

LAPORAN TUGAS KECIL I
IF2211 STRATEGI ALGORITMA

Penyelesaian *Cyberpunk 2077 Breach Protocol* dengan Algoritma Brute Force



Disusun oleh:

Indraswara Galih Jayanegara 1352119

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2024

DAFTAR ISI

Bagian 1 ALGORITMA BRUTE FORCE	3
Bagian 2 SOURCE CODE.....	4
Bagian 3 SCREENSHOT TEST.....	13
Input Keyboard	14
Input file.....	20
Link Repository	27
Checklist	27

Bagian 1

ALGORITMA BRUTE FORCE

Algoritma *Brute Force* adalah algoritma yang dengan pendekatan yang lempang untuk mencari sebuah solusi dari suatu masalah, Algoritma ini bergantung pada kekuatan komputasi karena semakin banyak langkah yang dilakukan semakin besar pula komputasinya. Dalam penyelesaian ***Breach Protocol Cyberpunk 2077*** dengan pendekatan *Brute force*, algoritma yang digunakan adalah sebagai berikut:

- Misalkan ada *sequence* yang dibentuk dari token unik dan setiap *sequence* memiliki *reward* random.
contoh *sequence*
{AB, CD, EF}, 20
{87, G5, J8}, 30
{B8, AB, CD, EF}, 50
- Pada Matriks game akan ditelusuri semua jalurnya dengan maksimal token yang diambil dari matriks sebanyak *buffer* yang telah ditentukan yang ada.
- setiap token yang diambil dari matriks akan di-*compare* dengan *sequence* yang memiliki *reward*. Jika kombinasi dari token ada yang sama dengan *Sequence* maka *total prize* akan bertambah sebanyak nilai dari *reward sequence* tersebut.
- Setiap kombinasi yang memiliki total *reward* yang sama, tetapi memiliki langkah yang lebih sedikit, maka kombinasi yang diambil adalah kombinasi yang memiliki langkah yang lebih sedikit sehingga solusi menjadi lebih optimal.

Bagian 2

SOURCE CODE

Projek ini ditulis dalam Bahasa C++ dengan menggunakan library

- | | |
|--------------|----------------|
| 1. iostream | 6. random |
| 2. string | 7. chrono |
| 3. limits | 8. fstream |
| 4. vector | 9. climits |
| 5. algorithm | 10. sys/stat.h |

Di dalam file *main.cpp* ada 2 jenis modul, pertama modul yang memang diperuntukkan algoritma *brute force* dan ada yang diperuntukkan sebagai *miscellaneous*

Algoritma Brute Force:

1. randomToken: fungsi untuk merandomize matriks dari token user
2. board: fungsi untuk generate matriks secara random
3. seqGenerator: fungsi untuk generate sequence secara random beserta reward
4. comparing: fungsi untuk membandingkan kombinasi dari token pada matriks dengan sequence
5. dfs: fungsi mencari kombinasi pada matriks

Miscellaneous:

1. isExist: fungsi untuk mengecek folder ada atau tidak
2. inputer: fungsi validasi input dalam main looping
3. inputInt: fungsi input yang diberikan message
4. file: fungsi menulis result pada file
5. saving: fungsi untuk menentukan menyimpan file atau tidak
6. isFound: fungsi yang menghasilkan keluaran dari algoritma *brute force*

Source Code-nya:

```
#include <iostream>
#include <string>
#include <limits>
#include <vector>
#include <algorithm>
#include <random>
#include <chrono>
#include <fstream>
#include <climits>
#include <sys/stat.h>

using namespace std;
using namespace std::chrono;

int maxPrize = INT_MIN;
int minMove = INT_MAX;

struct Path{
    int row;
    int col;
};

struct Reward{
    vector<string> sequence;
    int prize;
};

struct PathVal{
    int row;
    int col;
    string finalToken;
};

vector<Path> globalPath;
vector<string> globalCurrPos;
vector<PathVal> finalePath;

bool isExist(string name){
    struct stat buffer;
    return (stat(name.c_str(), &buffer) == 0);
}

int inputer(int awal, int akhir){
    string input;
    while (true) {
        cout << ">> ";
        cin >> input;
        try {
            int number = stoi(input);

            if(number >= awal && number <= akhir){
                return number;
            }
            else{
                cout << "Input invalid, masukkan ulang" << endl;
            }
        }
        catch (invalid_argument const &e) {
            cout << "Invalid input, masukkan ulang dengan angka" << endl;
        }
    }
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    cin >> input;

    return -1;
}
```

```

int inputInt(string msg){
    string input;
    while (true) {
        cout << msg;
        cin >> input;
        try {
            int number = stoi(input);

            if(number >= 0){
                return number;
            }
            else{
                cout << "Input invalid, masukkan ulang" << endl;
            }
        }
        catch (invalid_argument const &e) {
            cout << "Invalid input, masukkan ulang dengan angka" << endl;
        }
    }
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    cin >> input;

    return -1;
}

```

```

string randomToken(vector<string> tokens){
    int index = 0 + (rand() % (tokens.size() - 1));
    return tokens[index];
}

vector<vector<string>> board(int cols, int rows, vector<string> token){
    vector<vector<string>> board;

    for(int i = 0; i < rows; i++){
        vector<string> temp;
        for(int j = 0; j < cols; j++){
            temp.push_back(randomToken(token));
        }
        board.push_back(temp);
    }
    return board;
}

```

```

vector<Reward> seqGenerator(int maksSequence, vector<string> token, int jumlahSequence){
    int min = 2;
    vector<Reward> result;
    for(int i = 0; i < jumlahSequence; i++){
        vector<string> temp;
        random_device rd;
        mt19937 gen(rd());
        uniform_int_distribution<int> resultDis(0, token.size() - 1);
        uniform_int_distribution<int> resultLength(min, maksSequence);
        int length = resultLength(gen);
        for(int i = 0 ; i < length; i++){
            int index = resultDis(gen);
            temp.push_back(token[index]);
            // cout << gen << index << endl;
        }
        int value = 1 + (rand() % 100);
        result.push_back({temp, value});
    }
    return result;
}

```

```

void comparing(vector<string>& currentCombination, vector<Reward>& matrix, vector<Path> path) {
    int point = 0;
    for (auto& reward : matrix) {
        vector<string>& combination = reward.sequence;

        auto it = search(currentCombination.begin(), currentCombination.end(), combination.begin(), combination.end());
        if (it != currentCombination.end()) {
            point += reward.prize;
            if (point > maxPrize){
                finalePath.clear();
                maxPrize = point;
                // minMove = depth;
                for (int i = 0; i < path.size(); i++){
                    finalePath.push_back({path[i].row, path[i].col, currentCombination[i]});
                }
            }
            if (point == maxPrize && path.size() < minMove){
                finalePath.clear();
                minMove = path.size();
                for (int i = 0; i < path.size(); i++){
                    finalePath.push_back({path[i].row, path[i].col, currentCombination[i]});
                }
            }
        }
    }
}
}

```

```

void dfs(vector<vector<string>>& board, vector<Reward> base, vector<Path> path, vector<string> currentPos, int curRow, int curCol, int currDepth, int buffer, bool isHorizontal){
    int prizeTotal = 0;
    for (int i = 0; i < base.size(); i++){
        prizeTotal += base[i].prize;
    }
    if (prizeTotal == maxPrize){
        return;
    }

    if (curRow < 0 || curRow >= board.size() || curCol < 0 || curCol >= board[0].size()){
        return;
    }

    for (int i = 0; i < path.size(); i++){
        if (curRow == path[i].row && curCol == path[i].col){
            return;
        }
    };

    path.push_back({curRow, curCol});
    currentPos.push_back(board[curRow][curCol]);
    comparing(currentPos, base, path);

    if (currDepth == buffer){
        return;
    }

    if (isHorizontal){
        for (int i = 1; i < board.size(); i++){
            dfs(board, base, path, currentPos, curRow + i, curCol, currDepth + 1, buffer, false);
            dfs(board, base, path, currentPos, curRow - i, curCol, currDepth + 1, buffer, false);
        }
    }
    else{
        for (int i = 1; i < board[0].size(); i++){
            dfs(board, base, path, currentPos, curRow, curCol + i, currDepth + 1, buffer, true);
            dfs(board, base, path, currentPos, curRow, curCol - i, currDepth + 1, buffer, true);
        }
    }

    path.pop_back();
    currentPos.pop_back();
}

```

```

void file(string filename, auto duration){
    filename += ".txt";
    fstream savingData;
    savingData.open(filename, ios::out);

    savingData << maxPrize << endl;
    for (int i = 0; i < finalePath.size(); i++){
        savingData << finalePath[i].finalToken << " ";
    }
    savingData << "\n";
    for (int i = 0; i < finalePath.size(); i++){
        savingData << "(" << finalePath[i].col + 1 << ", " << finalePath[i].row + 1 << ")" << "\n";
    }

    savingData << "\n";
    savingData << duration << " ms";
    savingData.close();
}

```



```

int main(){
    cout<< "
    << "
    << " | / / / _ \ \ / / _ \ \ / / _ \ \ " << endl
    << " | / / / _ \ \ / / _ \ \ / / _ \ \ " << endl
    << " | / / _ \ \ / / _ \ \ / / _ \ \ " << endl;

    while(true){
        cout << "
        << " / _ ) _ _ _ _ _ / / " << endl
        << " / _ / _ _ \ \ _ / _ _ \ \ " << endl
        << " / / / / _ \ \ / / / / " << endl
        << " / _ \ \ _ \ \ / \ \ / / / " << endl
        << " / _ \ \ _ \ \ / / _ _ _ / / " << endl
        << " / / / _ _ \ \ / _ \ \ / _ \ \ / " << endl
        << " / _ \ \ / / / / / / / / / " << endl
        << " / / _ \ \ / \ \ / \ \ / \ \ / " << endl;

        //reset global variable
        cout << "choose input method: " << endl;
        globalCurrPos.clear();
        globalPath.clear();
        finalePath.clear();
        maxPrize = INT_MIN;
        minMove = INT_MAX;

        int jumlahTokenUnik, buffer, rowsMatriks, colsMatriks, jumlahSekuens, maksSekuens;
        vector<string> token;

        cout << "1. cli" << endl;
        cout << "2. file" << endl;
        cout << "3. quit" << endl;
        int input = inputer(1, 3);
    }
}

```

```

if(input == 1){
    jumlahTokenUnik = inputInt("Masukkan jumlah token unik: ");
    buffer = inputInt("Masukkan jumlah buffer: ");
    rowsMatriks = inputInt("Masukkan jumlah baris matriks: ");
    colsMatriks = inputInt("Masukkan jumlah kolom matriks: ");
    jumlahSekuens = inputInt("Masukkan total sekuens: ");
    maksSekuens = inputInt("Masukkan jumlah maksimum sekuens: ");
    cout << "Masukkan token: ";
    for(int i = 0; i < jumlahTokenUnik; i++){
        string seq;
        cin >> seq;
        token.push_back(seq);
    }

    // generate sequence reward and matriks board
    vector<vector<string>> boardGame = board(rowsMatriks, colsMatriks, token);
    vector<Reward> rewardSequence = seqGenerator(maksSekuens, token, jumlahSekuens);

    // reward debug
    for(int i = 0; i < rewardSequence.size(); i++){
        cout << "{ ";
        for(int j = 0; j < rewardSequence[i].sequence.size(); j++){
            cout << rewardSequence[i].sequence[j];
            if(j != rewardSequence[i].sequence.size() - 1){
                cout << ", ";
            }
        }
        cout << "}, ";
        cout << rewardSequence[i].prize;
        cout << " ";
        cout << endl;
    }

    auto start = high_resolution_clock::now();
    // main execution
    vector<string> currentPos;
    vector<Path> path;
    for(int i = 0; i < colsMatriks; i++){
        dfs(boardGame, rewardSequence, path, currentPos, 0, i, 0, buffer, true);
    }
    auto stop = high_resolution_clock::now();
    auto duration = duration_cast<milliseconds>(stop - start);
    // result
    isFound(duration, boardGame);
}

```

```

else if(input == 2){
    bool isValid = 1;
    vector<vector<string>> matriks;
    string line;
    string filePath = "test/";
    string fileName;
    vector<string> lines;
    fstream myData;

    cout << "Pastikan file ada di dalam folder test" << endl;
    cout << "tambahkan ekstensi file\ncontoh: data.txt\n";
    cout << "Masukkan nama file: ";
    cin >> fileName;
    filePath += fileName;
    while(!isExist(filePath)){
        cout << "file tidak ada di folder test" << endl;
        cout << "Masukkan nama file: ";
        cin >> fileName;
        filePath = "test/";
        filePath += fileName;
    }
    cout << "\n\nMemproses file " << fileName << endl;

    myData.open(filePath);
    if(myData.is_open()){
        while(getline(myData, line)){
            lines.push_back(line);
        }
    }
    myData.close();
    try{
        buffer = stoi(lines[0]);
    }catch(const invalid_argument& e){
        cerr << "file tidak sesuai format: buffer bukanlah angka";
        isValid = 0;
    }
}

```

```

    try{
        rowsMatriks = stoi(tempString);
    }catch(const invalid_argument& e){
        cerr << "file tidak sesuai format: rowsMatriks bukanlah angka\n";
        isValid = 0;
    }
    tempString.clear();
    for(int i = lengthBreak + 1; i < lines[1].size(); i++){
        if(lines[1][i] == ' ' || lines[1][i] == '\r'){
            break;
        }
        tempString += (lines[1][i]);
    }
}

```

```

    try{
        colsMatriks = stoi(tempString);
        tempString.clear();
    }catch(const invalid_argument& e){
        cerr << "file tidak sesuai format: colsMatriks bukanlah angka\n";
        isValid = 0;
    }

    string myStr;
    vector<string> temp;
    if(rowsMatriks > lines.size()){
        cout << "file tidak sesuai format: rowMatriks terlalu banyak\n";
        isValid = 0;
    }

    if(isValid){
        for(int i = 2; i <= rowsMatriks + 1; i++){
            if(isValid){
                temp.clear();
                for(int j = 0; j < lines[i].size(); j++){
                    if(isValid){
                        if(lines[i][j] != ' '){
                            myStr.push_back(lines[i][j]);
                            if(myStr.back() == '\r'){
                                myStr.pop_back();
                            }
                        }
                    }
                    if(lines[i][j] == ' ' || j == lines[i].size() - 1){
                        if(myStr.size() != 2){
                            isValid = 0;
                            cout << "file tidak sesuai format: token matriks tidak sesuai" << endl;
                            break;
                        }

```

```

if(isValid){
    for(int i = 2; i <= rowsMatriks + 1; i++){
        if(isValid){
            temp.clear();
            for(int j = 0; j < lines[i].size(); j++){
                if(isValid){
                    if(lines[i][j] != ' '){
                        myStr.push_back(lines[i][j]);
                        if(myStr.back() == '\r'){
                            myStr.pop_back();
                        }
                    }
                    if(lines[i][j] == ' ' || j == lines[i].size() - 1){
                        if(myStr.size() != 2){
                            isValid = 0;
                            cout << "file tidak sesuai format: token matriks tidak sesuai" << endl;
                            break;
                        }
                        temp.push_back(myStr);
                        myStr.clear();
                    }
                }
            }
            matriks.push_back(temp);
            if(matriks[i-2].size() != colsMatriks){
                cout << "file tidak sesuai format: jumlah kolom tidak sesuai dengan input" << endl;
                isValid = 0;
                break;
            }
        }
    }
}
}

```

```

if(isValid){
    for(int f = rowsMatriks + 3; f < lines.size(); f++){
        Reward tempReward;
        //ini reward
        if(isValid){
            temp.clear();
            //ini sequence
            for(int j = 0; j < lines[f].size(); j++){
                if(lines[f][j] != ' '){
                    myStr.push_back(lines[f][j]);
                    if(myStr.back() == '\r'){
                        myStr.pop_back();
                    }
                }
                if(lines[f][j] == ' ' || j == lines[f].size() - 1){
                    if(myStr.size() != 2){
                        cout << "file tidak sesuai format: sequence token tidak berjumlah 2" << endl;
                        isValid = 0;
                    }
                    temp.push_back(myStr);
                    myStr.clear();
                }
            }
            tempReward.sequence = temp;

            f += 1;
            try{
                tempReward.prize = stoi(lines[f]);
            }
            catch(const invalid_argument& e){
                cerr << "file tidak sesuai format: baris matriks tidak sesuai dengan input" << endl;
                cerr << "file tidak sesuai format: reward sequence bukanlah angka" << endl;
                isValid = 0;
            }
            rewardSequence.push_back(tempReward);
        }
    }
}

```

```

    if(rewardSequence.size() != jumlahSekuens && isValid){
        cout << "file tidak sesuai format: jumlah sekuens tidak sesuai" << endl;
        isValid = 0;
    }

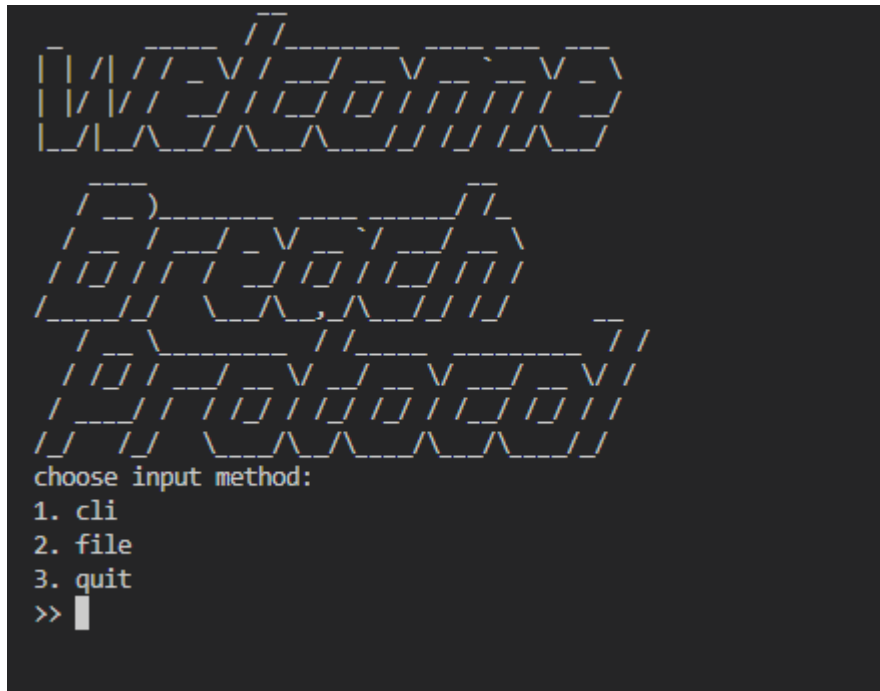
    if(isValid){
        // main execution
        auto start = high_resolution_clock::now();
        vector<string> currentPos;
        vector<Path> path;
        for(int i = 0; i < colsMatriks; i++){
            dfs(matriks, rewardSequence, path, currentPos, 0, i, 0, buffer, true);
        }
        auto stop = high_resolution_clock::now();
        auto duration = duration_cast<milliseconds>(stop - start);
        isFound(duration, matriks, rewardSequence);
    }else{
        cout << "\ncek kesalahan satu per satu" << endl;
    }

    // result
}
else{
    break;
}
}
return 0;
}

```

Bagian 3

SCREENSHOT TEST



Input Keyboard

Input 1:

```
Masukkan jumlah token unik: 5
Masukkan jumlah buffer: 6
Masukkan jumlah baris matriks: 6
Masukkan jumlah kolom matriks: 6
Masukkan total sekuens: 4
Masukkan jumlah maksimum sekuens: 5
Token harus kombinasi 1 angka 1 huruf, 2 angka, atau 2 huruf
Contoh: A8, BG, 97
Masukkan token: AB CD 7F HG 8F
{{HG,8F,CD,HG,CD}, 91}
{{AB,HG,8F}, 30}
{{AB,8F,CD}, 71}
{{AB,8F}, 51}
```

```
##  ## ##### ##### ##### ##### ##  ## ##  ## #####
### ## ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##
## ## ## ##  ##  ##### ##  ##  ##  ##  ##  ##
## ##### ##  ##  ##  ##  ##  ##  ##  ##  ##
##  ## #####  ##  ##  #####  ##### ##  ##  #####
```

```
Optimum reward:
total prize: 0
minimum buffer taken: 0
```

```
Execution Time: 846 milliseconds
simpan hasil dalam bentuk file? (y/n) y
Masukkan nama file: result1k
result1k telah tersimpan di folder test
```

```
test > result1k.txt
```

```
1  0
2
3
4  846 ms
```

Input2:

```
Masukkan jumlah token unik: 7
Masukkan jumlah buffer: 6
Masukkan jumlah baris matriks: 6
Masukkan jumlah kolom matriks: 6
Masukkan total sekuens: 5
Masukkan jumlah maksimum sekuens: 6
Token harus kombinasi 1 angka 1 huruf, 2 angka, atau 2 huruf
Contoh: A8, BG, 97
Masukkan token: AB 7U 4R 6D 3E 9I OP
{{9I,3E,OP,7U,AB}, 9}
{{6D,AB,OP}, 45}
{{OP,AB,9I}, 10}
{{9I,9I,4R}, 90}
{{6D,3E,4R,7U}, 3}
```

FOUND

Result

Matriks:

```
6D AB 6D AB 3E 7U
AB 9I 7U AB 4R AB
4R 7U 9I 4R 7U AB
7U 9I 6D AB 9I 7U
3E 9I 4R 9I 6D 4R
9I 7U 4R 3E 6D 4R
```

Reward Sequence:

```
9I 3E OP 7U AB (9)
6D AB OP (45)
OP AB 9I (10)
9I 9I 4R (90)
6D 3E 4R 7U (3)
```

Token combination: 6D 3E 4R 7U 9I 9I 4R

(1,1)

(1,5)

(3,5)

(3,2)

(2,2)

(2,5)

(6,5)

Optimum reward:

total prize: 93

minimum buffer taken: 7

test > resultk2.txt

```
1 93
2 6D 3E 4R 7U 9I 9I 4R
3 (1,1)
4 (1,5)
5 (3,5)
6 (3,2)
7 (2,2)
8 (2,5)
9 (6,5)
10
11 1025 ms
```

Input 3:

```
Masukkan jumlah token unik: 4
Masukkan jumlah buffer: 6
Masukkan jumlah baris matriks: 7
Masukkan jumlah kolom matriks: 7
Masukkan total sekuens: 3
Masukkan jumlah maksimum sekuens: 4
Token harus kombinasi 1 angka 1 huruf, 2 angka, atau 2 huruf
Contoh: A8, BG, 97
Masukkan token: AB 76 D9 IF
{{76,76,76}, 75}
{{76,D9,IF,76}, 21}
{{76,AB,AB,D9}, 97}
```

FOUND

Result

Matriks:

```
AB D9 D9 AB 76 D9 76
D9 76 AB 76 76 76 AB
D9 76 AB 76 D9 AB AB
D9 D9 76 D9 D9 AB D9
76 D9 76 AB D9 AB D9
AB D9 76 D9 AB D9 D9
AB 76 AB D9 D9 76 D9
```

Reward Sequence:

76 76 76 (75)

76 D9 IF 76 (21)

76 AB AB D9 (97)

Token combination: 76 76 76 AB AB D9

(5,1)

(5,2)

(6,2)

(6,3)

(7,3)

(7,4)

Optimum reward:

total prize: 172

minimum buffer taken: 6

Execution Time: 2691 milliseconds

```
test > resultk3.txt
1 172
2 76 76 76 AB AB D9
3 (5,1)
4 (5,2)
5 (6,2)
6 (6,3)
7 (7,3)
8 (7,4)
9
10 2691 ms
```


Input 4:

```
Masukkan jumlah token unik: 6
Masukkan jumlah buffer: 7
Masukkan jumlah baris matriks: 6
Masukkan jumlah kolom matriks: 6
Masukkan total sekuens: 5
Masukkan jumlah maksimum sekuens: 6
Token harus kombinasi 1 angka 1 huruf, 2 angka, atau 2 huruf
Contoh: A8, BG, 97
Masukkan token: AB FG HU 7Y 8U IF
{{AB,IF,8U,AB,HU}, 94}
{{IF,8U,7Y,AB,HU,AB}, 57}
{{HU,7Y,AB,7Y}, 12}
{{HU,8U,IF,AB}, 43}
{{8U,7Y,IF}, 30}
```

FOUND

Result

Matriks:

```
7Y FG HU AB 7Y AB
FG HU 8U FG HU HU
AB 8U 7Y FG AB FG
HU FG FG 7Y HU 8U
HU AB HU 7Y HU AB
8U HU HU 7Y 8U HU
```

Reward Sequence:

```
AB IF 8U AB HU (94)
IF 8U 7Y AB HU AB (57)
HU 7Y AB 7Y (12)
HU 8U IF AB (43)
8U 7Y IF (30)
```

Token combination: HU 7Y AB 7Y

(3,1)

(3,3)

(5,3)

(5,1)

Optimum reward:

total prize: 12

minimum buffer taken: 4

Execution Time: 7364 milliseconds

test > resultk4.txt

```
1 12
2 HU 7Y AB 7Y
3 (3,1)
4 (3,3)
5 (5,3)
6 (5,1)
7
8 7364 ms
```

Input 5:

```
Masukkan jumlah token unik: 8
Masukkan jumlah buffer: 8
Masukkan jumlah baris matriks: 8
Masukkan jumlah kolom matriks: 8
Masukkan total sekuens: 7
Masukkan jumlah maksimum sekuens: 6
Token harus kombinasi 1 angka 1 huruf, 2 angka, atau 2 huruf
Contoh: A8, BG, 97
Masukkan token: 6F TU YD HD H8 43 A4 K9
{{6F,K9,6F}}, 89}
{{K9,K9,TU,6F,A4,H8}}, 7}
{{YD,YD,43,6F,TU,A4}}, 41}
{{YD,K9,A4,A4,YD}}, 43}
{{6F,K9,A4}}, 65}
{{H8,H8,43,YD}}, 49}
{{H8,K9,A4,TU,6F,43}}, 47}
```

FOUND

Result

Matriks:

```
A4 TU A4 43 HD YD 43 6F
43 A4 6F 43 A4 A4 6F TU
A4 6F H8 H8 YD YD HD A4
43 A4 43 43 A4 TU TU 43
TU YD 6F H8 43 6F H8 HD
HD H8 H8 YD 43 YD 6F H8
TU TU H8 TU 6F H8 HD 6F
TU H8 43 TU HD H8 6F TU
```

Reward Sequence:

```
6F K9 6F (89)
K9 K9 TU 6F A4 H8 (7)
YD YD 43 6F TU A4 (41)
YD K9 A4 A4 YD (43)
6F K9 A4 (65)
H8 H8 43 YD (49)
H8 K9 A4 TU 6F 43 (47)
```

Token combination: A4 H8 H8 43 YD

```
(3,1)
(3,3)
(4,3)
(4,1)
(6,1)
```

Optimum reward:

```
total prize: 49
minimum buffer taken: 5
```

```
test > resultk5.txt
```

```
1 49
2 A4 H8 H8 43 YD
3 (3,1)
4 (3,3)
5 (4,3)
6 (4,1)
7 (6,1)
8
9 959981 ms
```

Input 6:

```
Masukkan jumlah token unik: 6
Masukkan jumlah buffer: 6
Masukkan jumlah baris matriks: 8
Masukkan jumlah kolom matriks: 8
Masukkan total sekuens: 4
Masukkan jumlah maksimum sekuens: 6
Token harus kombinasi 1 angka 1 huruf, 2 angka, atau 2 hurufContoh: A8, BG, 97
Masukkan token: 7K 8F AA BH FK LK
{{FK,FK,AA}, 89}
{{FK,8F,AA,7K,FK,LK}, 7}
{{7K,FK,7K,AA}, 41}
{{BH,8F}, 43}

FOUNDO

Result
Matriks:
8F AA FK 7K FK FK BH BH
AA FK 7K 7K 8F AA 8F 8F
7K AA AA 8F 8F FK AA BH
AA AA 8F 8F BH 7K AA 8F
8F BH FK AA AA FK 7K FK
BH 8F AA BH BH FK 8F 8F
BH BH AA FK AA AA FK
BH 8F FK BH 8F 7K 7K AA
Reward Sequence:
FK FK AA (89)
FK 8F AA 7K FK LK (7)
7K FK 7K AA (41)
BH 8F (43)
Token combination: FK FK AA BH 8F
(3,1)
(3,5)
(4,5)
(4,6)
(2,6)
Optimum reward:
total prize: 132
minimum buffer taken: 5
Execution Time: 9338 milliseconds
```

Input file

Input: data.txt

```
content: data.txt
Masukkan nama file: data.txt

Memproses file data.txt

FOUNDO

Result
Matriks:
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
Reward Sequence:
BD E9 1C (15)
BD 7A BD (20)
BD 1C BD 55 (30)
Token combination: 7A BD 7A BD 1C BD 55
(1,1)
(1,4)
(3,4)
(3,5)
(6,5)
(6,4)
(5,4)
Optimum reward:
total prize: 50
minimum buffer taken: 7
```

```
result7.txt U ×  main.cpp M  result.txt U ×  result2.txt U  README.md M ●
test > result.txt
1 50
2 7A BD 7A BD 1C BD 55
3 (1,1)
4 (1,4)
5 (3,4)
6 (3,5)
7 (6,5)
8 (6,4)
9 (5,4)
10
11 4306 ms
```

Input: data2.txt

Masukkan nama file: data2.txt

Memproses file data2.txt

FOUND

Result

Matriks:

```
42 69 69 AD BE AA AA
42 DE CC CC BE 42 42
AD BE EF 69 CC DE EF
CC 42 EF BE BE EF CC
42 CC AD EF AD 42 DE
EF CC DE 42 42 AA CC
BE AA AA BE CC 69 69
```

Reward Sequence:

```
DE 69 AA AD (87)
42 BE 42 EF AA (16)
AD DE 42 DE DE EF (52)
AD DE 42 AD CC (86)
EF AA AA (30)
```

Token combination: 42 AD DE 69 AA AD

(1,1)

(1,3)

(6,3)

(6,7)

(3,7)

(3,5)

Optimum reward:

total prize: 87

minimum buffer taken: 6

Execution Time: 29151 milliseconds

simpan hasil dalam bentuk file? (y/n) y

Masukkan nama file: result2

result2 telah tersimpan di folder test

result7.txt U main.cpp M result2.txt U X README.md M ●

test > result2.txt

```
1 87
2 42 AD DE 69 AA AD
3 (1,1)
4 (1,3)
5 (6,3)
6 (6,7)
7 (3,7)
8 (3,5)
9
10 27416 ms
```

input: data3.txt

```
Masukkan nama file: data3.txt

Memproses file data3.txt

FOUNDO

Result
Matriks:
69 42 00 69 00 EF
BE AD BE AD EF 42
42 42 DE 69 EF AD
BE 69 DE BE AD 42
BE 42 EF BE BE DE
00 00 00 EF 00 BE
Reward Sequence:
AD 69 42 69 BE (17)
69 AD DE AD (100)
42 AD 00 (97)
DE EF AD 00 42 (3)
Token combination: EF 42 AD 00
(6,1)
(6,2)
(2,2)
(2,6)
Optimum reward:
total prize: 97
minimum buffer taken: 4

Execution Time: 4493 milliseconds
simpan hasil dalam bentuk file? (y/n) y
Masukkan nama file: result3
result3 telah tersimpan di folder test

test > result3.txt
1 97
2 EF 42 AD 00
3 (6,1)
4 (6,2)
5 (2,2)
6 (2,6)
7
8 4493 ms
```

input: data4.txt

```
Masukkan nama file: data4.txt

Memproses file data4.txt

FOUND

Result
Matriks:
UF DO A9 B3 74 J1
DD 7Y DB L4 QD 8A
BG 61 W7 QN TJ 91
C8 UO T0 PA UA D4
CQ 11 R2 DQ KN CF
98 IQ TL AX 9I 8E
Reward Sequence:
AB UO AB UO AB (59)
UF 7Y 9I 9I (6)
BG UF (-50)
UF AB (92)
9I 9I 7Y BG (3)
Token combination: DO 61 BG UF
(2,1)
(2,3)
(1,3)
(1,1)
Optimum reward:
total prize: 0
minimum buffer taken: 4

test > result4.txt
1 0
2 DO 61 BG UF
3 (2,1)
4 (2,3)
5 (1,3)
6 (1,1)
7
8 5192 ms
```

input: data5.txt

```
Masukkan nama file: data5.txt
```

```
Memproses file data5.txt
```

```
FOUND
```

```
Result
```

```
Matriks:
```

```
AB Y4 H5 UI YK HT
```

```
PF PN 7Q 30 R0 E9
```

```
KU XW QN BI BI X4
```

```
UF 7U BF 2L L7 L7
```

```
TS J9 AX L7 BL PM
```

```
4B E6 0A 4E 44 E9
```

```
Reward Sequence:
```

```
AB GH AB J9 J9 KU (91)
```

```
UF UF J9 KU GH (30)
```

```
AB J9 UF J9 UF L7 (71)
```

```
KU AB (-100)
```

```
J9 UF GH (7)
```

```
Token combination: Y4 XW KU AB
```

```
(2,1)
```

```
(2,3)
```

```
(1,3)
```

```
(1,1)
```

```
Optimum reward:
```

```
total prize: 0
```

```
minimum buffer taken: 4
```

```
Execution Time: 23447 milliseconds
```

```
test > result5.txt
```

```
1 0
```

```
2 Y4 XW KU AB
```

```
3 (2,1)
```

```
4 (2,3)
```

```
5 (1,3)
```

```
6 (1,1)
```

```
7
```

```
8 23447 ms
```


input: data6.txt

```
Masukkan nama file: data6.txt

Memproses file data6.txt

FOUND

Result
Matriks:
K9 D7 0F 7B 7B 15 7B J8
FF K9 0F FF 7B D7 7B K9
0F 7B 15 00 FF FF 15 15
0F FF J8 J8 D7 15 J8 K9
J8 0F K9 0F 15 7B D7 J8
00 K9 0F 00 0F FF 7B J8
J8 D7 K9 K9 00 00 15 FF
7B 00 7B 0F 7B 15 00 15
Reward Sequence:
J8 FF 15 J8 K9 (50)
K9 K9 00 7B 7B (30)
15 7B J8 K9 7B 7B (34)
15 FF D7 D7 (5)
K9 15 FF J8 7B FF (90)
D7 FF K9 (32)
Token combination: D7 FF K9 J8 FF 15 J8 K9
(2,1)
(2,4)
(8,4)
(8,6)
(6,6)
(6,4)
(4,4)
(4,7)
Optimum reward:
total prize: 82
minimum buffer taken: 8

Execution Time: 90897 milliseconds
simpan hasil dalam bentuk file? (y/n) y
Masukkan nama file: result6
result6 telah tersimpan di folder test

test > result6.txt
1 82
2 D7 FF K9 J8 FF 15 J8 K9
3 (2,1)
4 (2,4)
5 (8,4)
6 (8,6)
7 (6,6)
8 (6,4)
9 (4,4)
10 (4,7)
11
12 90897 ms
```

```

Result
Matriks:
K9 D7 0F 7B 7B 15 7B J8
FF K9 0F FF 7B D7 7B K9
0F 7B 15 00 FF FF 15 15
0F FF J8 J8 D7 15 J8 K9
J8 0F K9 0F 15 7B D7 J8
00 K9 0F 00 0F FF 7B J8
J8 D7 K9 K9 00 00 15 FF
7B 00 7B 0F 7B 15 00 15
D7 FF K9 J8 FF 15 J8 K9
(2,1)
(2,4)
(8,4)
(8,6)
(6,6)
(6,4)
(4,4)
(4,7)
Optimum reward:
total prize: 88
minimum buffer taken: 8

Execution Time: 606981 milliseconds
simpan hasil dalam bentuk file? (y/n) y
Masukkan nama file: result7
result7 telah tersimpan di folder test

```

```
test > result7.txt
1      88
2      D7 FF K9 J8 FF 15 J8 K9
3      (2,1)
4      (2,4)
5      (8,4)
6      (8,6)
7      (6,6)
8      (6,4)
9      (4,4)
10     (4,7)
11
12     606981 ms
```

Link Repository

[Link repository github](#)

Checklist

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	•	•