Laporan Tugas Kecil Strategi Algoritma 1

Nama: Faris Wirakusuma Triawan
NIM:13524130

1. Algoritma Brute Force yang digunakan

```
bool find_solution(Shape_Arr allshape, int_arr& allpos, vector<int_arr>&
allpermute, int& jumlah_kasus, int_arr& valid_array, wxStaticText*
output_label, vector<string> initboard, QueenFrame* frame){
   int n_queens = (int)allshape.size();
   allpos.assign(n_queens, 0);
   allpermute.clear();
   jumlah_kasus = 0;
   auto update_ui_visual = [&](const int_arr& current_pos) {
       update_position(output_label, current_pos, initboard,
jumlah_kasus);
       vector<string> temp_board = output_str(current_pos, initboard);
       renderToImage(temp_board, initboard, "temp_view.png");

       wxImage img(wxString::FromUTF8("test/temp_view.png"),
wxBITMAP_TYPE_PNG);
       if (img.IsOk() && frame && frame->m_bitmapDisplay) {
           frame->m_bitmapDisplay->SetBitmap(wxBitmap(img));
           frame->Update();
           wxYield();
       }
   };
   for (int b1=0; b1<9; b1++) {
       allpos[0] = b1;
```

```
        jumlah_kasus++;
        if(jumlah_kasus % VIEW_SPEED == 0) update_position(output_label,
allpos, initboard, jumlah_kasus);
        if (n_queens == 1) {
            if (cek_setiap_shape(allpos, allshape,1)) { valid_array =
allpos; return true; }
            continue;
        }

        for (int b2=0; b2<9; b2++) {
            allpos[1] = b2;
            jumlah_kasus++;
            if(jumlah_kasus % VIEW_SPEED == 0)
update_position(output_label, allpos, initboard, jumlah_kasus);
            if (b2 == b1) continue;
            if (n_queens == 2) {
                if (diagonal_valid(allpos, 2) && cek_setiap_shape(allpos,
allshape, 2)) { valid_array = allpos; return true; }
                continue;
            }

            for (int b3=0; b3<9; b3++) {
                allpos[2] = b3;
                jumlah_kasus++;
                if(jumlah_kasus % VIEW_SPEED == 0)
update_position(output_label, allpos, initboard, jumlah_kasus);
                if (b3 == b1 || b3 == b2) continue;
                if (n_queens == 3) {
                    if (diagonal_valid(allpos, 3) &&
cek_setiap_shape(allpos, allshape, 3)) { valid_array = allpos; return
true; }
                    continue;
                }

                for (int b4=0; b4<9; b4++) {
                    allpos[3] = b4;
                    jumlah_kasus++;
                    if(jumlah_kasus % VIEW_SPEED == 0)
update_position(output_label, allpos, initboard, jumlah_kasus);
```

```
                    if (b4 == b1 || b4 == b2 || b4 == b3) continue;
                    if (n_queens == 4) {
                        if (diagonal_valid(allpos, 4) &&
cek_setiap_shape(allpos, allshape, 4)) { valid_array = allpos; return
true; }
                        continue;
                    }

                    for (int b5=0; b5<9; b5++) {
                        allpos[4] = b5;
                        jumlah_kasus++;
                        if(jumlah_kasus % VIEW_SPEED == 0)
update_position(output_label, allpos, initboard, jumlah_kasus);
                        if (b5 == b1 || b5 == b2 || b5 == b3 || b5 == b4)
continue;
                        if (n_queens == 5) {
                            if (diagonal_valid(allpos, 5) &&
cek_setiap_shape(allpos, allshape, 5)) { valid_array = allpos; return
true; }
                            continue;
                        }

                        for (int b6=0; b6<9; b6++) {
                            allpos[5] = b6;
                            jumlah_kasus++;
                            if(jumlah_kasus % VIEW_SPEED == 0)
update_position(output_label, allpos, initboard, jumlah_kasus);
                            if (b6 == b1 || b6 == b2 || b6 == b3 || b6 ==
b4 || b6 == b5) continue;
                            if (n_queens == 6) {
                                if (diagonal_valid(allpos, 6) &&
cek_setiap_shape(allpos, allshape, 6)) { valid_array = allpos; return
true; }
                                continue;
                            }

                            for (int b7=0; b7<9; b7++) {
                                allpos[6] = b7;
                                jumlah_kasus++;
```

```
                                if(jumlah_kasus % VIEW_SPEED == 0)
update_position(output_label, allpos, initboard, jumlah_kasus);
                                if (b7 == b1 || b7 == b2 || b7 == b3 ||
b7 == b4 || b7 == b5 || b7 == b6) continue;
                                if (n_queens == 7) {
                                    if (diagonal_valid(allpos, 7) &&
cek_setiap_shape(allpos, allshape, 7)) { valid_array = allpos; return
true; }
                                    continue;
                                }

                                for (int b8=0; b8<9; b8++) {
                                    allpos[7] = b8;
                                    jumlah_kasus++;
                                    if(jumlah_kasus % VIEW_SPEED == 0)
update_position(output_label, allpos, initboard, jumlah_kasus);
                                    if (b8 == b1 || b8 == b2 || b8 == b3
|| b8 == b4 || b8 == b5 || b8 == b6 || b8 == b7) continue;
                                    if (n_queens == 8) {
                                        if (diagonal_valid(allpos, 8) &&
cek_setiap_shape(allpos, allshape, 8)) { valid_array = allpos; return
true; }
                                        continue;
                                    }

                                    for (int b9=0; b9<9; b9++) {
                                        allpos[8] = b9;
                                        jumlah_kasus++;
                                        if(jumlah_kasus % VIEW_SPEED ==
0) update_position(output_label, allpos, initboard, jumlah_kasus);
                                        if (b9 == b1 || b9 == b2 || b9 ==
b3 || b9 == b4 || b9 == b5 || b9 == b6 || b9 == b7 || b9 == b8)
continue;
                                        if (n_queens == 9) {
                                            if (diagonal_valid(allpos, 9)
&& cek_setiap_shape(allpos, allshape, 9)) { valid_array = allpos; return
true; }
                                            continue;
                                        }
```

```
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
    }
    return false;
}
```

PENJELASAN:

Penjelasan parameter pada fungsi find_solution:

```
Bool      find_solution(Shape_Arr      allshape,      int_arr&      allpos,
std::vector<int_arr>&    allpermute,    int&    jumlah_kasus,    int_arr&
valid_array,    wxStaticText*    output_label,    std::vector<std::string>
initboard, QueenFrame* frame);
```

Shape_Arr : Merupakan sebuah array yang berisikan shape yang didefine sehingga di call nya Shape_Arr ,shape sendiri merupakan adt sederhana yang saya buat berikut adalah ADT shapenya:

```
struct Shape {
    char key;
```

```
    vector<kord> point;
    bool isvalid = false;
};


#define Shape_Arr vector<Shape>
```

Di dalam ADT tersebut ada atribut key,point,dan juga isvalid

1. Key = simbol shape yang dimana key didapatkan bedasarkan input dan dikasih tanda yaitu key agar mudah di bedakan dengan bentuk lainnya
2. Point = merupakan array yang berisikan adt kord yang berisikan integer x,y yang fungsi nya adalah memapping kordinat paternnya sebagai contoh ada pattern AAA berarti kita assign menjadi (0,0)(1,0)(2,0) berikut ADT pointnya:

```
struct kord{
    int x;
    int y;
};
```

3. Isvalid = is valid adalah boolean yang berfungsi untuk sebagai tanda jika semisal queen sudah ada disitu

Allpos: Adalah posisi queennya didalam sebuah int array contoh (1,2,..,9)
Titik x dalam allpos direpresentasikan di dalam x= allpos[idx] dan y = idx,

Penjelasan Algoritma find_solution nya:

Pertama Tama kita assign dulu allpos nya (n_queen,0) n_queen adalah jumlah queen yang akan muncul dan n_queen = jumlah_shape ,lalu kita melakukan pengecekan secara full brute force di cek di setiap barisnya jadi kita cek baris1,baris2,..,baris9 yang disimbolkan b1,b2,.. Agar queen tidak saling menyerang dan juga tidak bersebelahan sama sekali maka diperlukan pengecekan diagonal dan dan pengecekan validasi shape berikut

1. Pengecekan setiap shape

```
bool cek_setiap_shape(const int_arr& pos, Shape_Arr& shapes, int n) {
    for (int i=0; i < (int)shapes.size(); i++) shapes[i].isvalid = false;
    for (int y = 0; y < n; y++) {
        int x = pos[y];
        for (int i = 0; i < (int)shapes.size(); i++) {
            for (const auto& p : shapes[i].point) {
                if (p.x == x && p.y == y) {
                    if (shapes[i].isvalid) return false;
                    shapes[i].isvalid = true;
                    break;
                }
            }
        }
    }
    for (const auto& s : shapes) if (!s.isvalid) return false;
    return true;
}
```

Di dalam fungsi ini kita melakukan pengecekan apakah di setiap shape sudah di occupied jika sudah ada maka false jika belum true

2. Pengecekan diagonal

```
bool diagonal_valid(const int_arr &checked_arr, int n){
    for (int i=0; i < n; i++) {
        for (int j = i+1; j < n; j++) {
            if (abs(checked_arr[i]-checked_arr[j]) == abs(i-j)) {
                return false;
            }
        }
    }
    return true;
}
```

Di dalam fungsi ini nantinya akan di cek apakah queennya saling menyerang/bersebelahan jika iya maka diagonal invalid jika tidak maka valid

```
for (int b1=0; b1<9; b1++) {
       allpos[0] = b1;
       jumlah_kasus++;
       if(jumlah_kasus % VIEW_SPEED == 0) update_position(output_label,
allpos, initboard, jumlah_kasus);
       if (n_queens == 1) {
           if (cek_setiap_shape(allpos, allshape,1)) { valid_array =
allpos; return true; }
           continue;
       }

       for (int b2=0; b2<9; b2++) {
           allpos[1] = b2;
           jumlah_kasus++;
           if(jumlah_kasus % VIEW_SPEED == 0)
update_position(output_label, allpos, initboard, jumlah_kasus);
           if (b2 == b1) continue;
           if (n_queens == 2) {
               if (diagonal_valid(allpos, 2) && cek_setiap_shape(allpos,
allshape, 2)) { valid_array = allpos; return true; }
               continue;
           } # DAN SETERUSNYA HINGGA baris ke 9
```

Berikut merupakan proses pengecekan tiap barisnya di setiap baris kita assign allpos[idx] = baris_ke_idx ,menambah jumlah kasus serta update posisi UI nya secara live dan untuk mengurangi waktu kompile maka kita ringkas waktu updatenya menggunakan view_speed diisini saya set menjadi 729

Menggunakan:

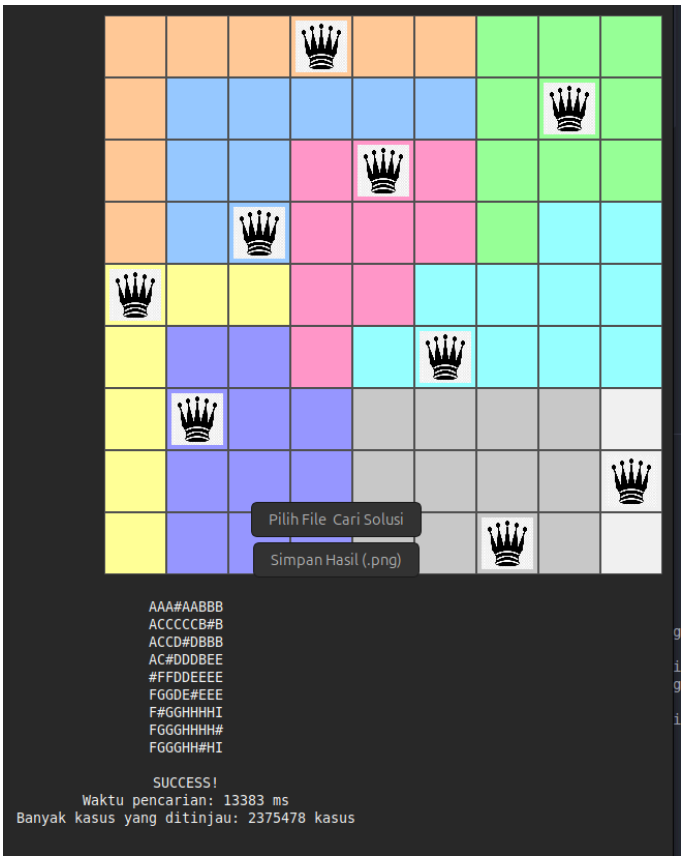
```
#define VIEW_SPEED 729
```

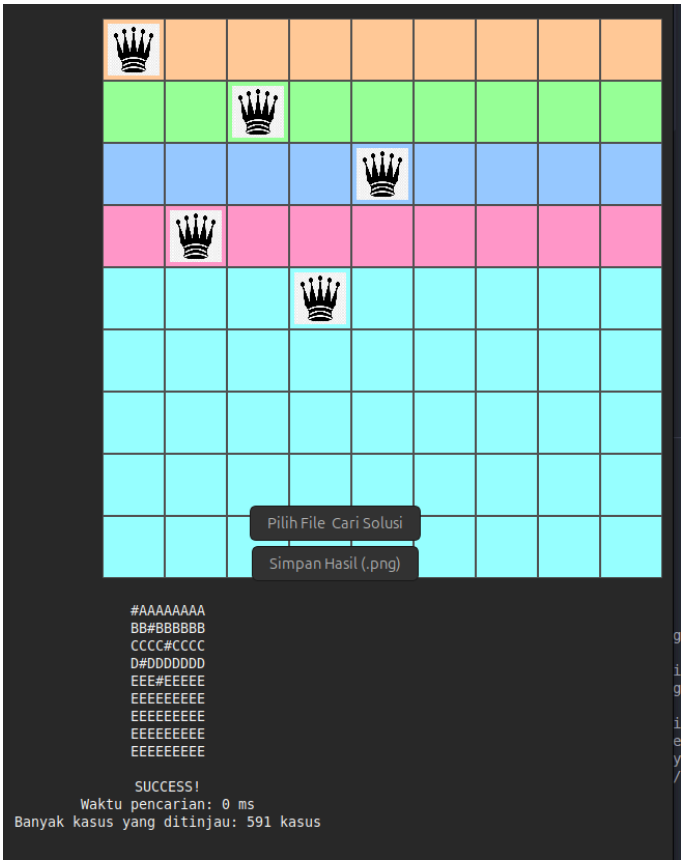
Sehingga akan mengupdate setiap kelipatan 729 agar lebih cepat,untuk setiap b1=b2,b1=b2=b3,dst kita skip karena sudah pasti salah dan tidak memenuhi sharat posisi queen karena pasti sejajar sehingga harus kita continue tetapi tetap terhitung ke dalam 1x pengecekan kasus.terdapat kondisi ketika n_queen=n dan jika iya maka akan di cek diagonal dan shapenya sesuai aturannya dan akan di skip(continue). Untuk setiap baris akan dilakukan proses yang sama sampai petak/baris ke 9 dan jika tidak ditemukan solusi maka akan tertulis “tidak ada solusi pada ui” tetapi jika ada solusi akan tertampil pada UI dan dapat disimpan sebagai PNG file.

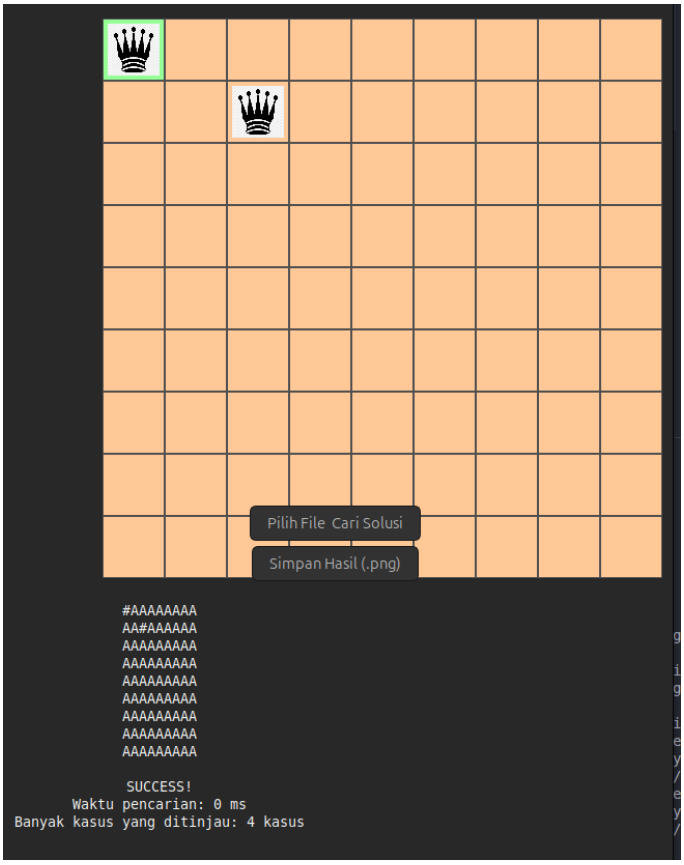
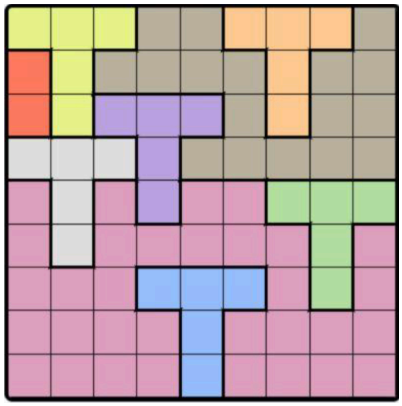
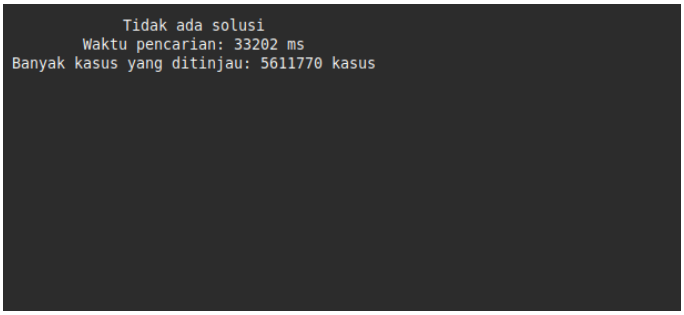
No	Poin	Ya	Tidak
1	Program berhasil di kompilasi tanpa kesalahan	✓	
2	Program berhasil di jalankan	✓	
3	Solusi yang diberikan program benar danmematuhi aturan permainan	✓	
4	Program dapat membaca masukan berkas .txt serta menyimpan solusi dalam berkas .txt	✓ (hanya input)	
5	Program memiliki Graphical User Interface(GUI)	✓	
6	Program dapat menyimpan solusi dalam bentukfile gambar	✓	

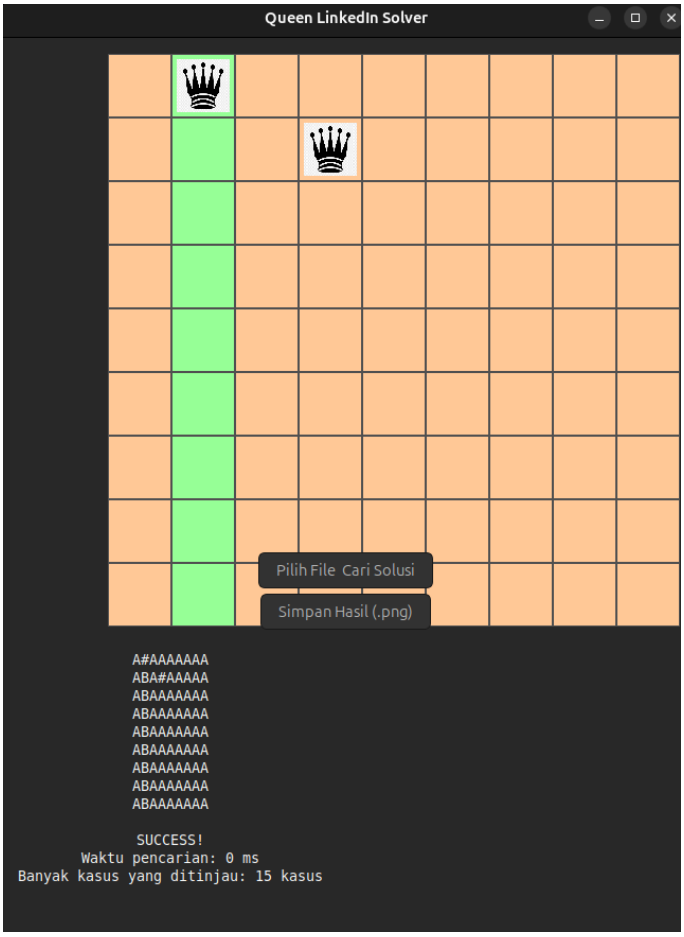
2. Contoh Input dan Output

no	input	output

1	AAAAAABBB ACCCCCBBB ACCDDDBBB ACCDDDBEE FFFDDEEEE FGGDEEEEE FGGGHHHHI FGGGHHHHI FGGGHHHHI	

2	AAAAAAAAA BBBBBBBBB CCCCCCCCC DDDDDDDDD EEEEEEEEE EEEEEEEEE EEEEEEEEE EEEEEEEEE EEEEEEEEE	

3	BAAAAAAAA AAAAAAAAA AAAAAAAAA AAAAAAAAA AAAAAAAAA AAAAAAAAA AAAAAAAAA AAAAAAAAA AAAAAAAAA	
4		

5	ABAAAAAAA ABAAAAAAA ABAAAAAAA ABAAAAAAA ABAAAAAAA ABAAAAAAA ABAAAAAAA ABAAAAAAA ABAAAAAAA	

3. Link Repo:https://github.com/fariswirakusuma/Tucil1_13524130

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

[Tanda tangan mahasiswa]

Faris Wirakusuma,13524130