**LAPORAN TUGAS KECIL 2**

**IF2211 STRATEGI ALGORITMA**

Membangun Kurva Bézier dengan Algoritma Titik Tengah berbasis *Divide and Conquer*



Disusun oleh:

Agil Fadillah Sabri (13522006)

**Program Studi Teknik Informatika**

**Sekolah Teknik Elektro dan Informatika**

**Institut Teknologi Bandung**

**2024**

**DAFTAR ISI**

DAFTAR ISI i

DAFTAR GAMBAR ii

DAFTAR TABEL iii

BAB I DESKRIPSI MASALAH 1

BAB II LANGKAH PEMECAHAN MASALAH

1. Analisis dan Implementasi dalam Algoritma *Brute Force*
2. Analisis dan Implementasi dalam Algoritma *Divide and Conquer*

BAB III IMPLEMENTASI PROGRAM

1. Implementasi Algoritma Brute Force
2. Implementasi Algoritma Divide and Conquer

BAB IV UJI COBA

* 1. Tampilan Awal Program
  2. Input Melalui File Berekstensi txt
  3. Input Melalui CLI (Terminal)

BAB V ANALISIS SOLUSI

BAB VI IMPLEMENTASI BONUS

1. Visualisasi Pembangkitan Kurva
2. Generalisasi Algoritma

DAFTAR PUSTAKA

LAMPIRAN

**DAFTAR GAMBAR**

Gambar 1. Solusi Optimal untuk Matriks berdasarkan Sekuens yang Diberikan 2

Gambar 2. Implementasi Program (1) 5

Gambar 3. Implementasi Program (2) 6

Gambar 4. Implementasi Program (3) 7

Gambar 5. Implementasi Program (4) 8

Gambar 6. Implementasi Program (5) 9

Gambar 7. Implementasi Program (6) 10

Gambar 8. Tampilan Awal Program 11

Gambar 9. Tampilan Input Tidak Valid 11

Gambar 10. Contoh Test 1 (test1.txt) 12

Gambar 11. Hasil Luaran Test 1 12

Gambar 12. Hasil Luaran Test 1 (*txt file*) 12

Gambar 13. Contoh Test 2 (test2.txt) 13

Gambar 14. Hasil Luaran Test 2 13

Gambar 15. Hasil Luaran Test 2 (*txt file*) 13

Gambar 16. Contoh Test 3 (test3.txt) 14

Gambar 17. Hasil Luaran Test 3 14

Gambar 18. Hasil Luaran Test 3 (*txt file*) 14

Gambar 19. Contoh Test 4 (test4.txt) 15

Gambar 20. Hasil Luaran Test 4 15

Gambar 21. Hasil Luaran Test 4 (*txt file*) 15

Gambar 22. Contoh Test 5 (test5.txt) 16

Gambar 23. Hasil Luaran Test 5 16

Gambar 24. Hasil Luaran Test 5 (*txt file*) 16

Gambar 25. Contoh Test 6 (Terminal) 17

Gambar 26. Hasil Luaran Test 6 17

Gambar 27. Hasil Luaran Test 6 (*txt file*) 17

Gambar 28. Contoh Test 7 (Terminal) 18

Gambar 29.Hasil Luaran Test 7 18

Gambar 30. Hasil Luaran Test 7 (*txt file*) 18

Gambar 31. Contoh Test 8 (Terminal) 19

Gambar 32.Hasil Luaran Test 8 19

Gambar 33. Hasil Luaran Test 8 (*txt file*) 19

Gambar 34. Contoh Test 9 (Terminal) 20

Gambar 35.Hasil Luaran Test 9 20

Gambar 36. Hasil Luaran Test 9 (*txt file*) 20

Gambar 37. Contoh Test 10 (Terminal) 21

Gambar 38.Hasil Luaran Test 10 21

Gambar 39. Hasil Luaran Test 10 (*txt file*) 21

ii

**DAFTAR TABEL**

Tabel 1. Contoh Matriks pada *Cyberpunk 2077 Breach Protocol*  1

Tabel 2. Contoh Masukan 1 dari File Berekstensi txt 12

Tabel 3. Contoh Masukan 2 dari File Berekstensi txt 13

Tabel 4. Contoh Masukan 3 dari File Berekstensi txt 14

Tabel 5. Contoh Masukan 4 dari File Berekstensi txt 15

Tabel 6. Contoh Masukan 5 dari File Berekstensi txt 16

Tabel 7. Contoh Masukan 1 dari Terminal 17

Tabel 8. Contoh Masukan 2 dari Terminal 18

Tabel 9. Contoh Masukan 3 dari Terminal 19

Tabel 10. Contoh Masukan 4 dari Terminal 20

Tabel 11.Contoh Masukan 5 dari Terminal 21

iii

**BAB I**

**DESKRIPSI MASALAH**

*Cyberpunk 2077 Breach Protocol* adalah minigame meretas pada permainan video *Cyberpunk 2077*. *Minigame* ini merupakan simulasi peretasan jaringan lokal dari ICE (*Intrusion Countermeasures Electronics*) pada permainan *Cyberpunk 2077*. Komponen pada permainan ini antara lain adalah:

1. Token – terdiri dari dua karakter alfanumerik seperti E9, BD, dan 55.
2. Matriks – terdiri atas token-token yang akan dipilih untuk menyusun urutan kode.
3. Sekuens – sebuah rangkaian token (dua atau lebih) yang harus dicocokkan.
4. Buffer – jumlah maksimal token yang dapat disusun secara sekuensial.

Aturan permainan Breach Protocol antara lain:

1. Pemain bergerak dengan pola horizontal, vertikal, horizontal, vertikal (bergantian) hingga semua sekuens berhasil dicocokkan atau buffer penuh.
2. Pemain memulai dengan memilih satu token pada posisi baris paling atas dari matriks.
3. Sekuens dicocokkan pada token-token yang berada di buffer.
4. Satu token pada buffer dapat digunakan pada lebih dari satu sekuens.
5. Setiap sekuens memiliki bobot hadiah atau reward yang variatif.
6. Sekuens memiliki panjang minimal berupa dua token.

Berikut adalah ilustrasi kasus. Misal terdapat matriks sebagai berikut:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 7A | 55 | E9 | E9 | 1C | 55 |
| 55 | 7A | 1C | 7A | E9 | 55 |
| 55 | 1C | 1C | 55 | E9 | BD |
| BD | 1C | 7A | 1C | 55 | BD |
| BD | 55 | BD | 7A | 1C | 1C |
| 1C | 55 | 55 | 7A | 55 | 7A |

**Tabel 1.** Contoh Matriks pada *Cyberpunk 2077* *Breach Protocol*

Jika panjang buffernya adalah tujuh, dan terdapat tiga sekuens sebagai berikut:

1. BD E9 1C dengan hadiah berbobot 15.
2. BD 7A BD dengan hadiah berbobot 20.
3. BD 1C BD 55 dengan hadiah berbobot 30.

Maka solusi yang optimal untuk matriks dan sekuens yang diberikan adalah sebagai berikut:

1. Total bobot hadiah : 50 poin.
2. Total langkah : 6 langkah.



**Gambar 1.** Solusi Optimal untuk Matriks berdasarkan Sekuens yang Diberikan

**BAB II**

**LANGKAH PEMECAHAN MASALAH**

Metode algoritma yang digunakan untuk menyelesaikan permainan *Cyberpunk 2077 Breach Protocol* ini adalah metode *Brute Force*. Adapun langkah-langkah penyelesaiannya adalah sebagai berikut:

1. Pertama-tama, program menerima masukan dari pengguna, bisa melalui file berekstensi txt ataupun melalui *Command Line Interface* (CLI) pada terminal. Dari sini akan diperoleh ukuran *buffer*, isi matriks, sekuens beserta *reward* masing-masing sekuens. Sebelum dilakukan pencarian, terlebih dahulu dihitung *reward* maksimum yang mungkin bisa diperoleh dari sekuens yang ada (dengan menjumlahkan semua *reward* sekuens yang bernilai positif). Jika semua sekuens memiliki reward negatif atau 0, maka reward maksimum yang bisa diperoleh dapat dianggap sama dengan 0 (karena 0 ≥ semua bilangan negatif). Hal ini berguna nantinya untuk meningkatkan efisiensi pencarian sekuens.
2. Karena permainan selalu dimulai dengan memilih salah satu token yang ada pada baris pertama matriks, maka pada *main* *program*, proses pencarian diawali dengan melakukan perulangan (*looping*) sebanyak jumlah kolom untuk mengiterasi satu-persatu kemungkinan yang ada pada setiap kolom pada baris pertama. Pada tahap ini, *buffer* pada posisi pertama telah terisi.
3. Selanjutnya, dalam setiap perulangan pada program utama tersebut, akan dilakukan pengisian *buffer* pada posisi kedua, dengan melakukan pergerakan secara vertikal. Seperti langkah sebelumnya, akan dilakukan perulangan untuk mengiterasi setiap kemungkinan token yang ada pada kolom yang sama dengan token pada *buffer* di posisi pertama (karena pergerakan vertikal). Pada tahap ini, *buffer* pada posisi kedua telah terisi. Adapun jumlah perulangan dilakukan sebanyak (jumlah baris – 1), karena tidak mengikutkan token yang sudah mengisi *buffer* pada posisi tepat sebelumnya dari posisi saat ini. Selain itu, untuk sel matriks yang telah diambil, tidak bisa lagi digunakan, sehingga ketika bertemu dengan sel matriks yang telah diambil, proses perulangan langsung berlanjut untuk sel berikutnya. Hal ini dilakukan dengan memberikan tanda pada setiap sel matriks, dengan ‘0’ belum diambil dan ‘1’ telah diambil.
4. Selanjutnya, untuk setiap perulangan pada langkah ketiga, akan kembali dilakukan perulangan untuk mengisi buffer pada posisi ketiga, dengan pergerakan secara horisontal. Perulangan ini dilakukan untuk mengiterasi setiap kemungkinan token yang ada pada baris yang sama dengan token pada posisi kedua. jumlah perulangan dilakukan sebanyak (jumlah kolom – 1), karena tidak mengikutkan token yang sudah mengisi *buffer* pada posisi tepat sebelumnya dari posisi saat ini. Sama seperti sebelumnya, sel matriks yang telah diambil, tidak bisa dipakai kembali pada *buffer* yang sama.
5. Langkah 3 dan 4 selanjutnya diulangi secara terus-menerus secara bergantian hingga *buffer* telah terisi penuh (telah mencapai posisi terakhir) dengan menggunakan pendekatan secara rekursif. Terdapat dua fungsi yang digunakan, yaitu fungsi untuk melakukan iterasi secara vertikal (*function* vertical) dan fungsi untuk melakukan iterasi secara horizontal (*function* horizontal). Basis dari kedua fungsi rekursif ini adalah ketika *buffer* telah terisi penuh, dimana akan langsung dilakukan proses perhitungan *reward* yang didapatkan dan keluar dari proses rekursif. Jika seandainya *reward* saat ini lebih tinggi dari *reward* sebelumnya, maka *reward* tertinggi akan ditukar dan isi *buffer*-nya akan disimpan. Adapun proses rekursifnya dilakukan dengan cara sebagai berikut:
6. Pada setiap iterasi pada program utama (langkah 2), dilakukan pemanggilan fungsi vertikal (*function* vertical).
7. Selanjutnya, di dalam fungsi vertikal, dilakukan pengecekan apakah kondisi *buffer* telah memenuhi basis atau belum. Jika belum, maka akan dilakukan perulangan kembali secara vertikal (langkah 3). Pada setiap perulangan ini akan dilakukan pemanggilan fungsi horizontal (*function* horizontal).
8. Selanjutnya, di dalam fungsi horizontal, juga dilakukan pengecakan apakah kondisi *buffer* telah memenuhi basis atau belum. Jika belum, maka akan dilakukan perulangan kembali secara horizontal (langkah 4). Pada setiap perulangan ini akan dilakukan kembali pemanggilan fungsi vertikal (*function* vertical) dan proses diulangi kembali.
9. Proses ini terus berlanjut hingga *buffer* telah terisi penuh.
10. Untuk meningkatkan efisiensi program, pada fungsi rekursif ditambahkan satu basis tambahan, dimana jika isi *buffer* saat ini telah mencapai *reward* maksimum yang mungkin bisa didapatkan (hasil paling optimal), maka akan segera keluar dari proses rekursif.
11. Setelah proses pencarian selesai, program kemudian menampilkan pada terminal *reward* maksimum yang bisa diperoleh dari matriks yang ada, urutan isi token pada *buffer* yang diperoleh, posisi masing-masing token pada *buffer* di dalam matriks (dengan format: kolom, baris), waktu yang diperlukan untuk melakukan proses pencarian (dalam milisecond/ms) dan *prompt* atau pertanyaan apakah hasil ini ingin disimpan dalam file berekstensi txt (dengan format yang sama). Jika sebelumnya masukan via CLI, maka juga akan ditampilkan matriks yang digunakan, sekuens serta reward masing-masing sekuens.

**BAB III**

**IMPLEMENTASI PROGRAM**

Program penyelesaian permainan *Cyberpunk 2077 Breach Protocol* ini diimplementasikan menggunakan bahasa pemrograman python (.py). Adapaun kode programnya adalah sebagai berikut:

A screenshot of a computer program

Description automatically generated

**Gambar 2.** Implementasi Program (1)

A screen shot of a computer program

Description automatically generated

**Gambar 3.** Implementasi Program (2)

A screenshot of a computer program

Description automatically generated

**Gambar 4.** Implementasi Program (3)

A screen shot of a computer program

Description automatically generated

**Gambar 5.** Implementasi Program (4)

A screen shot of a computer program

Description automatically generated

**Gambar 6.** Implementasi Program (5)

A screenshot of a computer program

Description automatically generated

**Gambar 7.** Implementasi Program (6)

**BAB IV**

**UJI COBA**

1. **Tampilan Awal Program**

A screenshot of a computer menu

Description automatically generated

**Gambar 8.** Tampilan Awal Program

A screenshot of a computer

Description automatically generated

**Gambar 9.** Tampilan Input Tidak Valid

1. **Input Melalui File Berekstensi txt**
   1. **Contoh 1**

|  |  |
| --- | --- |
| **Input** | **Output** |
| A screenshot of a computer  Description automatically generated  **Gambar 10.**  Contoh Test 1 (test1.txt) | A screenshot of a computer  Description automatically generated  **Gambar 11.** Hasil Luaran Test 1 |
| A screenshot of a computer  Description automatically generated  **Gambar 12.** Hasil Luaran Test 1 (*txt file*) |
| **Keteragan:** Masukan dari file test1.txt. *Reward* maksimal yang diperoleh adalah 50 dengan sekuens 1) BD 7A BD, 20 poin; 2) BD IC BD 55, 30 poin. | |

**Tabel 2.** Contoh Masukan 1 dari File Berekstensi txt

* 1. **Contoh 2**

|  |  |
| --- | --- |
| **Input** | **Output** |
| A screenshot of a computer  Description automatically generated  **Gambar 10.**  Contoh Test 2 (test2.txt) | A computer screen shot of a computer program  Description automatically generated  **Gambar 11.** Hasil Luaran Test 2 |
| A screenshot of a computer  Description automatically generated  **Gambar 12.** Hasil Luaran Test 2 (*txt file*) |
| **Keteragan:** Masukan dari file test2.txt. *Reward* maksimal yang diperoleh adalah 0. Hal ini terjadi karena seluruh sekuens memiliki panjang yang melebihi ukuran *buffer*. | |

**Tabel 3.** Contoh Masukan 2 dari File Berekstensi txt

* 1. **Contoh 3**

|  |  |
| --- | --- |
| **Input** | **Output** |
| A screenshot of a computer  Description automatically generated  **Gambar 13.**  Contoh Test 3 (test3.txt) | A screenshot of a computer  Description automatically generated  **Gambar 14.** Hasil Luaran Test 3 |
| A screenshot of a computer  Description automatically generated  **Gambar 15.** Hasil Luaran Test 3 (*txt file*) |
| **Keteragan:** Masukan dari file test3.txt. *Reward* maksimal yang diperoleh adalah 55 dengan sekuens 1) AH 47 BR 5A AH 8L, 40 poin; 2) LL FA, 15 poin. | |

**Tabel 4.** Contoh Masukan 3 dari File Berekstensi txt

* 1. **Contoh 4**

|  |  |
| --- | --- |
| **Input** | **Output** |
| A screenshot of a computer  Description automatically generated  **Gambar 16.**  Contoh Test 4 (test4.txt) | A screenshot of a computer  Description automatically generated  **Gambar 17.** Hasil Luaran Test 4 |
| A screenshot of a computer  Description automatically generated  **Gambar 18.** Hasil Luaran Test 4 (*txt file*) |
| **Keteragan:** Masukan dari file test4.txt. *Reward* maksimal yang diperoleh adalah 50 dengan sekuens 1) YT IT HM 22, 50 poin. | |

**Tabel 5.** Contoh Masukan 4 dari File Berekstensi txt

* 1. **Contoh 5**

|  |  |
| --- | --- |
| **Input** | **Output** |
| A screenshot of a computer  Description automatically generated **Gambar 19.**  Contoh Test 5 (test5.txt) | A computer screen shot of a program  Description automatically generated  **Gambar 20.** Hasil Luaran Test 5 |
| A screenshot of a computer program  Description automatically generated  **Gambar 21.** Hasil Luaran Test 5 (*txt file*) |
| **Keteragan:** Masukan dari file test5.txt. *Reward* maksimal yang diperoleh adalah 0. Hal ini terjadi karena seluruh sekuens memiliki *reward* negatif, sehingga 0 menjadi *reward* yang paling optimal. | |

**Tabel 6.** Contoh Masukan 5 dari File Berekstensi txt

1. **Input Melalui CLI (Terminal)**

Pada masukan melalui CLI, matriks, sekuens, serta *reward* masing-masing sekuens diperoleh secara *random* (acak). Adapun rentang *reward* yang digunakan adalah [-50,50] dengan selang 5 (-50, -45, -40, …, 0, …, 40, 45, 50).

* 1. **Contoh 1**

|  |  |
| --- | --- |
| **Input** | **Output** |
| A screen shot of a computer  Description automatically generated  **Gambar 22.**  Contoh Test 6 (terminal) | A screenshot of a computer program  Description automatically generated  **Gambar 23.** Hasil Luaran Test 6 |
| A screenshot of a computer  Description automatically generated  **Gambar 24.** Hasil Luaran Test 6 (*txt file*) |
| **Keteragan:** Masukan dari terminal. *Reward* maksimal yang diperoleh adalah 45 dengan sekuens 1) 7A 7A E9 E9, 30 poin; 2) E9 BD 1C, 15 poin. | |

**Tabel 7.** Contoh Masukan 1 dari Terminal

* 1. **Contoh 2**

|  |  |
| --- | --- |
| **Input** | **Output** |
| A screen shot of a computer  Description automatically generated **Gambar 25.**  Contoh Test 7 (terminal) | A screenshot of a computer program  Description automatically generated**Gambar 26.** Hasil Luaran Test 7 |
| A screenshot of a computer  Description automatically generated  **Gambar 27.** Hasil Luaran Test 7 (*txt file*) |
| **Keteragan:** Masukan dari terminal. *Reward* maksimal yang diperoleh adalah 95 dengan sekuens 1) 1T 1F IL 22, 50 poin; 2) HM B0, 45 poin. | |

**Tabel 8.** Contoh Masukan 2 dari Terminal

* 1. **Contoh 3**

|  |  |
| --- | --- |
| **Input** | **Output** |
| A screen shot of a computer program  Description automatically generated  **Gambar 28.**  Contoh Test 8 (terminal) | A screenshot of a computer  Description automatically generated  **Gambar 29.** Hasil Luaran Test 8 |
| A screenshot of a computer  Description automatically generated  **Gambar 30.** Hasil Luaran Test 8 (*txt file*) |
| **Keteragan:** Masukan dari terminal. *Reward* maksimal yang diperoleh adalah 35 dengan sekuens 1) L1 L1 L1 GA, 35. | |

**Tabel 9.** Contoh Masukan 3 dari Terminal

* 1. **Contoh 4**

|  |  |
| --- | --- |
| **Input** | **Output** |
| **Gambar 31.**  Contoh Test 9 (terminal) | A screenshot of a computer program  Description automatically generated  **Gambar 32.** Hasil Luaran Test 9 |
| A screenshot of a computer  Description automatically generated  **Gambar 33.** Hasil Luaran Test 9 (*txt file*) |
| **Keteragan:** Masukan dari terminal. *Reward* maksimal yang diperoleh adalah 0. Hal ini terjadi karena seluruh sekuens memiliki panjang yang melebihi ukuran *buffer*. | |

**Tabel 10.** Contoh Masukan 4 dari Terminal

* 1. **Contoh 5**

|  |  |
| --- | --- |
| **Input** | **Output** |
| **Gambar 34.**  Contoh Test 10 (terminal) | A screenshot of a computer program  Description automatically generated  **Gambar 35.** Hasil Luaran Test 10 |
| A screenshot of a computer  Description automatically generated  **Gambar 36.** Hasil Luaran Test 10 (*txt file*) |
| **Keteragan:** Masukan dari terminal. *Reward* maksimal yang diperoleh adalah 45 dengan sekuens 1) K2 25 BB, 45 poin. | |

**Tabel 11.** Contoh Masukan 5 dari Terminal

**DAFTAR PUSTAKA**

<https://cyberpunk-hacker.com/> (Diakses pada 8 Februari 2024).

**LAMPIRAN**

1. **Tampilan Awal Program**

Link : <https://github.com/Agil0975/Tucil1_13522006>

1. **Tabel Checkpoint Program**

|  |  |  |
| --- | --- | --- |
| **Poin** | **Ya** | **Tidak** |
| 1. Program berhasil dikompilasi tanpa kesalahan | ✓ |  |
| 2. Program berhasil dijalankan | ✓ |  |
| 3. Program dapat membaca masukan berkas .txt | ✓ |  |
| 4. Program dapat menghasilkan masukan secara acak | ✓ |  |
| 5. Solusi yang diberikan program optimal | ✓ |  |
| 6. Program dapat menyimpan solusi dalam berkas .txt | ✓ |  |
| 7. Program memiliki GUI |  | ✓ |