesaikannya secara terpisah, sehingga meningkatkan efisiensi dan kejelasan kode.

# Tangkapan Layar

A close up of a white background

Description automatically generated

A diagram of a triangle

Description automatically generated

A white background with black text

Description automatically generatedA diagram of a triangle

Description automatically generatedA close up of words

Description automatically generatedA graph with lines and dots

Description automatically generatedA white background with black text

Description automatically generatedA graph with lines and dots

Description automatically generatedA close up of words

Description automatically generatedA diagram of a triangle

Description automatically generatedA close up of a white background

Description automatically generatedA diagram of a triangle

Description automatically generated

A screen shot of a computer code

Description automatically generated

# Hasil analisis perbandingan solusi brute force dengan divide and conquer

Terbukti divide and conquer sangat efisien dalam hal membuat kurva bezier. Bahkan kode berjalan dengan runtime yang sangat kecil mendekati nol.

Pendekatan Brute Force melibatkan penghitungan langsung titik-titik kurva Bézier dengan menggunakan persamaan umum kurva. Proses ini melibatkan evaluasi polinomial pada setiap titik parameter t dari 0 hingga 1 untuk menemukan koordinat (x, y) dari kurva pada setiap titik t. Pendekatan ini sederhana dan mudah dipahami, namun efisiensi waktu eksekusinya cenderung menurun secara eksponensial seiring dengan peningkatan jumlah titik yang harus dihitung.

Divide and Conquer:

Pendekatan Divide and Conquer, di sisi lain, membagi kurva Bézier menjadi segmen-segmen yang lebih kecil dan kemudian menghitung titik-titiknya secara terpisah. Misalnya, dalam kasus kurva Bézier kubik, kurva dibagi menjadi dua segmen kubik yang lebih kecil dengan dua titik kontrol di tengahnya. Kemudian, titik-titik kurva dihitung untuk setiap segmen secara terpisah. Pendekatan ini meningkatkan efisiensi karena mengurangi jumlah titik yang harus dievaluasi pada setiap iterasi, terutama untuk kurva dengan derajat yang tinggi.

Kompleksitas Algoritma

Setiap panggilan rekursif menghasilkan dua panggilan rekursif baru, kecuali pada kasus dasar ketika num\_segments == 1.

Dalam setiap panggilan rekursif, terdapat beberapa operasi sederhana untuk menghitung titik-titik tengah antara titik kontrol.

Karena setiap panggilan rekursif menghasilkan dua panggilan rekursif baru, maka jumlah total panggilan rekursif adalah $2^{num\_segments}$. Kompleksitas waktu algoritma ini adalah eksponensial terhadap num\_segments.

# Bonus

Bonus menggunakan matplotlip yang menlakukan iterasi dari 1 sampai ke n menggunakan matplotlib

# Repository

https://github.com/jimlynurarif/Tucil2\_13522123