IF2211 Strategi Algoritma
Tugas Kecil 1
Penyelesaian Permainan Queens Linkedin

Dipersiapkan oleh:

13524033 Ray Owen Martin

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
JL. GANESA 10, BANDUNG 40132
2026

Daftar Isi

Daftar Tabel

Daftar Gambar

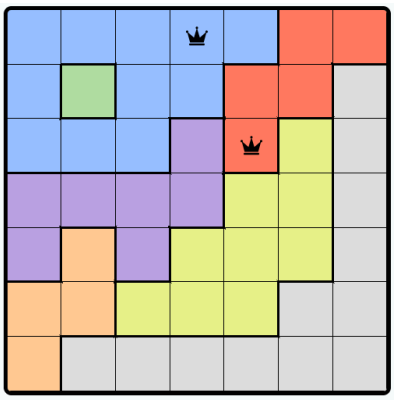
1 Pendahuluan

Permainan Queens yang dikembangkan oleh LinkedIn merupakan contoh permainan papan yang melatih kemampuan kognitif, berpikir kreatif, dan kemampuan membuat keputusan. Adapun permainan ini tersedia di platform online LinkedIn dan dapat dimainkan setiap harinya.

Permainan ini dibuat dalam suatu papan berisikan NxN kotak berwarna di mana pemain harus menempatkan sejumlah Queens berdasaran beberapa aturan ini:

- Setiap baris, kolom, dan daerah (yang ditandai dengan warna yang sama) harus memiliki tepat satu ratu
- Setiap ratu juga tidak boleh bersentuhan dengan ratu lain secara diagonal

Contoh permainan dapat dilihat di sini:



Gambar 1.1 Permainan Queens edisi 17 Februari 2026
(sumber: https://www.linkedin.com/games/queens)

Tugas ini membahas program yang dapat mencari solusi penempatan Queen dengan menggunakan algoritma *brute force*. Dalam tugas ini, papan selalu dimulai kosong dan jumlah daerah dan warna tepat N buah sesuai ukuran sisi papan.

2 Algoritma Program

Dalam program ini perlu diketahui beberapa variabel global:

- q_number (integer) : banyak queens, untuk validasi selama pembuatan solusi
- nrows (integer) : banyak baris papan
- ncols (integer) : banyak kolom papan
- isLevelTwo (boolean) : adalah level 3 (level menunjukkan cara kerja program)
- isLevelThree (boolean) : adalah level 3
- isLevelFour (boolean) : adalah level 4
- printFile (boolean) : perlu output solusi ke file “output.txt”
- q_pos (array of integer) : posisi queen dalam papan yang dipadatkan jadi 1 dimensi
- inp_file (vector of string) : hasil input file yang diinput

Selanjutnya, program ini menggunakan paradigma prosedural yang ditulis dalam bahasa C++, dan memiliki beberapa fungsi dan prosedur:

a. Fungsi countColours
   Source code:

```
map<char, int> countColours(vector<string> &inp, map<char, int> &clr_map){
    int len_vec = inp.size();

    for (int i = 0; i < len_vec; i++){
        int len_str = inp[i].size();
        for (int j = 0; j < len_str; j++){
            if (clr_map.find(inp[i][j]) == clr_map.end()){ // kalau belum ada di map
                clr_map.insert({inp[i][j], 1});
            }
            else{ // kalau sudah ada di map
                clr_map[inp[i][j]] += 1;
            }
        }
    }
    return clr_map;
}
```

Gambar 2.1 Source Code Fungsi countColours

Langkah-langkah:

- Kode ini membaca input suatu pointer ke array of string &inp yang merupakan pointer ke hasil input file yang diinput dan pointer ke mapping dari char ke integer &clr_map yang boleh kosong dan output suatu mapping dari char ke integer
- Inti fungsi ini adalah membaca setiap warna pada inp dan terakhir output clr_map yang sudah berisi tiap warna dan banyak kotak dengan warna tersebut

b. Fungsi isValidInput
Source code:

```
bool isValidInput(vector<string> s){
    int scols = s[0].size();
    for (int i = 0; i<s.size(); i++){
        if (scols != s[i].size()){
            return false;
        }
    }
    return (s.size() == scols);
}
```

Gambar 2.2 Source Code Fungsi isValidInput

Langkah-langkah:

- Kode ini membaca input suatu array of string s yang merupakan pointer ke hasil input file yang diinput dan mengembalikan suatu boolean
- Inti fungsi ini adalah menghitung banyak kolom tiap baris input dan banyak baris input dan mengembalikan true jika banyak baris sama dengan banyak kolom dan banyak kolom pada tiap baris konsisten/sama

c. Fungsi createBoolArr
Source code:

```
// enumerated: jadi 1 dimensi
vector<bool> createBoolArr(vector<string> &inp){
    int len_vec = inp.size() * inp[0].size();
    return (vector<bool>(len_vec, false));
}
```

Gambar 2.3 Source Code Fungsi createBoolArr

Langkah-langkah:

- Kode ini membaca input suatu pointer ke array of string &inp yang merupakan pointer ke hasil input file yang diinput dan mengembalikan suatu array of boolean.

- Inti fungsi ini adalah melakukan inisialisasi array of boolean dalam 1 dimensi dengan tiap elemen diset sebagai false dan ukuran sebesar banyak baris dikalikan dengan banyak kolom inp

d. Prosedur printBoolArr
Source code:

```
void printBoolArr(vector<bool> &inp){
    for (int i = 0; i< nrows; i++){
        for (int j = 0; j< ncols; j++){
            if (inp[ncols*i + j]){
                cout<<"#";
                if (printFile){
                    out_file << "#";
                }
            }
            else{
                cout << inp_file[i][j];
                if (printFile){
                    out_file << inp_file[i][j];
                }
            }
        }
        cout << endl;
        if (printFile){
            out_file << endl;
        }
    }
}
```

Gambar 2.4 Source Code Prosedur printBoolArr

Langkah-langkah:
- Kode ini membaca input suatu array of boolean &inp
- Inti fungsi ini adalah melakukan output inp ke CLI dalam ukuran yang sesuai.
- Jika printFile diset menjadi true maka akan output ke file juga

e. Fungsi findPos
Source code:

```
vector<int> findPos(vector<bool> &q_bool_array){
    vector<int> pos;
    for (int i = 0; i< q_bool_array.size(); i++){
        if (q_bool_array[i]){
            pos.push_back(i);
        }
    }
    return pos;
}
```

Gambar 2.5 Source Code Fungsi findPos

Langkah-langkah:
- Kode ini membaca input suatu array of boolean &q_bool_array dan mengembalikan array of integer pos yang adalah sejumlah indeks posisi queens
- Inti fungsi ini adalah mengembalikan indeks posisi queens pada papan (q_bool_array) saat ini

f. Fungsi toCoordinates
Source code:

```
tuple<int, int> toCoordinates(int idx){
    // tidak menghandle kasus out of bounds
    return make_tuple((idx/ncols), (idx%ncols));
}
```

Gambar 2.6 Source Code Fungsi toCoordinates

Langkah-langkah:

- Kode ini membaca input suatu integer idx (posisi queen secara 1 dimensi) dan mengembalikan tuple <integer, integer> dengan urutan integer pertama sebagai indeks baris dan kedua sebagai indeks kolom
- Inti fungsi ini adalah mengembalikan indeks posisi queens dalam 2 dimensi berdasarkan banyak kolom dan baris

g. Fungsi toIndex
Source code:

```
int toIndex(int x, int y){
    // tidak menghandle kasus out of bounds
    return (x*ncols+y);
}
```

Gambar 2.7 Source Code Fungsi toIndex

Langkah-langkah:
- Kode ini membaca input dua integer idx x dan y (x sebagai indeks baris dan y sebagai indeks kolom) dan mengembalikan indeks posisi queen dalam 1 dimensi
- Inti fungsi ini adalah mengembalikan indeks posisi queens dalam 1 dimensi berdasarkan banyak kolom dan baris

h. Fungsi isSoloRow
Source code:

```
bool isSoloRow(vector<bool> &q_bool_array, int idx){
    if (idx >= q_bool_array.size()){
        cout << "Index " << idx << " out of bounds! Expected < " << q_bool_array.size() << endl;
        if (printFile){
            out_file  << "Index " << idx << " out of bounds! Expected < " << q_bool_array.size() << endl;
        }
        return false;
    }
    auto coor = toCoordinates(idx);
    int x = get<0> (coor);
    bool found_one = false;
    for (int i = 0; i<ncols; i++){
        if (q_bool_array[x*ncols+i]){
            if (!found_one){
                found_one = true;
                continue;
            }
            else{
                return false;
            }
        }
    }
    if (!found_one){
        cout << "No queen in idx: " << idx << endl;
        if(printFile){
            out_file << "No queen in idx: " << idx << endl;
        }
        return false;
    }
    return true;
}
```

Gambar 2.8 Source Code Fungsi isSoloRow

Langkah-langkah:
- Kode ini membaca input suatu array of boolean &q_bool_array (sebagai posisi papan saat ini) dan integer idx sebagai indeks calon queen yang hendak diperiksa
- Inti fungsi ini adalah menghasilkan true jika row pada indeks idx akan hanya memiliki 1 queen dengan diletakkannya queen pada indeks tersebut, false jika tidak
- Jika printFile diset menjadi true maka akan output ke file juga

i. Fungsi isSoloCol
Source code:

```
bool isSoloCol(vector<bool> &q_bool_array, int idx){
    if (idx >= q_bool_array.size()){
        cout << "Index " << idx << " out of bounds! Expected < " << q_bool_array.size() << endl;
        if(printFile){
            out_file << "Index " << idx << " out of bounds! Expected < " << q_bool_array.size() << endl;
        }
        return false;
    }
    auto coor = toCoordinates(idx);
    int y = get<1> (coor);
    bool found_one = false;
    for (int i = 0; i<ncols; i++){
        if (q_bool_array[i*ncols+y]){
            if (!found_one){
                found_one = true;
                continue;
            }
            else{
                return false;
            }
        }
    }
    if (!found_one){
        cout << "No queen in idx: " << idx << endl;
        if (printFile){
            cout << "No queen in idx: " << idx << endl;
        }
        return false;
    }
    return true;
}
```

Gambar 2.9 Source Code Fungsi isSoloCol

Langkah-langkah:

- Kode ini membaca input suatu array of boolean &q_bool_array (sebagai posisi papan saat ini) dan integer idx sebagai indeks calon queen yang hendak diperiksa
- Inti fungsi ini adalah menghasilkan true jika col pada indeks idx akan hanya memiliki 1 queen dengan diletakkannya queen pada indeks tersebut, false jika tidak
- Jika printFile diset menjadi true maka akan output ke file juga

j. Fungsi isNoDNeighbour
Source code:

```
bool isNoDNeighbour(vector<bool> &q_bool_array, int idx){
    auto coor = toCoordinates(idx);
    int x = get<0> (coor);
    int y = get<1> (coor);
    bool cond1 = false; //x-1, y-1
    bool cond2 = false; //x-1, y+1
    bool cond3 = false; //x+1, y-1
    bool cond4 = false; //x+1, y+1

    if (y != 0){ //bukan ujung kiri
        if (x!=0){ // bukan ujung atas
            cond1 = q_bool_array[toIndex(x-1, y-1)];
        }
        if (x!=nrows-1){ // bukan ujung bawah
            cond3 = q_bool_array[toIndex(x+1, y-1)];
        }
    }

    if (y!= ncols-1){ // bukan ujung kanan
        if (x!=0){ // bukan ujung atas
            cond2 = q_bool_array[toIndex(x-1, y+1)];
        }
        if (x!=nrows-1){ // bukan ujung bawah
            cond4 = q_bool_array[toIndex(x+1, y+1)];
        }
    }
    return (!(cond1 || cond2 || cond3 || cond4));
}
```

Gambar 2.10 Source Code Fungsi isNoDNeighbour

Langkah-langkah:

- Kode ini membaca input suatu array of boolean &q_bool_array (sebagai posisi papan saat ini) dan integer idx sebagai indeks calon queen yang hendak diperiksa
- Inti fungsi ini adalah menghasilkan true jika tidak ada queens pada diagonal yang bersentuhan langsung dengan queen pada indeks idx, false jika ternyata ada

k. Fungsi isSoloColour
Source code:

```
bool isSoloColour(vector<bool> &q_bool_array, vector<string> &inp){
    map<char, bool> clr_map;
    int len_vec = inp.size();

    for (int i = 0; i<q_number; i++){
        auto coor = toCoordinates(q_pos[i]);
        int x = get<0> (coor);
        int y = get<1> (coor);
        if (clr_map.find(inp[x][y]) == clr_map.end()){ // kalau belum ada di map
            clr_map.insert({inp[x][y], true});
        }
        else{ // kalau sudah ada di map
            return false;
        }
    }
    return true;
}
```

Gambar 2.11 Source Code Fungsi isSoloColour

Langkah-langkah:
- Kode ini membaca input suatu array of boolean &q_bool_array (sebagai posisi papan saat ini) dan array of string &inp yang adalah input file awal untuk diambil ukurannya
- Inti fungsi ini adalah menghasilkan true jika tidak ada queens pada area yang sama warnanya dengan area queen pada indeks idx, false jika ternyata ada

l. Fungsi isRuleValid
Source code:

```
bool isRuleValid(vector<bool> &q_bool_array){
    for (int i = 0; i<q_number; i++){
        if (!isSoloRow(q_bool_array, q_pos[i]) ||
            !isSoloCol(q_bool_array, q_pos[i]) ||
            !isNoDNeighbour(q_bool_array, q_pos[i])){
            return false;
        }
    }
    return (isSoloColour(q_bool_array, inp_file));
}
```

Gambar 2.12 Source Code Fungsi isRuleValid

Langkah-langkah:
- Kode ini membaca input suatu array of boolean &q_bool_array (sebagai posisi papan saat ini)
- Inti fungsi ini adalah menghasilkan true jika semua queens memenuhi aturan permainan, false jika ternyata ada yang tidak memenuhi

m. Fungsi nextPoss
Source code:

```cpp
int nextPoss(vector<bool> &q_bool_array, int n_queens, bool isPrint)
{
    int arr_len = q_bool_array.size();

    if (isLevelFour){
        vector<int> col(nrows); // simpan posisi sekarang
        for (int i = 0; i < nrows; i++){
            auto coor = toCoordinates(q_pos[i]);
            col[i] = get<1>(coor);
        }

        int i = nrows - 1; // Mulai dari belakang

        while (i >= 0){
            if (col[i] < ncols - 1){ // kalau bisa gerak ke kanan, ke kanan
                col[i]++;
                break;
            }
            else{ // kalau mentok, balek ke paling kiri
                col[i] = 0;
                i--;
            }
        }

        if (i < 0){
            cout << "Semua hasil berhasil ditunjukkan" << endl;
            if (printFile){
                out_file << "Semua hasil berhasil ditunjukkan" << endl;
            }
            return 0;
        }

        for (int j = 0; j < q_bool_array.size(); j++){ // reset papan
            q_bool_array[j] = false;
        }

        q_pos.clear(); // reset q_pos juga

        for (int r = 0; r < nrows; r++){
            int idx = r * ncols + col[r];
            q_bool_array[idx] = true;
            q_pos.push_back(idx); // masukin kembali queens yang sesuai ke q_pos
        }
    }
    else{
```

Gambar 2.13 Source Code Fungsi nextPoss (1)

```cpp
            // reset posisi setelah pivot
            for (int j = i + 1; j < n_queens; j++) {
                q_pos[j] = q_pos[j - 1] + 1;
            }
            for (int j = 0; j < q_bool_array.size(); j++){
                q_bool_array[j] = false;
            }
            for (int j = 0; j < n_queens; j++) {
                q_bool_array[q_pos[j]] = true;
            }

            if (isLevelTwo) {
                bool allRowColValid = true;
                for (int r = 0; r < n_queens; r++) {
                    allRowColValid = (isSoloRow(q_bool_array, q_pos[r]) && isSoloCol(q_bool_array, q_pos[r]));
                    if (isLevelThree){
                        allRowColValid = allRowColValid && (isNoDNeighbour(q_bool_array, q_pos[r]));
                    }
                    if (!allRowColValid){
                        break;
                    }
                }
                if (!allRowColValid) {
                    continue;
                }
            }
            break;
        }


        bool isValidSolution = isRuleValid(q_bool_array);
        if (isPrint || isValidSolution){
            printBoolArr(q_bool_array);
            if (isValidSolution){
                return -1;
            }
        }
        return 1;
    }
```

Gambar 2.14 Source Code Fungsi nextPoss (2)

Langkah-langkah:

- Kode ini membaca input suatu array of boolean &q_bool_array (sebagai posisi papan saat ini), integer n_queens (yang menyatakan banyak queens), dan bool isPrint (yang menyatakan apakah di iterasi ini, output perlu diprint atau tidak)

- Inti fungsi ini adalah membangkitkan satu kombinasi posisi queens pada papan, lalu cek apakah posisi-posisi queens valid atau tidak. Secara spesifiknya, fokusnya pada loop while sebagai berikut:
  * Cari posisi pivot, yaitu queen paling kanan yang masih bisa bergerak ke kanan. Jika sudah tidak ada, berarti papan telah mencapai posisi terakhirnya. Jika ada, maka posisi pivot akan tersimpan dalam i
  * Geser posisi queen pivot ke kanan sebanyak satu kali, lalu "reset", yaitu seluruh queen di sebelah kanan pivot diletakkan sedemikian rupa sehingga berurutan dengan pivot
  * Jika ternyata program dalam level 1, langsung kembalikan -1 apabila hasil valid, 1 jika tidak. Jika program di level 2, cek aturan row dan kolom terlebih dahulu sebelum dioutput. Jika level 3, juga cek aturan diagonal yang bersentuhan langsung. Apabila tidak memenuhi, maka tidak perlu dioutput
  * Jika isPrint true atau solusi valid, maka hasil akan dioutput.

- Jika menggunakan level 4, maka program akan langsung menempatkan queens pada baris yang berbeda-beda, sehingga mempercepat waktu eksekusi secara eksponensial. Cara proses masih mirip, hanya berbeda pada proses reset, queens dikembalikan ke awal baris, alih alih ke sebelah kanan queens pivot

n. Fungsi putQueens
Source code:

```
int putQueens(vector<bool> &q_bool_array, int n_queens){
    if (isLevelFour){
        q_pos.clear();
        for (int r = 0; r < nrows; r++){
            int idx = r * ncols;
            q_bool_array[idx] = true;
            q_pos.push_back(idx);
        }
    }
    else{
        for (int i = 0; i< n_queens; i++){
            int j = i;
            int k = 0;
            while (j >= ncols){ // lanjut ke baris selanjutnya
                j -= ncols;
                k++;
            }
            for (j; j < nrows; j++){
                if (!q_bool_array[k*ncols+j]){
                    q_bool_array[k*ncols+j] = true;
                    break;
                }
            }
        }
    }
```

Gambar 2.15 Source Code Fungsi putQueens (1)

```
    q_pos = findPos(q_bool_array);
    if (q_pos.size() == 0){
        cout << "No queens found!" << endl;
        if (printFile){
            out_file << "No queens found!" << endl;
        }
        return -1;
    }

    int ctr = 0; //counter banyak posisi baru
    int err = 0;
    bool isModZ = true;
    int Z = pow(10, nrows/2); // Bisa diatur tergantung kebutuhan
    if (!isLevelTwo && nrows > 5){
        if (isLevelFour){
            Z *= 10;
        }
        else{
            Z *= 100;
        }
    }
    if (isLevelThree){
        Z /= 10;
    }
```

Gambar 2.16 Source Code Fungsi putQueens (2)

```
    do{
        isModZ = (ctr%Z == 0);
        err = nextPoss(q_bool_array, n_queens, isModZ);
        if (err == 1 || err == -1){
            ctr++;
            if (isModZ || err == -1){
                cout << "Hasil ke-" << ctr << endl << endl;
                if (printFile){
                    out_file << "Hasil ke-" << ctr << endl << endl;
                }
            }
        }
    }
    while (err == 1);

    q_pos = findPos(q_bool_array);
    if (q_pos.size() == 0){
        cout << "No queens found!" << endl;
        if (printFile){
            out_file << "No queens found!" << endl;
        }
        return -1;
    }

    if (err == -1){
        cout << "Solution found!" << endl;
        if (printFile){
            out_file << "Solution found!" << endl;
        }
    }
    cout << endl;
    if (printFile){
        out_file << endl;
    }
    return 0;
}
```

Gambar 2.17 Source Code Fungsi putQueens(3)

Langkah-langkah:

- Kode ini membaca input suatu array of boolean &q_bool_array (sebagai posisi papan saat ini) dan integer n_queens (yang menyatakan banyak queens)
- Inti fungsi ini adalah membangkitkan semua kombinasi posisi queens pada papan, lalu cek apakah posisi-posisi queens valid atau tidak.
  * Taruh semua queen secara berurutan pada q_bool_array, lalu update hasil ini ke q_pos
  * Inisialisasi variabel ctr (sebagai penghitung banyak kombinasi yang telah dioutput), err (sebagai kode apakah program sudah selesai output semua atau belum), isModZ (yang akan mengubah frekuensi output berdasarkan yang dibutuhkan atau rumus bawaan), dan Z (yang mengikuti rumus tertentu sehingga output tampak rapi)
  * Lakukan loop dengan memanggil fungsi nextPoss hingga hasil fungsi tersebut bukan 1, baik karena ada kesalahan ataupun karena sudah ditemukan solusi
  * Update q_pos dan munculkan pesan pesan yang dibutuhkan

o. Fungsi main
Source code:

```
int main()
{
    ifstream file("input_4x4.txt");
    string s;
    char c;
    int i = 0;
    cout << "Reading input: " << endl;
    while (getline(file, s)) {
        inp_file.push_back(s);
        inp_file[i] = s;
        cout << s << endl;
        i++;
    }
    file.close();

    if (!isValidInput(inp_file)){
        cout << "Error: jumlah kolom tiap baris tidak sama!" << endl;
        return 0;
    }

    int inp_digit;
```

Gambar 2.18 Source Code Fungsi main(1)

```
for (;;) {
    cout << "Pilih level program (1-4; 0 untuk bantuan): ";
    if (cin >> inp_digit) {
        if (inp_digit >= 0 && inp_digit <= 4){
            if (inp_digit == 0){
                cout << "Level 1: Pure brute force kombinasi" << endl;
                cout << "Level 2: Validate row and column before outputs" << endl;
                cout << "Level 3: Validate diagonals before outputs" << endl;
                cout << "Level 4: Brute force permutasi (langsung skip baris yang sama)" << endl;
                continue;
            }
            if (inp_digit >= 2){
                if (inp_digit == 3){
                    isLevelThree = true;
                    cout << "Anda memilih level 3" << endl;
                }
                else if (inp_digit == 4){
                    isLevelFour = true;
                    cout << "Anda memilih level 4" << endl;
                }
                else{
                    isLevelTwo = true;
                    cout << "Anda memilih level 2" << endl;
                }
                break;
            }
            break;
        }
    }
    cout << "Masukkan angka yang valid" << endl;
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
}


cout << endl << "Finding solutions..." << endl << endl;
if (printFile){
    out_file << endl << "Finding solutions..." << endl << endl;
}

nrows = i;
ncols = inp_file[0].size();

map<char, int> colour_map;
colour_map = countColours(inp_file, colour_map);

q_number = colour_map.size();
```

Gambar 2.19 Source Code Fungsi main(2)

```
    auto start = high_resolution_clock::now();
    vector<bool> q_bool_array;
    q_bool_array = createBoolArr(inp_file);

    putQueens(q_bool_array, q_number);

    auto stop = high_resolution_clock::now();
    auto duration = duration_cast<milliseconds>(stop-start);
    cout << "Durasi program: " << duration.count() << "ms" << endl;
    if (printFile){
        out_file << "Durasi program: " << duration.count() << "ms" << endl;
    }
    out_file.close();
    return 0;
}
```

Gambar 2.20 Source Code Fungsi main(3)

Langkah-langkah:

- Kode ini akan membaca file input terlebih dahulu, melakukan output isi papan permainan (belum dengan queens nya), lalu memvalidasinya

- Program akan looping hingga mendapatkan input yang sesuai mengenai level program
- Program akan mencari solusi, dimulai dengan menghitung banyak baris dan kolom, menghitung banyak kotak per warna atau area yang ada, lalu memulai perhitungan waktu untuk algoritma utama yang dilakukan pada fungsi putQueens
- Terakhir, output durasi program dan tutup file output

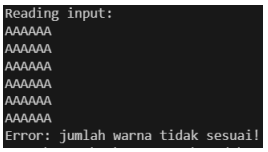
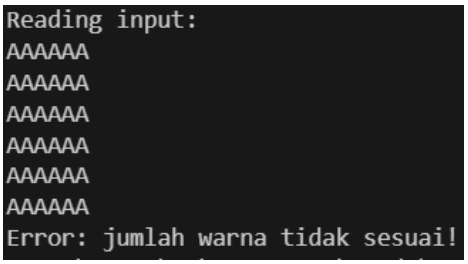
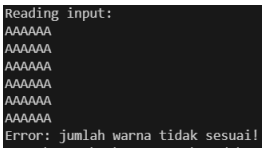
3 Contoh Tangkapan Layar

Pada proses ini digunakan level 1 untuk menunjukkan proses yang paling mencerminkan brute force.

No	Input	Potongan Proses Eksekusi	Hasil akhir

1	input_4x4.txt
	AAAA
	CCBB
	CDDB
	DDDB
	
	`Reading input:`
	`AAAA`
	`CCBB`
	`CDDB`
	`DDDB`
	`Pilih level program (1-4; 0 untuk bantuan): 1`
	
	`Finding solutions...`
		`#AA#`
		`CCB#`
		`CD#B`
		`DDDB`
		`Hasil ke-201`
		
		`#AAA`
		`C#B#`
		`CD#B`
		`DDDB`
		`Hasil ke-301`
		
		`#AAA`
		`CCBB`
		`##DB`
		`#DDB`
		`Hasil ke-401`
		
		`A##A`
		`CCB#`
		`CDDB`
		`#DDB`
		`Hasil ke-501`
		
		`A#AA`
		`##BB`
		`#DDB`
		`DDDB`
		`Hasil ke-601`
		
		`A#AA`
		`CC##`
		`CD#B`
		`DDDB`
		`Hasil ke-701`
			`A#AA`
			`CCB#`
			`#DDB`
			`DD#B`
			`Hasil ke-740`
			
			`Solution found!`
			
			`Durasi program: 26ms`

2	input_7.txt AAAAAAD AABACDD EBBBCDD EEFBCCD EEFBBCC EBBBBBG EEBBGGG Reading input: AAAAAAD AABACDD EBBBCDD EEFBCCD EEFBBCC EBBBBBG EEBBGGG Pilih level program (1-4; 0 untuk bantuan): 1 Finding solutions...	A#AA#AD AABACDD EBBBCDD EEF#C#D EEFBBCC EB##BBG EEBB#GG Hasil ke-15900001 A#AAA## AABAC#D EBBBCDD #EFBCCD EEFB#CC EB#BBBG EEBBGGG Hasil ke-16000001 A#AAA#D #ABAC#D EBBB#DD EEFB#CD EEFBBCC #BBBBBG EEBBGGG Hasil ke-16100001 A#AAA#D A#BACD# EBBBCDD EEF#C#D EEFBBCC EBBBBBG #EBBGGG Hasil ke-16200001	A#AAAD AABA#DD EBBBCD# EE#BCCD #EFBBCC EBB#BBG EEBBG#G Hasil ke-20572184 Solution found! Durasi program: 42355ms
3	input_6.txt AAAAAA AABACD EBBBCD EEFBCC EEFBBC EBBBBB Reading input: AAAAAA AABACD EBBBCD EEFBCC EEFBBC EBBBBB Pilih level program (1-4; 0 untuk bantuan): 1 Finding solutions...	A#AAAA AABACD EBBBCD EEF##C #EFBBC EB#B#B Hasil ke-600001 AA#AAA ##BACD EBBB#D EEFBCC EEFB#C EBB#BB Hasil ke-700001 AA#AAA AABACD #B#BCD E#FBCC EEF#BC EBBBB# Hasil ke-800001	AAA#AA AABAC# E#BBCD EEFB#C EE#BBC #BBBBB Hasil ke-991453 Solution found! Durasi program: 1501ms
4	input_6_r.txt AAAAAA AABACD EBB EEFBCC EEFBBC EBBBB Reading input: AAAAAA AABACD EBB EEFBCC EEFBBC EBBBB Error: jumlah kolom tiap baris tidak sama!	Reading input: AAAAAA AABACD EBB EEFBCC EEFBBC EBBBB Error: jumlah kolom tiap baris tidak sama!	Reading input: AAAAAA AABACD EBB EEFBCC EEFBBC EBBBB Error: jumlah kolom tiap baris tidak sama!

5	input_6_a.txt AAAAAA AAAAAA AAAAAA AAAAAA AAAAAA AAAAAA 		

4 Checklist

No	Poin	Ya	Tidak
1	Program berhasil dikompilasi tanpa kesalahan	v	
2	Program berhasil dijalankan	v	
3	Solusi yang diberikan program benar dan mematuhi aturan permainan	v	
4	Program dapat memasukkan berkas .txt serta menyimpan solusi dalam berkas .txt	v	
5	Program memiliki Graphical User Interface (GUI)		v
6	Program dapat menyimpan solusi dalam bentuk file gambar		v

5 Pernyataan Tidak Melakukan Kecurangan

Tugas ini disusun sepenuhnya tanpa bantuan kecerdasan buatan (Generative AI), melainkan hasil pemikiran dan analisis mandiri.

Ray Owen Martin

6 Lampiran

Repository dapat diakses pada: https://github.com/Tensai-033/Tucil1_13524033