Analisis Algoritma Divide and Conquer dan Program Dinamis dalam Pembentukan *Smooth Shape* pada Aplikasi Desain

Melati Anggraini - 13522035
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): melatianggraini40@gmail.com

***Abstract*— Pembentukan bentuk halus (smooth shape) merupakan elemen penting dalam desain grafis yang mempengaruhi kualitas estetika dan fungsionalitas desain Memungkinkan designer menciptakan bentuk yang lebih halus dan presisi tinggi. Hal ini tidak hanya meningkatkan kualitas visual produk akhir tetapi juga meningkatkan produktivitas dan kreativitas desainer dengan menyediakan alat yang sesuai untuk kebutuhan spesifik proyek mereka. Program akan dibuat dengan membentuk kurva Bezier di setiap control point masukan pengguna. Pembentukan kurvanya melibatkan dua algoritma yaitu Divide and Conquer dan Program Dinamis. Makalah ini akan menganalisis dan membandingkan dua pendekatan algoritmik tersebut.**

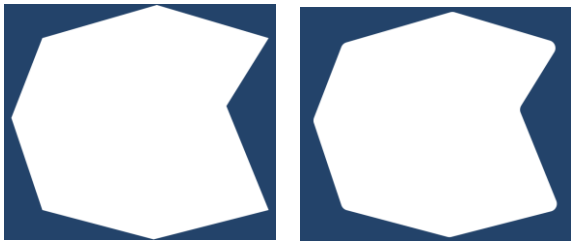
Keywords—desain grafis; bentuk halus; program dinamis; divide and conquer, kurva bezier

I. PENDAHULUAN

Dalam perkembangan teknologi desain grafis, pembentukan bentuk halus (smooth shape) merupakan salah satu aspek penting yang perlu diperhatikan. Bentuk halus diperlukan untuk menghasilkan desain yang estetis dan fungsional, baik dalam pembuatan logo, animasi, maupun berbagai elemen grafis lainnya. Untuk mencapai bentuk halus yang optimal, diperlukan algoritma yang efisien dan efektif. Dua pendekatan algoritmik yang sering digunakan dalam pembentukan bentuk halus adalah algoritma Divide and Conquer dan Program Dinamis.

Algoritma Divide and Conquer membagi masalah besar menjadi submasalah yang lebih kecil, menyelesaikan setiap submasalah secara terpisah, dan kemudian menggabungkan hasilnya untuk mendapatkan solusi akhir. Pendekatan ini dapat diterapkan dalam berbagai bidang, termasuk dalam pembentukan bentuk halus di aplikasi desain. Sementara itu, program dinamis adalah metode optimasi yang menggunakan tabel untuk menyimpan hasil submasalah yang telah diselesaikan sehingga dapat digunakan kembali tanpa perlu menghitung ulang, sehingga menghemat waktu komputasi.

Makalah ini bertujuan untuk menganalisis dan membandingkan kinerja kedua algoritma tersebut dalam pembentukan bentuk halus di aplikasi desain. Analisis dilakukan dengan mengukur keefektifan dan efisiensi kedua algoritma dalam menghasilkan bentuk halus, serta melihat kelebihan dan kekurangan masing-masing pendekatan



Gambar 1.1 *Contoh fitur border untuk pembentukan smooth shape pada aplikasi figma*

II. TEORI DASAR

A. *Kurva Bezier*

Kurva Bezier 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.

Menurut Haryono [1] kurva Bezier adalah suatu kurva yang halus, kurva Bezier terdapat `n+1` titik untuk mengkonstruksikannya, titik tersebut yaitu titik `P0,P1,..,Pn`. Kurva Bezier dimulai dari titik `P0` yang merupakan titik awal dan kemudian melengkung ke arah titik `Pn` yang merupakan titik akhir dari kurva Bezier. Menurut Kusno [2] Persamaan kurva Bezier yang memiliki derajat disajikan pada persamaan berikut. Perumusan untuk kurva Bézier kuadratik, kubik, dan kuartik seperti berikut :

$$C(t) = \sum_{i=0}^{n} P_i B_i^n(t), 0 \le t \le 1 \quad (1)$$

Dengan:

$$B_i^n(t) = C_i^n (1-t)^{n-i}.t^i \quad (2)$$

$$C_i^n = \frac{n!}{i!\,(n-i)!}$$

P_i = Koefisien geometri atau titik kontrol $C(t)$ [3]

Kurva Bezier dimulai pada titik `P0` dan berakhir di titik `Pn`, tetapi belum tentu melewati titik-titik kontrolnya. Untuk

menggambarkan titik-titik pada kurva Bezier maka dilakukan perhitungan titik dari `P0` sampai dengan `Pn` dengan `t` dari 0 sampai dengan 1.

B. *Divide and Conquer*

Divide artinya membagi persoalan menjadi beberapa upa-persoalan yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil (idealnya setiap upa-persoalan berukuran hamper sama). Sedangkan Conquer artinya menyelesaikan (*solve*) masing-masing upas-persoalan (secara langsung jika sudah berukuran kecil atau secara rekursif jika masih berukuran besar). Sehingga Divide and Conquer merupakan algoritma yang menggabungkan solusi masing-masing upa persoalan sehingga membentuk solusi persoalan semula.[3]

Divide and Conquer secara umum terbagi menjadi 3 fase yakni membagi masalah kedalam sub-sub masalah yang lebih kecil, conquer yakni menyelesaikan sub-sub masalah secara rekursif, dan combine menggabungkan hasil dari penyelesaian sub-sub masalah menjadi penyelesaian yang dikehendaki. Ada 4 hal yang harus diperhatikan dalam strategi Divide and Conquer yaitu *branching factor* (jumlah problem minimal 2), *balance*(dapat dibagi dalam ukuran yang sama), *data dependence of divide function*(saling ketergantungan, rekursif), dan *control parallelism or sequentiality*(berurutan).

C. *Program Dinamis*

Program Dinamis adalah metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpuan tahapan (*stage*) sedemikian sehingga solusi persoalaan dapat dipandang sebagai serangkaian keputusan yang saling berkaitan untuk mencapai solusi yang optimal(Prinsip Optimalitas).

Karakteristik dari persoalan program dinamis antara lain persoalan dapat dibagi menjadi beberapa tahap yang di setiap tahapnya akan diambil satu keputusan. Setiap tahap terdiri dari beberapa status yang berhubungan dengan tahap tersebut.

Terdapat 2 jenis pendekatan Program Dinamis yaitu dinamis maju dan dinamis mundur. Sesuai dengan namanya, program dinamis maju akan bergerak mulai dari tahap 1, 2,3. sampai ke n. Sedangkan program dinamis mundur sebaliknya.

Adapun Langkah-langkah pengembangan algoritma program dinamis yaitu mengkarakteristiskkan struktur Solusi optimal dengan mendefinisikan tahap, variable, Keputusan, dan status. Kemudian mendefinisikan secara rekursif Solusi optimal. Selanjutnya menghitung solusi optimal secara maju atau mundur. Terakhir, merekontruksi solusi optimal.

D. *Divide and Conquer Kurva Bezier dalam Pembentukan Bentuk Halus*

Algoritma Divide and Conquer dapat diterapkan pada kurva Bezier untuk pembentukan bentuk halus. Kurva Bezier akan digunakan di masing-masing titik untuk membentuk border (smooth shape). Proses ini melibatkan membagi masalah menjadi sub-masalah yang lebih kecil (misalnya, segmen-segmen kurva Bezier), menyelesaikan setiap segmen secara rekursif, dan kemudian menggabungkan hasilnya untuk membentuk kurva Bezier yang lengkap.

E. Program Dinamis Kurva Bezier dalam Pembentukan Bentuk Halus

Dalam pembangunan bentuk halus (smooth shape) pada aplikasi desain, akan diterapkan program dinamis maju yang digunakan untuk meminimalkan kesalahan atau ketidaksempuraan bentuk yang dihasilkan

Program Dinamis digunakan untuk menghitung titik-titik pada kurva secara efisien. Program dinamis menyimpan hasil perhitungan titik-titik intermediate dari kurva Bezier dalam tabel, memungkinkan penggunaan kembali hasil ini untuk menghitung titik-titik lainnya tanpa perlu mengulang perhitungan. Hal ini meningkatkan efisiensi komputasi, terutama ketika kurva Bezier memiliki banyak titik kontrol dan titik-titik yang perlu dihitung secara berulang. Metode ini sangat berguna dalam aplikasi desain yang membutuhkan bentuk halus dan presisi tinggi.

III. PEMBAHASAN

Untuk melakukan analisis yang mendalam terkait bagaimana tingkat keefektifan program yang dicapai (pembentukan bentuk halus (*smooth shape*)) dengan menggunakan algoritma divide and conquer dan program dinamis. Algoritma akan dimodifikasi, berikut penjelasanya :

A. *Pembentukan Dua Titik Baru di Setiap Titik Kontrol Asli*

Untuk menghasilkan bentuk halus dalam desain grafis menggunakan kurva Bezier. Langkah pertam kita buat dua titik baru dengan jarak tertentu di titik kontrol asli.Proses ini melibatkan perhitungan dengan jarak antar 2 titik, yaitu titik-titik tetangganya.

Misalkan kita memiliki titik control `Pi`. Kita akan menghasilkan dua titik baru `Pi1` dan `Pi2` yang berada pada jarak tertentu dari `Pi` dengan sudut sesuai dengan arak dari titik kontrol tetangganya.

B. *Menerapkan Fungsi Kurva Bezier ke Kumpulan Titik yang Titik Pusatnya Titik Kontrol*

Langkah selanjutnya adalah menerapkan fungsi kurva Bezier pada kumpulan titik-titik tersebut Kurva Bezier digunakan untuk menghubungkan titik titik disetiap titik kontrol

Kemudian titik-titik yang dihasilkan dari pensubtitusian ke fungsi Bezier akan disimpan dalam array. Array ini siap untuk digunakan untuk rendering atu pemrosesan lebih lanjut dalam aplikasi desain.

C. *Implementasi Program Divide and Conquer*

Langkah-langkah pengembangan algoritma Divide and Conquer yang digunakan dalam pranala kode dapat dijelaskan sebagai berikut:

- Divide : Setiap segmen garis antara dua titik kontrol dibagi menjadi dua bagian dengan menambahkan titik tengah. Proses ini dilakukan untuk setiap segmen.

- Conquer: Metode Bezier diterapkan pada setiap segmen yang telah dibagi untuk meratakan kurva. Ini dilakukan dengan memilih titik kontrol tambahan di sekitar setiap titik kontrol asli, dan kemudian menghitung kurva Bezier baru menggunakan titik kontrol tambahan ini.
- Combine : Hasil dari proses meratakan kurva Bezier untuk setiap segmen digabungkan menjadi satu set titik-titik yang merepresentasikan kurva(shape) yang lebih halus.

Kode modifikasi untuk algoritma Divide and Conquer

```
function bezier_dnc(input control_points : int, input all_points : array, input iterations : int):
Deklarasi
    mid1                        : integer
    mid2                        : integer
    points_to_curve             : array
    new_control_points          : array
Algoritma
       if iterations == 0 then
           return all_points
       else
           for point in control_points do
               mid1.append(midpoint(control_points[i-1], control_points[i]))
           all_points.append(mid1)
           for point in titiktengah1 do
               mid2.append(midpoint(mid1[i-1], mid1[i]))
           if iterasi terakhir then

point_to_curve.append(control_points[0])
               point_to_curve curve.extend(mid2)
               point_to_curve curve.append(control_points[-1])
               all_points.append(curve)
               return all_points

new_control_points.append(control_points[0])

new_control_points.extend(insert_midpoints(mid1, mid2))
           new_control_points.append(control_points[-1]
        return bezier(new_control_points, all_points, iterations - 1)
```

Tambahan : fungsi midpoint untuk menghitung titik tengah atara 2 titik dan fungsi insert_midpoint untuk menyisipkan titik tengah dari dua list.

D. *Implementasi Program Dinamis*

Langkah-langkah pengembangan algoritma Program Dinamis yang digunakan dalam pranala kode dapat dijelaskan sebagai berikut:

- Karakteristikkan Struktur Solusi Optimal:
  Tahap: Setiap tahap adalah perhitungan posisi titik pada kurva Bezier untuk nilai parameter t yang berbeda.
  Variabel Keputusan: Keputusan diambil berdasarkan nilai parameter t yang bervariasi dari 0 hingga 1.
  Status (State): Status ditentukan oleh indeks titik kontrol saat ini i jumlah total titik kontrol n, dan nilai parameter t.
- Definisikan Secara Rekursif Nilai Solusi Optimal:
  Nilai optimal suatu tahap ditentukan secara rekursif dengan memanfaatkan titik kontrol sebelumnya dan berikutnya untuk menghitung koordinat titik pada kurva Bezier. Fungsi compute_BC_xy mendefinisikan hubungan ini, di mana hasil perhitungan titik (x,y) untuk suatu tahap disimpan dalam tabel `mem_x` dan `mem_y`.
- Hitung Nilai Solusi Optimal Secara Maju atau Mundur:
  Perhitungan dilakukan secara maju untuk setiap nilai t dalam interval 0 hingga 1. Tabel mem_x dan mem_y digunakan untuk menyimpan hasil perhitungan sementara, sehingga menghindari perhitungan ulang yang tidak perlu, membuat proses lebih efisien.
- Rekonstruksi Solusi Optimal (Opsional):
  Dalam konteks ini, rekonstruksi solusi tidak diperlukan secara eksplisit karena hasil titik-titik pada kurva Bezier langsung dihitung dan disimpan dalam output_table yang merupakan hasil akhir dari algoritma.

Kode modifikasi untuk Algoritma Program Dinamis :

```
function bezier_dp(input control_points : integer, input n_sampled_points : integer ):
Deklarasi
      n                             : integer
      output_table                  : array
      mem_x                         : array of array
      mem_y                         : array of array
Algoritma
       n = len(control_points)
       t_interval = 1.0 / (n_sampled_points - 1)
       for point in n_sampled_points :
           t = i * t_interval
           mem_x = [[None for _ in range(n+1)] for _ in range(n)]
           mem_y = [[None for _ in range(n+1)] for _ in range(n)]
           output_table.append(compute_BC_xy(0,    n, t))
      return output_table
```

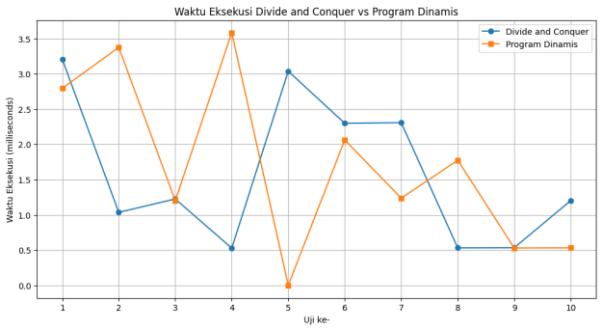
Tambahan : Fungsi `compute_BC_xy` bertanggung jawab untuk menghitung koordinat x dan y dari titik pada kurva Bezier pada nilai parameter t tertentu. Algoritma ini menggunakan metode pemrograman dinamis untuk menghindari perhitungan berulang dari nilai yang sama, yang membuatnya lebih efisien.

E. *Analisis Waktu Eksekusi*

Iterasi : 4

Uji-ke	Waktu Eksekusi (miliseconds)	
	Divide and Conquer	***Program Dinamis***
1.	3.20673	2.79474
2.	1.03521	3.37505
3.	1.22547	1. 19996
4.	0.52714	3.58343
5.	3.03674	0.00000
6.	2.29859	2.06161
7.	2.30789	1.23644
8.	0.53120	1.77479
9.	0.53263	0.52953
10.	1.20234	0.53144

Tabel 3.1 Hasil Uji Program



Gambar 3.1Grafik Uji Eksekusi Program

Dari 10 hasil pengujian, terlihat pada grafik diatas program dinamis memberikan waktu eksekusi yang dominan lebih cepat daripada divide and conquer. Hal ini disebabkan karena algoritma program dinamis menggunakan **memoization** (penyimpanan hasil perhitungan submasalah) yang menghindari penghitungan ulang submasalah yang sama. Fungsi `compute_BC_xy` menggunakan tabel `mem_x` dan `mem_y` untuk menyimpan hasil perhitungan, sehingga setiap submasalah hanya dihitung sekali. Dengan melakukan hal ini program dinamis juga memperhatikan keefisienan waktu. Sedangkan divide and conquer akan melakukan perhitungan berkali-kali untuk masalah yang sama sebelumnya karena tidak menyimpanya di tabel.

F. *Analisis Efektifitas Memori*

Untuk menganalisis keefektifan memori dari dua algoritma yang Anda berikan, yaitu program dinamis (DP) dan divide and conquer (DNC), kita akan melihat bagaimana masing-masing algoritma menggunakan memori dalam proses perhitungannya. Analisis Keefektifan Memori Berdasarkan Program yang diberikan untuk menganalisis keefektifan memori dari dua algoritma yang dibuat, yaitu program dinamis (DP) dan divide and conquer (DNC), kita akan melihat bagaimana masing-masing algoritma menggunakan memori dalam proses perhitungannya.

1. Program Dinamis (DP)
   a. Implementasi Memoization:
   Struktur Penyimpanan Program dinamis menggunakan tabel `mem_x` dan `mem_y` untuk menyimpan hasil perhitungan submasalah pada setiap titik t.
   b. Ukuran Tabel
   Tabel `mem_x` dan `mem_y` masing-masing memiliki ukuran `(n+1) x n`, di mana `n` adalah jumlah titik kontrol. Ukuran ini dapat meningkatkan penggunaan memori secara signifikan jika `n` besar.
   c. Penggunaan Memori
   - Efisiensi Penyimpanan
   Dengan menggunakan memoization, program dinamis menyimpan hasil perhitungan submasalah yang telah dilakukan, sehingga menghindari penghitungan ulang. Ini mengurangi jumlah operasi yang diperlukan dan, secara tidak langsung, mengurangi penggunaan memori untuk operasi berulang.
   - Trade-off
   Meskipun menghemat waktu, penggunaan tabel untuk menyimpan hasil sementara ini mengonsumsi lebih banyak memori karena perlu menyimpan setiap hasil perhitungan submasalah.

2. Divide and Conquer (DNC)

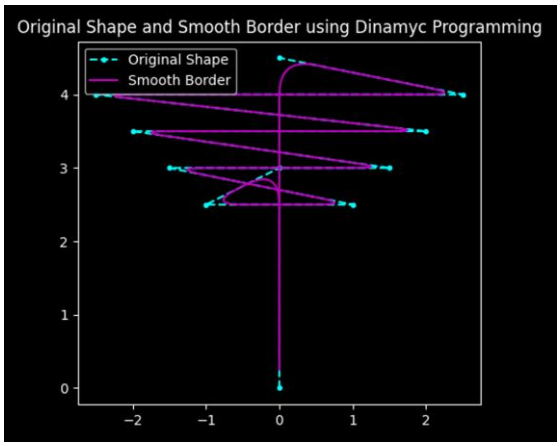
   a. Rekursi dan Penghitungan Ulang
   Struktur Penyimpanan: Algoritma DNC tidak menyimpan hasil perhitungan submasalah. Sebaliknya, ia menghitung ulang setiap submasalah ketika diperlukan.
   b. Penggunaan Memori
   Setiap kali fungsi rekursif dipanggil, stack memori digunakan untuk menyimpan status fungsi saat ini dan parameter yang sedang diproses. Ini dapat meningkatkan penggunaan memori jika kedalaman rekursi tinggi.
   c. Tidak Ada Penyimpanan Sementara
   Tidak ada tabel atau struktur data yang digunakan untuk menyimpan hasil perhitungan sementara, sehingga konsumsi memori untuk penyimpanan lebih rendah dibandingkan dengan program dinamis.

Sehinga dari analisis diatas, bisa disimpulkan bahwa program dinamis memiliki keuntungan yaitu mengurangi waktu eksekusi dengan menghindari penghitungan ulang submasalah melalui memorization. Sedangkan kerugianya

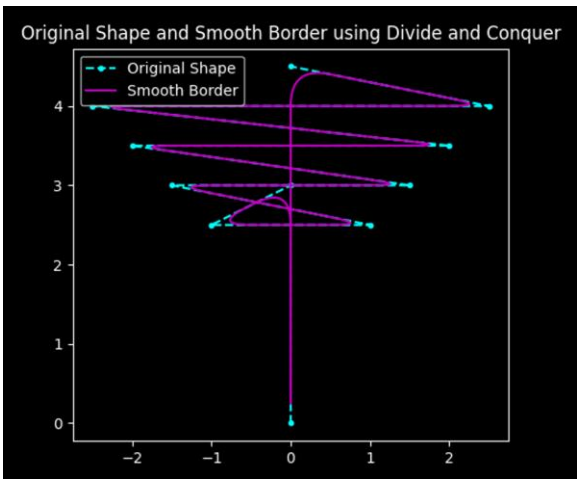
adalah boros memori karena menyimpan hasil perhitungan sementara dalam tabel `mem_x` dan `mem_y`.

Sedangkan keuntungan menggunakan program divide and conquer adalah lebih sedikit memori untuk penyimpanan sementara karena tidak ada memoization. Kerugianya pada waktu eksekusi yang memakan waktu lebih lama karena setiap submasalah dihitung ulang setiap kali diperlukan, dan penggunaan memori stack meningkat dengan kedalaman rekursi.

G. *Analisis Hasil Kurva*



Gambar 3.2 Smooth Shape dengan Algoritma Program Dinamis



Gambar 3.3 Smooth Shape dengan Algoritma Divide and Conquer

Hasil Pengujian hasil visualisasi identic. Selengkapnya dapat dilihat pada link https://bit.ly/HasilUjiStima . Beberapa alasan mengapa kedua algoritma secara visual identik antara lain :

1. Prinsip Dasar Kurva Bezier

   Kurva Bezier didefinisikan oleh titik-titik kontrol dan

   rumus interpolasi polinomial yang menghubungkan titik-titik tersebut. Terlepas dari metode yang digunakan (DP atau DNC), rumus matematika yang mendasarinya tetap sama(tertera pada teori dasar).

2. Interpolasi yang Sama

   Baik DP maupun DNC melakukan interpolasi yang sama untuk setiap nilai t dalam interval [0,1]. Mereka hanya berbeda dalam cara menghitung koordinat titik-titik pada kurva, tetapi interpolasi polinomial yang dilakukan tetap identik

3. PemrosesanTtitk-titik Kontrol yang Sama

   Kedua algoritma menggunakan titik-titik kontrol yang sama sebagai input. Metode apapun yang digunakan untuk menghitung kurva, titik-titik kontrol awal dan aturan interpolasi tidak berubah, yang menyebabkan hasil kurva yang dihasilkan tetap sama.

Oleh karena itu, hasil kurva dari kedua algoritma secara visual identik karena keduanya didasarkan pada prinsip interpolasi kurva Bezier yang sama. Meskipun pendekatan mereka berbeda, hasil akhirnya tetap sama.

IV. KESIMPULAN

Melalui analisis yang mendalam mengenai penggunaan algoritma divide and conquer dan program dinamis membawa penulis mencapai kesimpulan bahwa program dinamis lebih efisien dalam hal waktu eksekusi tetapi membutuhkan lebih banyak memori untuk penyimpanan hasil sementara, sementara divide and conquer mengonsumsi lebih sedikit memori tetapi kurang efisien dalam hal waktu eksekusi. Kedua algoritma menghasilkan kurva yang secara visual identik, memberikan panduan bagi pengembang aplikasi desain dalam memilih algoritma yang sesuai berdasarkan kebutuhan spesifik dan karakteristik desain yang diinginkan.

Kesimpulan ini diharapkan dapat menjadi salah satu panduan bagi pengembang aplikasi desain grafis untuk memilih algoritma yang paling sesuai, sehingga dapat meningkatkan pengalaman pengguna dan kualitas hasil desain secara keseluruhan.

GITHUB

Implementasi kode dapat dilihat pada link berikut :
https://github.com/mlatia/Analisis-Pembentukan-Smooth-Shape-pada-Aplikasi-Desain

UCAPAN TERIMAKASIH

Dengan kerendahan hati penulis, penulis ingin menyampaikan ucapan terima kasih kepada Allah dan semua pihak yang berkontribusi dalam penyelesaian makalah ini. Terima kasih kepada Bapak Rinaldi Munir, selaku dosen pengajar mata kuliah IF2211 Strategi Algoritma K01 yang telah membimbing penulis dan teman-teman selama perkuliahan Strategi Algoritma Tahun Ajaran 2023/2024. Keberhasilan makalah ini juga tidak terlepas dari kerja sama dan bantuan dari semua pihak. Terakhir, terima kasih kepada pembaca yang telah meluangkan waktunya untuk membaca

makalah ini. Semoga makalah ini bisa memberikan manfaat dan kontribusi positif dalam bidang penerapan konsep Strategi Algoritma.

REFERENSI

[1] Haryono, Agung. (2014). Studi Pembentukan Huruf Font Dengan Kurva Bezier. Diakses pada tanggal 11 Juni 2024, dari https://www.researchgate.net/publication/332329682_Studi_Pembentukan_Huruf_Font_Dengan_Kurva_Bezier

[2] Andi611. (2024). Dynamic-Programming-Algorithm. GitHub. Diakses pada 11 Juni 2024, dari https://github.com/andi611/Dynamic-Programming-Algorithm

[3] Munir, Rinaldi. (2023). Program-Dinamis-2020-Bagian1. Diakses pada tanggal 11 Juni 2024, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian1.pdf

[4] Munir, Rinaldi. (2023). Algoritma Divide and Conquer Bagian 1. Diakses pada tanggal 11 Juni 2024, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-(2024)-Bagian1.pdf

[5] Munir, Rinaldi. (2023). Program-Dinamis-2020-Bagian2. Diakses pada tanggal 11 Juni 2024, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian1.pdf

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, 12 Juni 2024

Melati Anggraini-13522035