Laporan Tugas Kecil 1 IF2211 Strategi Algoritma

Disusun oleh:

Yosef Rafael Joshua / 13522133

PROGRAM STUDI TEKNIK INFORMATIK

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2024

1. Algoritma Brute Force
   j
2. Source Code

```cpp
#include <iostream>
#include <string>
#include <typeinfo>
#include <fstream>
#include <sstream>
#include <list>
#include <random>
#include <tuple>
#include <algorithm>

using namespace std;


class Array_string{
    private:
        string arr[50];
        int effIndex; // efektif indeks. bukan max indeks

    public:
        Array_string(){
            setEffIdx(-1);
        }

        Array_string(int idx){
            setEffIdx(idx-1);
        }

        string getElement(int idx){
            if(idx <= effIndex){
                return arr[idx];
            } else{
                return "";
            }
        }

        void showEffIndex(){
            cout << "effidx: " << effIndex << endl;
        }

        void setEffIdx(int aIdx){
            if(aIdx <= 50){
                effIndex = aIdx;
            } else{
                cout << "index > 50" << endl;
            }
        }

        void setArrElement(string input, int index){
            arr[index] = input;
        }

        void printElmementArray(){
            for(int i = 0; i < effIndex+1; i++){
                cout << arr[i] << endl;
            }
        }
};

class Path{
    private:

        int length;
        int reward;

    public:
        list<tuple<int, int>> coordinate;
        Array_string tokens;
        // setter
        Path(){
            length = 0;
            reward = 0;
        }

        void setLength(int newLength){
            length = newLength;
        }

        void setReward(int newReward){
            reward = newReward;
        }

        //void add

        // getter
        int getLength(){
            return length;
        }

        int getReward(){
            return reward;
        }
        void add(int row, int col, string input){
            tuple a = make_tuple(col, row);
            if(find(coordinate.begin(), coordinate.end(), a) ==
coordinate.end()){
                setLength(getLength()+1);
                coordinate.push_back(a);
                tokens.setEffIdx(getLength()-1);
                tokens.setArrElement(input, getLength()-1);
            }
        }

        void set(int row, int col, string input, int position){
            tuple a = make_tuple(col, row);

            auto c = coordinate.begin();
            advance(c, position);
            *c = a;
            tokens.setArrElement(input, position);

        }

        void showInfo(){
            cout << "length: " << length << endl;
            cout << "reward: " << reward << endl;
            tokens.printElmementArray();
            cout << endl;
            for(auto temp : coordinate){
                int first = get<0>(temp);
                int second = get<1>(temp);
                cout << "(" << first << "," << second << ")" << endl;
            }
        }

};

class Matrix{
    private:
        int Col;
        int Row;

    public:
        Path path;
        string matriks[50][50];

        // getter
        Matrix(int aCol, int aRow){
            setCol(aCol);
            setRow(aRow);
```

```cpp
}

string getElement(int row, int col){
    return matriks[row][col];
}

int getCol(){
    return Col;
}

int getRow(){
    return Row;
}

// setter
void setCol(int col){
    if(col >= 1 && col <= 50){
        Col = col;
    } else{
        cout << "jumlah kolom tidak valid" << endl;
    }
}
void setRow(int row){
    if(row >= 1 && row <= 50){
        Row = row;
    } else{
        cout << "jumlah baris tidak valid" << endl;
    }
}

void setElement(int row, int col, string input){
    matriks[row][col] = input;
}

void printElementMatrix(){
    for(int i = 0; i < Row; i++){
        for(int j = 0; j < Col; j++){
            if(j == Col-1){
                cout << matriks[i][j] << endl;
            } else{
                cout << matriks[i][j] << " ";
            }
        }
    }
}
```

```
        void searchPath(int row, int col, bool is_vertical, int buffer_size, 
Path &aPath, int pos){
            if(buffer_size == 0){
                path.showInfo();
            }
            else{
                // set to path
                path.set(row, col, getElement(row, col), pos);

                if(!is_vertical){
                    for(int columns = 0; columns < Col; columns++){

                        tuple b = make_tuple(columns, row);
                        if(find(path.coordinate.begin(), 
path.coordinate.end(), b) == path.coordinate.end()){
                            searchPath(row, columns, true, buffer_size-1, 
aPath, pos+1);
                        }
                    }
                } else{ // is_vertical
                    for(int Rowss = 0; Rowss < Row; Rowss++){

                        tuple b = make_tuple(col, Rowss);
                        if(find(path.coordinate.begin(), 
path.coordinate.end(), b) == path.coordinate.end()){
                            searchPath(Rowss, col, false, buffer_size-1, 
aPath, pos+1);
                        }
                    }
                }


            }
        }
};

class Prized_sequence{
    private:
        string sequence;
        int reward;

    public:
        // getter
        Prized_sequence(string text, int num){
            setSequence(text);
            setReward(num);
        }
        string getSequence(){
            return sequence;
        }

        int getReward(){
            return reward;
        }

        // setter
        void setSequence(string aText){
            sequence = aText;
        }

        void setReward(int aNum){
            reward = aNum;
        }

};


int main(){

    //kamus
    int baris, kolom, buffer_size, number_of_sequences, tempReward;
    int number_of_unique_tokens, max_sequence_size;
    Matrix matriks(4,4);
    string pair, tempSequence, choice, tokens, data_file;
    list<Prized_sequence> pSequence;

    cout << "--------------------------------------------------" << endl;
    cout << "Selamat datang di Breach Protocol Solver!" << endl;
    cout << "Apakah Anda ingin memasukkan data dari file txt? (y/n)" << endl;

    cin >> choice;
    while(choice != "y" && choice != "n"){
        cout << "input tidak valid" << endl;
        cout << "Apakah Anda ingin memasukkan data dari file txt? (y/n)" <<
endl;
        cin >> choice;
    }

    if(choice == "y"){
        // data dari file txt
        cout << "Data akan dimasukkan dari file txt" << endl;
        cout << "Silahkan masukkan nama txt file yang akan dibaca dari folder
data" << endl;
        cout << "Format: {nama file.txt}" << endl;
        cin.ignore();
        cin >> data_file;
        fstream fileData;
        fileData.open("data/" + data_file, ios::in);
        if(fileData.is_open()){
            string line;
            getline(fileData, line);
            buffer_size = stoi(line);
            getline(fileData, line);
            istringstream stream1;
            stream1.str(line);
            stream1 >> kolom >> baris;
            stream1.clear();

            // set kolom dan baris dari data
            matriks.setCol(kolom);
            matriks.setRow(baris);


            for(int i = 0; i < baris; i++){
                getline(fileData, line);

                // masukin ke stream
                stream1.str(line);

                for(int j = 0; j < kolom; j++){
                    stream1 >> pair;
                    matriks.setElement(i, j, pair);
                }
                stream1.clear();
            }
            getline(fileData, line);
            number_of_sequences = stoi(line);


            for(int i = 0; i < number_of_sequences; i++){
                getline(fileData, line);
                tempSequence = line;
                getline(fileData, line);
                tempReward = stoi(line);

                pSequence.push_back(Prized_sequence(tempSequence,
tempReward));

            }

            fileData.close();
        }

        cout << "--------------------------------------------------" << endl;
        cout << "Matriks: " << endl;

        matriks.printElementMatrix();
        cout << endl;
        for(auto itterator = pSequence.begin(); itterator != pSequence.end();
++itterator){
            cout << "sequence: " << itterator -> getSequence() << endl;
            cout << "reward: " << itterator -> getReward() << endl;
        }

    } else{
        // data random generated
        cout << "Data akan dibuat secara acak" << endl;
        cout << "Jumlah token unik: "; cin >> number_of_unique_tokens;
        cout << "Token: "; cin.ignore(); getline(cin, tokens);
        cout << "Ukuran buffer: "; cin >> buffer_size;
        cout << "Ukuran matriks: "; cin >> kolom >> baris;
        cout << "Jumlah sekuens: "; cin >> number_of_sequences;
        cout << "Ukuran maksimal sekuens: "; cin >> max_sequence_size; cout <<
endl;

        // proses random generation
        Array_string Tokens_array(number_of_unique_tokens);
        string aToken;
        istringstream stream2;
        stream2.str(tokens);
        Tokens_array.showEffIndex();
        for(int i = 0; i < number_of_unique_tokens; i++){
            stream2 >> aToken;
            Tokens_array.setArrElement(aToken, i);
        }

        Tokens_array.printElmementArray();

        matriks.setCol(kolom);
        matriks.setRow(baris);

        random_device rd;
        uniform_int_distribution<int> dist_token(0, number_of_unique_tokens-
1);
        uniform_int_distribution<int> dist_sequence(2, max_sequence_size);


        int temp;
        for(int i = 0; i < baris; i++){
            for(int j = 0; j < kolom; j++){
                temp = dist_token(rd);
                matriks.setElement(i, j, Tokens_array.getElement(temp));
            }
        }

        // tampilan matriks dan sequence random generated
        cout << "Matriks: " << endl;
        matriks.printElementMatrix();
        cout << endl;

        int temp2;
        for(int i = 0; i < number_of_sequences; i++){
            temp = dist_sequence(rd);
            for(int j = 0; j < temp; j++){
                temp2 = dist_token(rd);
                if(j == 0){
                    tempSequence = Tokens_array.getElement(temp2);
                } else {//j != 0
                    tempSequence = tempSequence + " " +
Tokens_array.getElement(temp2);
                }
            }
            pSequence.push_back(Prized_sequence(tempSequence, temp*5));
        }

        for(auto itterator = pSequence.begin(); itterator != pSequence.end();
++itterator){
            cout << "sequence: " << itterator -> getSequence() << endl;
            cout << "reward: " << itterator -> getReward() << endl;
        }

    }
    string enter;
    cout << endl;
    cout << "ketik 'y' untuk melanjutkan" << endl;
    cin >> enter;

    // algoritma brute force
    Path optimized;

    for(int size = 2; size <= buffer_size; size++){

        matriks.path.setLength(size);
        matriks.path.tokens.setEffIdx(size-1);

        for(int i = 0; i < kolom; i++){
            matriks.path.coordinate.clear();
            for(int g = 0; g < size; g++){
                matriks.path.coordinate.push_back(make_tuple(-1, -1));
            }
            matriks.searchPath(0, i, true, size, optimized, 0);
        }
    }

    return 0;
}
```

3. Screenshot
4. Pranala Repository
   https://github.com/Syfoe/Tucil1_13522133