LAPORAN TUGAS KECIL 1

IF2211 STRATEGIALGORITMA

Penyelesaian Permainan Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force

Disusun oleh:

Mohammad Nugraha Eka Prawira (13522001)

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2024

BAB I

LATAR BELAKANG

Deskripsi Tugas :

Permainan Breach Protocol Cyberpunk 2077 Breach Protocol adalah minigame meretas pada permainan video Cyberpunk 2077. Minigame ini merupakan simulasi peretasan jaringan local 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. Pemainmemulai 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

SpesifikasiTugasKecil1 :

Buatlah program sederhana dalam Bahasa C/C#/C++/Java/Python/JavaScript/Go/PHP yang mengimplementasikan algoritma BruteForce untuk mencari Solusi paling optimal permainan BreachProtocol.

Algoritma BruteForce

Algoritma BruteForce sendiri adalah suatu pendekatan penyelesaian masalah yang sederhana dan langsung, di mana sistem mencoba semua solusi mungkin secara berurutan hingga menemukan solusi yang benar atau optimal. Pendekatan ini tidak menggunakan strategi atau peningkatan kecerdasan tertentu untuk mempercepat pencarian, tetapi secara sistematis memeriksa semua kemungkinan jawaban.

Pendekatan brute force cenderung mencakup beberapa karakteristik umum:

1. Eksploratif Penuh:

Brute force memeriksa semua solusi yang mungkin, tanpa meninggalkan celah atau melewatkan kemungkinan solusi yang sah. Ini termasuk pengujian setiap kombinasi yang mungkin, memeriksa semua jalur yang dapat diambil, atau mencoba semua nilai yang mungkin.

2. Kesederhanaan dan Keterbacaan:
   Algoritma brute force sering kali relatif sederhana dan mudah dimengerti. Mekanisme pengujian yang langsung membuatnya mudah diimplementasikan.
3. Performa yang Kurang Efisien:
   Meskipun sederhana dan jelas, algoritma brute force dapat kurang efisien dalam hal waktu dan sumber daya. Terutama pada masalah dengan ruang solusi yang besar, karena mencoba setiap kombinasi mungkin memerlukan waktu yang sangat lama.
4. Penggunaan pada Kasus Kecil:
   Brute force dapat efektif pada masalah kecil atau ketika jumlah solusi yang mungkin terbatas.

Kelemahan utama dari algoritma brute force adalah kinerjanya yang kurang efisien pada masalah yang lebih besar. Oleh karena itu, pada beberapa kasus, diperlukan pendekatan algoritmik yang lebih canggih untuk mengatasi masalah dengan waktu eksekusi yang lebih cepat.

Pada persoalan kali ini kita mengimplementasikan algoritma bruteforce dalam penyelesaian Permainan Breach Protocol CyberPunk 2077. Algoritma ini digunakan untuk mencoba mencari Solusi paling optimal satu persatu. Langkah-Langkah yang dilakukan dalam implementasi yang digunakan berupa :

Algoritma solvePuzzle

- Inisialisasi Variabel:
  Inisialisasi variabel seperti indeks, counters, point, dan buffer sementara.
- Inisialisasi Counters dan maxScore:
  Inisialisasi counters dan menghitung maxScore dari seluruh reward sequence.
- Inisialisasi tempBuffer:
  Menetapkan panjang dan ukuran buffer sementara sesuai dengan ukuran buffer input.
- Iterasi dengan Brute Force:
  Melakukan iterasi dengan brute force selama counters dalam batas matriks dan skor final belum mencapai maksimum.
- Penanganan Kasus Base:
  Jika ukuran buffer 0, set finalScore menjadi 0 dan selesaikan fungsi.
- Penanganan Level Teratas:
  Jika berada pada level teratas, tambahkan koordinat pada counters[0] ke tempBuffer dan tingkatkan index.
- Penanganan Level Tengah dan Akhir:
  Navigasi dan rekursi sesuai dengan aturan tertentu untuk level tengah atau akhir.
- Pengecekan dan Pembaruan Final Score:
  Cek skor sementara, perbarui finalScore dan buffer jika ditemukan solusi yang lebih baik.
- Inkrementasi Counters atau Backtracking:

Inkrementasi counters atau lakukan backtracking tergantung pada kondisi yang berlaku.

- Hasil Akhir:
  Setelah proses brute force selesai, perbarui finalScore dan buffer dengan solusi terbaik yang ditemukan.

Fungsi ini secara rekursif mencari solusi dengan mengeksplorasi semua kemungkinan kombinasi koordinat buffer pada matriks, dengan tujuan mencapai skor optimal.

BABII

SOURCE CODE PROGRAM

Program ini dibuat dalam bahasa C dengan menggunakan library:

2 - main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <stdbool.h>
#include <math.h>
#include "algorithm.h"


int main()
{
  clock_t t;
  int bufferSize;
  int matrixWidth, matrixHeight;
  Matrix matrixs;
  int numSequences;
  int totaltoken;
  int buffers;
  int sequenceSize;
  sequence seqnums[numSequences];
  int totalscore =0;
  Buffer buffer;
 char answer;

  printf("Cyberpunk 2077 Breach Protocol\n");
  printf("\n\nPILIH MODE PERMAINAN :\n");
  printf("1. Inputan text.\n");
  printf("2. Inputan Pemain.\n");

  printf("Input: ");
  scanf("%c", &answer);
  while (answer != '1' && answer != '2')
  {
    printf("Hanya ada dua pilihan.\n");
    scanf("Input: %c",&answer);
  }

  if (answer == '1')
  {

    FILE *file = fopen("input.txt", "r");
      if (file == NULL) {
          printf("Gagal membuka berkas\n");
          return 1;
      }
    // Baca buffer size
    fscanf(file, "%d", &bufferSize);
    // Baca jumlah kolom dan baris

    fscanf(file, "%d %d", &matrixWidth, &matrixHeight);
    // Baca matrix
    readMatrix(file, &matrixs, matrixWidth, matrixHeight);
    printf("Matrix :\n");
    displayMatrix(matrixs,matrixHeight,matrixWidth);

// baca sequence

    fscanf(file, "%d", &numSequences);


    for (int i = 0; i < numSequences; ++i) {
        int leng=0;
            readSequences(file, &seqnums[i] ,&leng);
            int reward;
            fscanf(file, "%d", &reward);
            seqnums[i].rewards=reward;
            printf("Sequences %d:\n", i + 1);
            for (int j = 0; j < leng; ++j) {
              printf("%X ",seqnums[i].seqs[j]);
            }
            printf("\nReward: %d\n", reward);
        }
        qsort(seqnums, numSequences, sizeof(sequence), compareRewards);
  buffer.lengths = 0;
  buffer.buffSize = bufferSize;
  t = clock();    // Temukan sekuens optimal
  solvePuzzle(matrixs,seqnums,numSequences,&buffer, &totalscore);

    // Cetak hasil optimal
   printf("\n\nSkor optimal: %d\n", totalscore);
        if (totalscore == 0) {
            printf("Tidak ada solusi\n\n");
        } else {
            printf("Buffer: "); displayBuffMat(buffer,matrixs);printf("\n");
            printf("Lokasi pada matriks:\n");displayBuff(buffer);printf("\n");
        }
    t = clock() - t;
    int totaltime = (int)(((double)(t) / CLOCKS_PER_SEC)*1000);
    printf("%d ms \n", totaltime);
    saveText( buffer,totaltime, totalscore, matrixs);
  }

  else{
    clock_t t;
  int matrixWidth, matrixHeight;
  Matrix matrixs;
  int numSequences;
  int totaltoken;
  int buffers;
  int sequenceSize;

  int totalscore =0;
  Buffer buffer;

    scanf("%d",&totaltoken);
    int token[totaltoken];
    for (int i =0;i<totaltoken;i++)
    {
      scanf("%X",&token[i]);
    }
    scanf("%d",&buffers);
    scanf("%d %d",&matrixWidth,&matrixHeight);
    scanf("%d",&numSequences);
    sequence seqnums[numSequences];
    scanf("%d",&sequenceSize);
    buildMatrix(&matrixs,matrixHeight,matrixWidth,token,totaltoken);
    displayMatrix(matrixs,matrixHeight,matrixWidth);
    for (int i =0;i<numSequences;i++)
    {

      buildSequence(&seqnums[i],sequenceSize,token,totaltoken);
    }

    for (int i=0;i<numSequences;i++)
    {
      printf("Sequences %d:\n", i + 1);
      for (int j =0;j<(seqnums[i].lengs);j++)
      {
        printf("%X ",seqnums[i].seqs[j]);
      }
      printf("\nReward: %d\n",seqnums[i].rewards);
    }
    qsort(seqnums, numSequences, sizeof(sequence), compareRewards);
    buffer.lengths = 0;
    buffer.buffSize = buffers;
    t = clock();    // Temukan sekuens optimal
    solvePuzzle(matrixs,seqnums,numSequences,&buffer, &totalscore);

      // Cetak hasil optimal
    printf("\n\nSkor optimal: %d\n", totalscore);
          if (totalscore == 0) {
              printf("Tidak ada solusi\n\n");
          } else {
              printf("Buffer: "); displayBuffMat(buffer,matrixs);printf("\n");
              printf("Lokasi pada matriks:\n");displayBuff(buffer);printf("\n");
          }
    t = clock()   t;
    int totaltime = (int)(((double)(t) / CLOCKS_PER_SEC)*1000);
    printf("%d ms \n", totaltime);
    saveText( buffer,totaltime, totalscore, matrixs);
  }



    return 0;
}
```

2 - algorithm.h

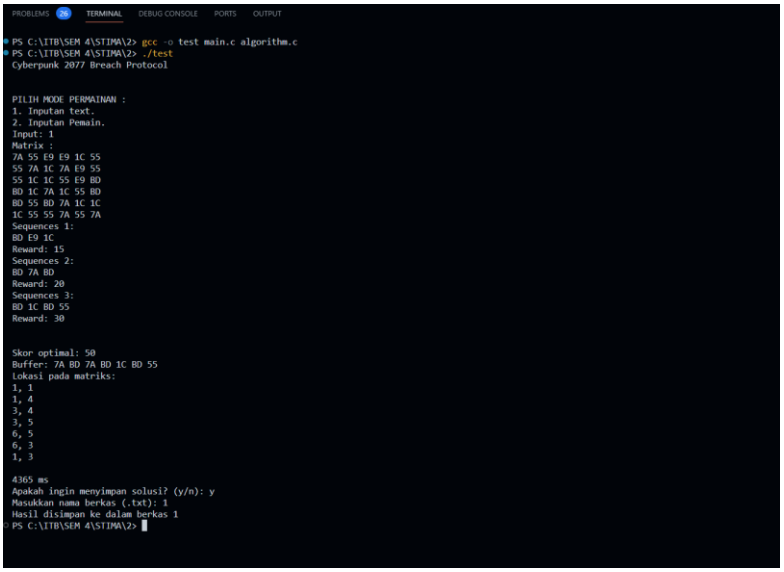
```h
#ifndef PUZZLE_SOLVER_H
#define PUZZLE_SOLVER_H

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <stdbool.h>
#include <math.h>

#define MAXS 100
#define ELMT(M, i, j) (M).mem[(i)][(j)]

typedef struct
{
    int seqs[MAXS];
    int lengs;
    int rewards;

} sequence;

typedef struct
{
    int x;
    int y;
} Coordinate;

typedef struct
{
    int mem[MAXS][MAXS];
    int height;
    int width;
} Matrix;

typedef struct
{
    int lengths;
    int buffSize;
    Coordinate container[MAXS];
} Buffer;

int compareRewards(const void *a, const void *b);

void readMatrix(FILE *file, Matrix *matrix, int width, int height);

void readSequences(FILE *file, sequence *sequences, int *leng);

void displayMatrix(Matrix matrix, int row, int col);

bool sameCoordinateToken(int token, Matrix m, Coordinate p);

bool isValidBuffer(Buffer b, Matrix m, sequence s);

int bufferScore(Buffer b, Matrix m, sequence *l, int numSequences);


void createCoordinate(Coordinate *coord, int a, int b);

void push(Buffer *b, Coordinate p);

void pop(Buffer *b);

bool isValidPoint(Buffer b, int x, int y);

void solvePuzzle(Matrix matrix, sequence *sequences, int numSequences, Buffer *buffer, int *finalScore);

void displayCoordinate(Coordinate coord);

void displayBuff(Buffer b);

void displayBuffMat(Buffer b, Matrix m);

void saveText(Buffer buffer, int t, int totalscore, Matrix matrixs);

unsigned int generateSeed();

// Function to build a random matrix
void buildMatrix(Matrix *matrixs, int row, int col, int *token, int total);

// Function to build a random sequence
void buildSequence(sequence *sequences, int maxSeq, int *token, int total);


#endif // PUZZLE_SOLVER_H

```

BABIII

SCREENSHOTHASILTEST

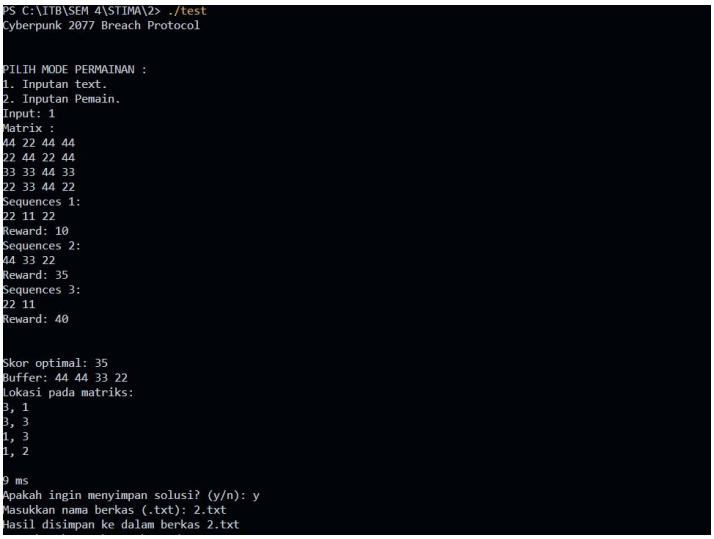
1. Test Case I

Masukan diterima melalui teks.



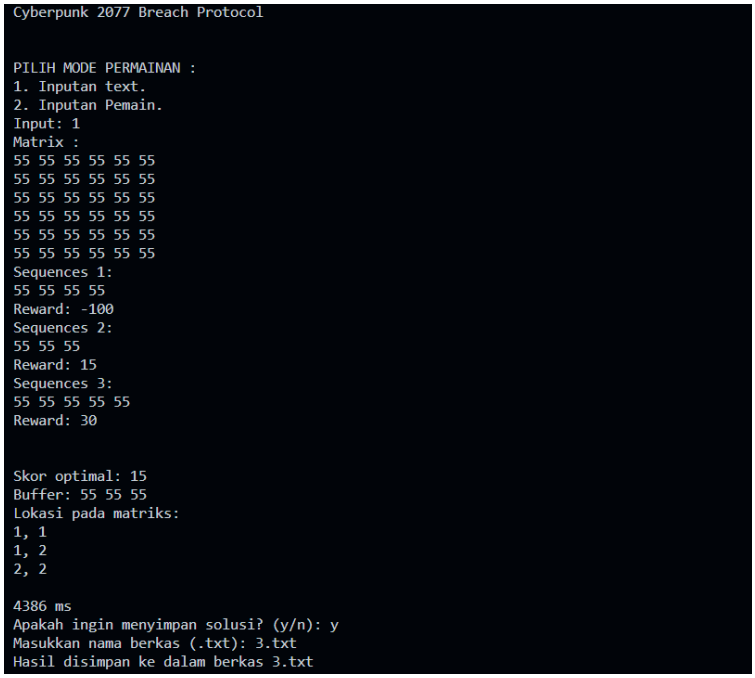
2. Test Case II

Masukan diterima melalui teks



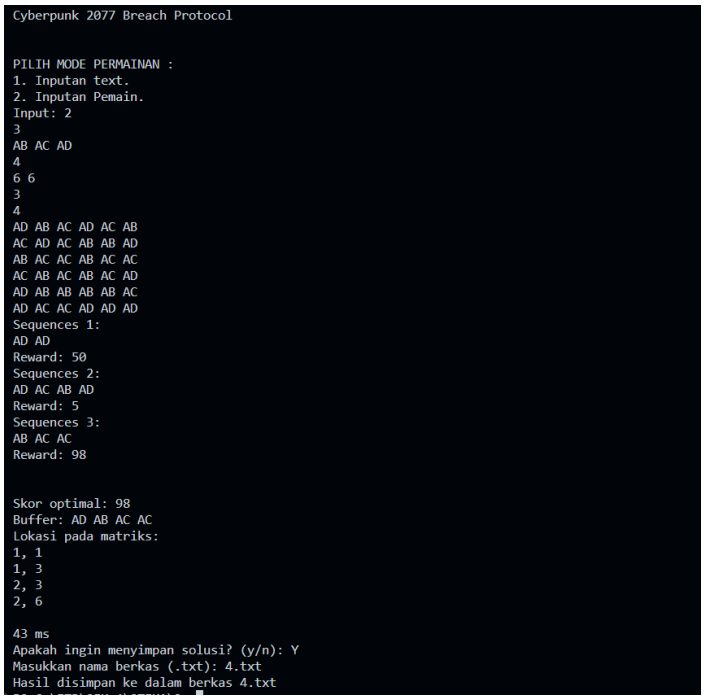
3. Test Case III

Masukan diterima melalui berkas.



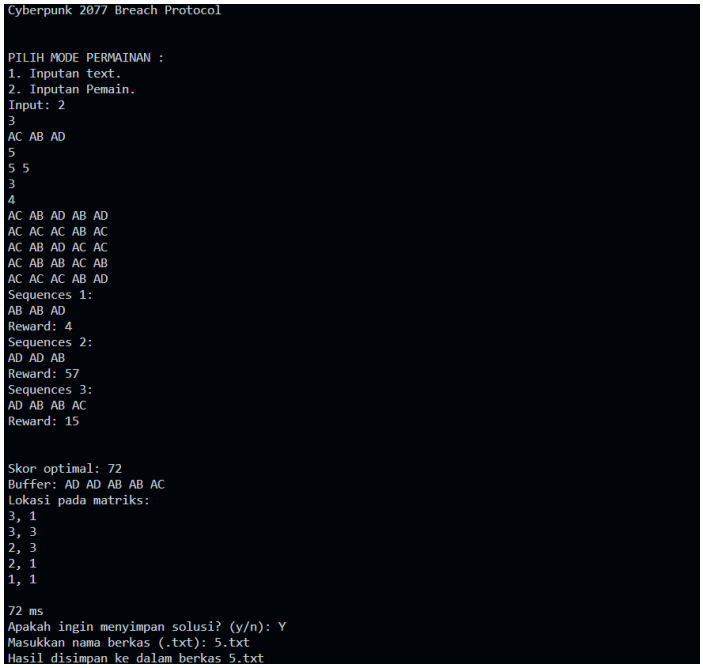
4. Test Case IV

Masukan memiliki *reward*diterima melalui *keyboard*.



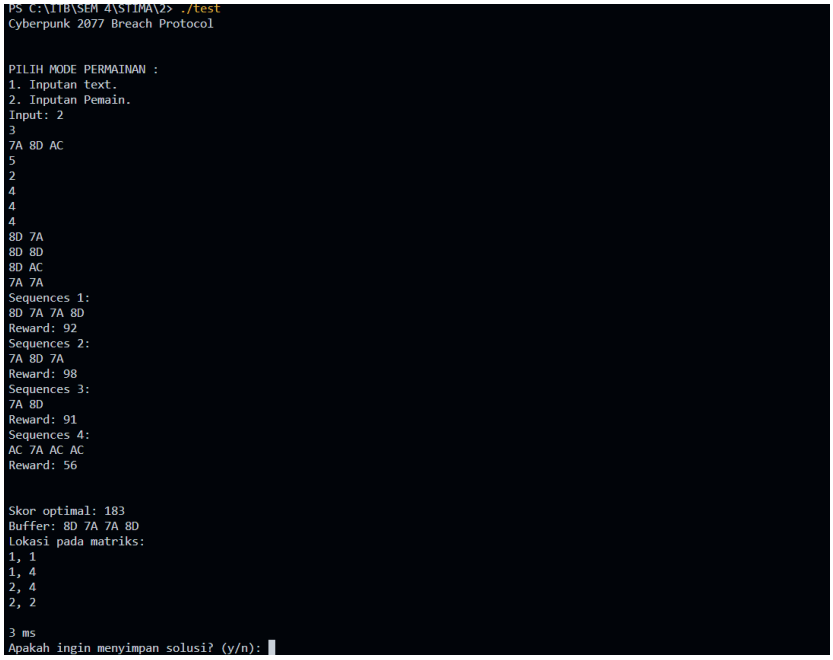
5. Test Case V

Masukan diterima melalui keyboard.



6. Test Case VI

Masukan diterima melalui *keyboard*.



LINK REPOSITORI

Pranala ke *repository*: https://github.com/EkaaPrawiraa/Tucil-Stima

LAMPIRAN

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		✓