Laporan Tugas Kecil 1 Strategi Algoritma

Penyelesaian Puzzle Queen dengan Algoritma Brute Force

Nama : Kloce Paul William Saragih
NIM : 1352404
Kelas : K2

1. Algoritma Brute Force yang Digunakan

A. Algoritma brute force tanpa pruning warna (Solver 1)

Algoritma ini mencoba menempatkan Ratu berdasarkan batasan posis yang sesuai dengan batasan pada permasalahan yakni, tidak boleh ada pasangan ratu yang terletak satu kolom maupun baris dan tidak boleh bersentuhan, namun pengecekan validitas warna baru dilakukan setelah seluruh papan terisi penuh.

Langkah-langkah algoritma Solver 1:

- Mulai dari baris awal (paling atas)
- Iterasi setiap kolom dari kiri ke kanan pada baris tersebut.
- Cek apakah penempatan Ratu di kolom saat ini aman secara lokasi (tidak satu kolom dengan Ratu di baris sebelumnya, dan tidak bersentuhan secara diagonal/samping dengan Ratu di baris tepat di atasnya).
- Jika aman secara lokasi, tempatkan Ratu sementara, lalu lanjut rekursif ke baris berikutnya (baris + 1).
- Jika baris sudah mencapai N (papan penuh), kita akan melakukan pengecekan warna. Iterasi seluruh Ratu yang telah ditempatkan dan pastikan tidak ada warna yang sudah di occupy oleh queen lainya. Jika ternyata kondisi tidak terpenuhi maka kita akan mencoba konfigurasi lainnya.
- Jika solusi ditemukan, hentikan kalkulasi dan catat jawaban yang ditemukan.

B. Algoritma brute force dengan pruning (Solver 2)

Pendekatan ini lebih efisien karena melakukan pemangkasan (*pruning*) cabang pencarian segera setelah ditemukan pelanggaran aturan warna, tanpa menunggu papan penuh.

Langkah-langkah algoritma Solver 2:

- Siapkan map/array untuk mencatat warna yang sudah terpakai (set awalnya false).
- Mulai dari baris ke paling atas.
- Iterasi setiap kolom dari kiri ke kanan
- Cek apakah penempatan Ratu di kolom saat ini aman secara lokasi (tidak satu kolom dengan Ratu di baris sebelumnya, dan tidak bersentuhan secara diagonal/samping dengan Ratu di baris tepat di atasnya) **dan aman secara pengisian warna**..

- Jika aman secara lokasi dan **warna**, tempatkan Ratu sementara, lalu lanjut rekursif ke baris berikutnya (baris + 1).
- Jika baris sudah tidak ada lagi peletakan yang memungkinkan pada suatu baris, maka kita akan lanjut dengan konfigurasi peletakan lainnya tanpa harus mengecek sampai baris akhir (pruning).
- Jika baris sudah mencapai N (papan penuh), kita akan melakukan pengecekan warna. Iterasi seluruh Ratu yang telah ditempatkan dan pastikan tidak ada warna yang sudah di occupy oleh queen lainya. Jika ternyata kondisi tidak terpenuhi maka kita akan mencoba konfigurasi lainnya.
- Jika solusi ditemukan, hentikan kalkulasi dan catat jawaban yang ditemukan.

Catatan : Program menggunakan backtrack yang (sejauh saya baca pada QNA) bukanlah brute force murni. Menurut saya permasalahan ini memerlukan rekursi sehingga penggunaan backtrack tidak dapat dibatasi, ditambah lagi backtrack membuat simulasi lebih intuitif.

Catatan 2 : Untuk load image as input, image yang diberikan harus merupakan image dengan extension .jpeg , .jpg, maupun .png. Selain itu, input akan ditolak apabila image tidak membentuk image berukuran persegi (jumlah pixel width dan height yang sama). Kualitas gambar mungkin akan mempengaruhi hasil konversi gambar ke input pada program. Jumlah baris dan kolom juga harus dimasukkan secara eksplisit pada entry GUI karena program akan melakukan pengambilan warna dengan membagi image menjadi beberapa cell dan mengambil warna pada titik tengah tiap cell yang bergantung pada jumlah kolom dan baris.

2. Source Program

Program dibuat menggunakan bahasa pemrograman Go (Golang) dengan library GUI Fyne. Seluruh source code terletak pada folder `/src` . Berikut adalah implementasi kode program:

`Globals.go` : Berisi variabel ataupun data global

```
package main

import (
    "image/color"

    "fyne.io/fyne/v2"
    "fyne.io/fyne/v2/canvas"
)

var (
    n     int
    g     []string
    ans   []int
    found bool
    cnt   int
    stop  bool
```

```
    delay int = 50

    grid   *fyne.Container
    rects  [][]*canvas.Rectangle
    texts  [][]*canvas.Image
    uiChan chan bool
)

var Colors = []color.Color{
    color.RGBA{220, 20, 60, 255},
    color.RGBA{65, 105, 225, 255},
    color.RGBA{34, 139, 34, 255},
    color.RGBA{255, 215, 0, 255},
    color.RGBA{138, 43, 226, 255},
    color.RGBA{255, 140, 0, 255},
    color.RGBA{0, 128, 128, 255},
    color.RGBA{255, 105, 180, 255},
    color.RGBA{112, 128, 144, 255},
    color.RGBA{139, 69, 19, 255},
    color.RGBA{0, 191, 255, 255},
    color.RGBA{154, 205, 50, 255},
    color.RGBA{210, 105, 30, 255},
    color.RGBA{75, 0, 130, 255},
    color.RGBA{255, 127, 80, 255},
    color.RGBA{70, 130, 180, 255},
    color.RGBA{128, 128, 0, 255},
    color.RGBA{218, 112, 214, 255},
    color.RGBA{178, 34, 34, 255},
    color.RGBA{95, 158, 160, 255},
    color.RGBA{189, 183, 107, 255},
    color.RGBA{147, 112, 219, 255},
    color.RGBA{46, 139, 87, 255},
    color.RGBA{240, 128, 128, 255},
    color.RGBA{100, 149, 237, 255},
    color.RGBA{255, 165, 0, 255},
} // credits to : https://www.w3schools.com/cssref/css_colors.php
```

`solver.go` : Berisi algoritma brute force yang digunakan (yakni Solver1 dan Solver2)

```
package main
```

```
import (
   "time"
)

func bad(r, c int) bool {
   for i := 0; i < r; i++ {
       pc := ans[i]
       if pc == c {
           return true
       }

       if i == r-1 {
           if pc == c-1 || pc == c+1 {
               return true
           }
       }
   }
   return false
}

// no prune
func solve1(r int, live bool) {
   if found || stop {
       return
   }

   if live {
       uiChan <- true
       time.Sleep(time.Duration(delay) * time.Millisecond)
   }

   cnt++

   if r == n {
       mask := make(map[uint8]bool)
       ok := true
       for i := 0; i < n; i++ {
           c := g[i][ans[i]]
           if mask[c] {
               ok = false
```

```
                break
            }
            mask[c] = true
        }
        if ok {
            found = true
        }
        return
    }

    for c := 0; c < n; c++ {
        if !bad(r, c) {
            ans[r] = c
            solve1(r+1, live)
            if found {
                return
            }
            ans[r] = -1
        }
    }
}

// with pruning
func solve2(r int, live bool, used map[uint8]bool) {
    if found || stop {
        return
    }
    cnt++

    if live {
        uiChan <- true
        time.Sleep(time.Duration(delay) * time.Millisecond)
    }

    if r == n {
        found = true
        return
    }

    for c := 0; c < n; c++ {
```

```
        colour := g[r][c]
        if !bad(r, c) && !used[colour] {
            ans[r] = c
            used[colour] = true

            solve2(r+1, live, used)
            if found {
                return
            }

            used[colour] = false
            ans[r] = -1
        }
    }
}
```

main.go : Berisi kode tampilan UI dan interface input/output file maupun teks

```
package main

import (
    "bufio"
    "bytes"
    "fmt"
    "image"
    "image/draw"
    "image/png"
    "os"
    "path/filepath"
    "strconv"
    "time"

    "fyne.io/fyne/v2"
    "fyne.io/fyne/v2/app"
    "fyne.io/fyne/v2/canvas"
    "fyne.io/fyne/v2/container"
    "fyne.io/fyne/v2/dialog"
    "fyne.io/fyne/v2/layout"
    "fyne.io/fyne/v2/storage"
```

```
    "fyne.io/fyne/v2/widget"
)

// Labels
var lblTime, lblIter, lblStatus *widget.Label

var queen fyne.Resource

func valid_grid() bool {
    if n <= 0 {
        return false
    }
    if len(g) != n {
        return false
    }

    unique := make(map[byte]bool)

    for r := 0; r < n; r++ {
        if len(g[r]) != n {
            return false
        }

        for c := 0; c < n; c++ {
            str := g[r][c]

            if str < byte('A') || str > byte('Z') {
                return false
            }
            unique[str] = true
        }
    }

    if len(unique) != n {
        fmt.Println("DEBUG : FALSE")
        fmt.Println(len(unique))
        return false
    }

    return true
```

```
}

func read_file(path string) bool {
   f, err := os.Open(path)
   if err != nil {
       return false
   }
   defer f.Close()

   sc := bufio.NewScanner(f)
   g = []string{}
   for sc.Scan() {
       cnt += 1
       line := sc.Text()
       if len(line) > 0 {
           g = append(g, line)
       }
   }
   n = len(g)

   if !valid_grid() {
       lblStatus.SetText("Status: Invalid input! make sure the number of
row, columns and colour are the same")

       g = []string{}
       n = 0
       found = false
       cnt = 0
       build_grid()
       refresh_grid()

       return false
   }

   ans = make([]int, n)
   for i := range ans {
       ans[i] = -1
   }

   lblTime = widget.NewLabel("Time: -")
```

```
    lblIter = widget.NewLabel("Iterations: -")

    return true
}

func refresh_grid() {
    if n == 0 {
        return
    }
    for r := 0; r < n; r++ {
        for c := 0; c < n; c++ {
            if ans[r] == c {
                texts[r][c].Show()
            } else {
                texts[r][c].Hide()
            }
            texts[r][c].Refresh()
        }
    }
}

func run_solver(mode int, live bool, w *fyne.Window) {
    if n == 0 {
        dialog.ShowError(fmt.Errorf("Load file first"), *w)
        return
    }

    // reset
    found = false
    cnt = 0
    stop = false
    ans = make([]int, n)
    for i := range ans {
        ans[i] = -1
    }

    lblIter.SetText("Iterations: -")
    lblTime.SetText("Time: -")
    lblStatus.SetText("Status: Running...")
    start := time.Now()
```

```
    go func() {
        if mode == 1 {
            solve1(0, live)
        } else {
            mask := make(map[byte]bool)
            solve2(0, live, mask)
        }

        dur := time.Since(start)

        refresh_grid()
        if found {
            lblStatus.SetText("Status: found a solution!")
        } else {
            lblStatus.SetText("Status: No solution found :(")
        }
        lblTime.SetText(fmt.Sprintf("Time: %d ms", dur.Milliseconds()))
        lblIter.SetText(fmt.Sprintf("Iterations: %d", cnt))
    }()

}

func build_grid() {
    rects = make([][]*canvas.Rectangle, n)
    texts = make([][]*canvas.Image, n)
    objects := []fyne.CanvasObject{}

    for r := 0; r < n; r++ {
        rects[r] = make([]*canvas.Rectangle, n)
        texts[r] = make([]*canvas.Image, n)
        for c := 0; c < n; c++ {
            rect := canvas.NewRectangle(Colors[g[r][c]-'A'])
            rect.StrokeColor = Colors[g[r][c]-'A']

            img := canvas.NewImageFromResource(queen)
            img.FillMode = canvas.ImageFillContain
            img.Hide()

            rects[r][c] = rect
            texts[r][c] = img
```

```
            stack := container.NewStack(rect, img)
            objects = append(objects, stack)
        }
    }
    grid.Layout = layout.NewGridLayout(n)
    grid.Objects = objects
    grid.Refresh()
}

func save_to_image(filename string) {
    if n == 0 {
        return
    }

    sz := 130
    var q image.Image

    decoded, err := png.Decode(bytes.NewReader(queen.Content()))
    if err == nil {
        q = decoded
        sz = q.Bounds().Dx()
    }

    width := n * sz
    height := n * sz
    ul := image.Point{0, 0}
    lr := image.Point{width, height}

    img := image.NewRGBA(image.Rectangle{ul, lr})

    for r := 0; r < n; r++ {
        for c := 0; c < n; c++ {
            x := c * sz
            y := r * sz

            color := Colors[g[r][c]-'A']
            outline := image.Rect(x, y, x+sz, y+sz)
            draw.Draw(img, outline, image.Black, image.Point{0, 0},
draw.Src)
```

```
            bgRect := image.Rect(x+3, y+3, x+sz-3, y+sz-3)
            draw.Draw(img, bgRect, &image.Uniform{color}, image.Point{0,
0}, draw.Over)

            if ans[r] == c {
                offset := image.Point{
                    X: (sz - q.Bounds().Dx()) / 2,
                    Y: (sz - q.Bounds().Dy()) / 2,
                }
                targetRect := image.Rect(x+offset.X, y+offset.Y,
x+offset.X+q.Bounds().Dx(), y+offset.Y+q.Bounds().Dy())
                draw.Draw(img, targetRect, q, q.Bounds().Min, draw.Over)
            }
        }
    }

    filename += ".png"

    outputPath := filepath.Join("../test", filename)

    f, err := os.Create(outputPath)
    if err != nil {
        lblStatus.SetText("Status: failed to save image.")
        return
    }
    defer f.Close()

    err2 := png.Encode(f, img)
    if err2 != nil {
        fmt.Println(err2)
    }
}

func save_to_txt(filename string) {
    if n == 0 {
        return
    }

    filename += ".txt"
```

```
    outputPath := filepath.Join("../test", filename)

    f, err := os.Create(outputPath)
    if err != nil {
        lblStatus.SetText("Status: failed to save txt.")
        return
    }
    defer f.Close()

    for r := 0; r < n; r++ {
        var line string
        for c := 0; c < n; c++ {
            if ans[r] == c {
                line += "#"
            } else {
                line += string(g[r][c])
            }
        }

        if _, err := f.WriteString(line + "\n"); err != nil {
            lblStatus.SetText("Status: failed to save txt.")
            return
        }
    }

}

type MappedColor struct {
    R, G, B uint32
    Char    byte
}

func map_image_input(path string, sz int) bool {
    f, err := os.Open(path)
    if err != nil {
        return false
    }
    defer f.Close()

    img, _, err := image.Decode(f)
```

```
    if err != nil {
        return false
    }

    bounds := img.Bounds()
    width := bounds.Dx()
    height := bounds.Dy()

    if width != height {
        lblStatus.SetText("Status: input image isn't valid")
    }

    step := width / sz

    n = sz
    g = make([]string, n)
    mp := []MappedColor{}
    cur := byte('A')

    for r := 0; r < n; r++ {
        rowStr := ""
        for c := 0; c < n; c++ {
            cx := (c * step) + step/2
            cy := (r * step) + step/2

            r, g, b, _ := img.At(cx, cy).RGBA()
            curColor := MappedColor{R: r, G: g, B: b}

            ch := byte(0)
            for _, k := range mp {
                if curColor.R == k.R && curColor.G == k.G && curColor.B
== k.B {
                    ch = k.Char
                    break
                }
            }

            if ch == 0 {
                ch = cur
                curColor.Char = ch
```

```
                mp = append(mp, curColor)
                cur++
            }

            rowStr += string(ch)
        }
        g[r] = rowStr
    }

    for i := range mp {
        fmt.Println(mp[i].R, mp[i].G, mp[i].B)
    }
    ans = make([]int, n)
    for i := range ans {
        ans[i] = -1
    }

    return true
}

func main() {
    fmt.Println("working")
    n = 0
    found = false
    queen, _ = fyne.LoadResourceFromPath("../assets/queen.png")

    a := app.New()
    w := a.NewWindow("Tucil 1")
    w.Resize(fyne.NewSize(1200, 600))

    grid = container.New(layout.NewGridLayout(1))
    lblTime = widget.NewLabel("Time: -")
    lblIter = widget.NewLabel("Iterations: -")
    lblStatus = widget.NewLabel("Status: Waiting for input")

    btn1 := widget.NewButton("Solver 1 (Live without pruning)", func() {
run_solver(1, true, &w) })
    btn2 := widget.NewButton("Solver 2 (Live with Pruning)", func() {
run_solver(2, true, &w) })
```

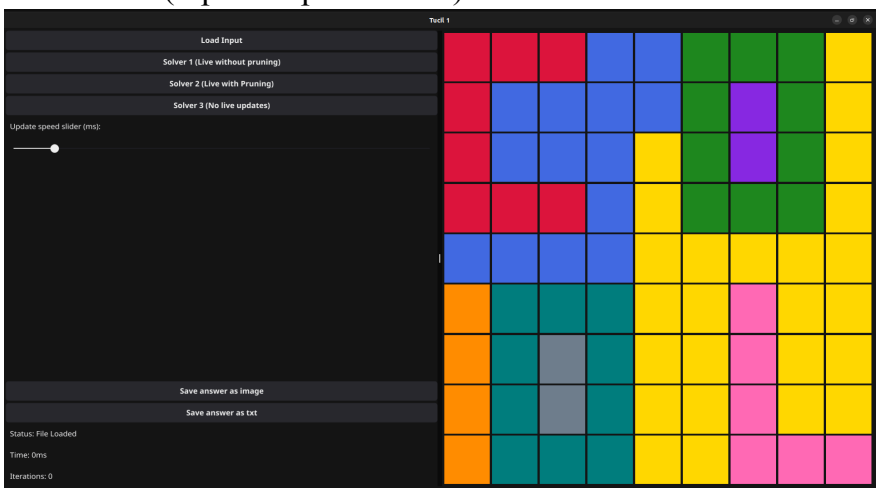
```
    btn3 := widget.NewButton("Solver 3 (No live updates)", func() {
run_solver(1, false, &w) })
    btnLoad := widget.NewButton("Load Input", func() {
        dialog.ShowFileOpen(func(r fyne.URIReadCloser, err error) {
            if r == nil {
                return
            }

            if read_file(r.URI().Path()) {
                build_grid()
                lblStatus.SetText("Status: File Loaded")
            }
        }, w)
    })
    btnSaveImg := widget.NewButton("Save answer as image", func() {
        if !found {
            lblStatus.SetText("Status: Unable to save solution, no
solution found!")
            return
        }
        save_to_image("sol-img-" + time.Now().Format("20060102150405"))
        lblStatus.SetText("Status: Solution saved as image!")
    })
    btnSaveTxt := widget.NewButton("Save answer as txt", func() {
        if !found {
            lblStatus.SetText("Status: Unable to save solution, no
solution found!")
            return
        }
        save_to_txt("sol-txt-" + time.Now().Format("20060102150405"))
        lblStatus.SetText("Status: Solution saved as text!")
    })

    entryN := widget.NewEntry()
    entryN.SetPlaceHolder("Enter the number of row or column")
    btnLoadImg := widget.NewButton("Load Image Input", func() {
        inputSize, err := strconv.Atoi(entryN.Text)
        if err != nil || inputSize <= 0 {
            dialog.ShowError(fmt.Errorf("Please enter a valid number for
N first"), w)
```

```
            return
        }

        fd := dialog.NewFileOpen(func(reader fyne.URIReadCloser, err
error) {
            if err != nil || reader == nil {
                return
            }
            defer reader.Close()

            cek := map_image_input(reader.URI().Path(), inputSize)
            if cek {
                lblStatus.SetText("Status: Image Loaded.")
                if !valid_grid() {
                    lblStatus.SetText("Status: Image unable to be
loaded.")
                    n = 0
                    found = false
                    cnt = 0
                } else {
                    build_grid()
                }
            } else {
                fmt.Println("Eror on image process")
            }
        }, w)

        fd.SetFilter(storage.NewExtensionFileFilter([]string{".png",
".jpg", ".jpeg"}))
        fd.Show()
    })

    lblSlider := widget.NewLabel("Update speed slider (ms): ")
    slider := widget.NewSlider(1, 500)
    slider.SetValue(50)
    slider.OnChanged = func(v float64) { delay = int(v) }

    uiChan = make(chan bool)
    go func() {
        for range uiChan {
```

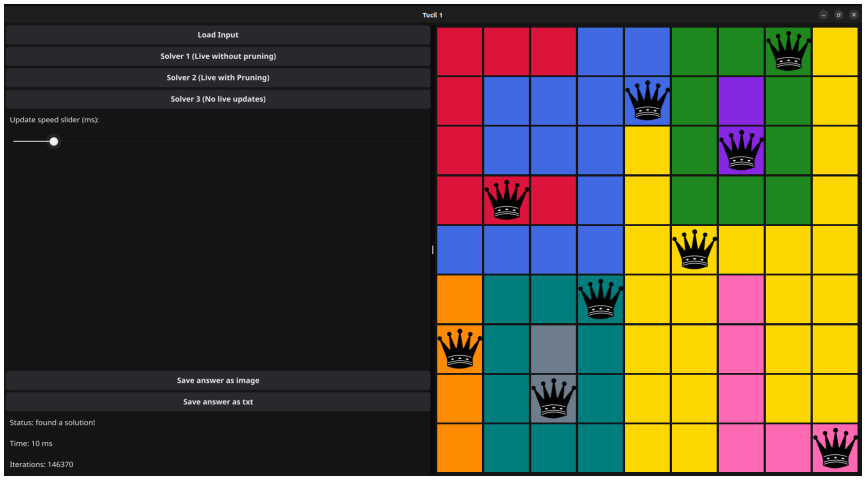
```
            refresh_grid()
        }
    }()

    sidepanel := container.NewVBox(
        btnLoad,
        widget.NewLabel("Or... upload as a file:"),
        entryN,
        btnLoadImg,
        btn1, btn2, btn3,
        lblSlider,
        slider,
        layout.NewSpacer(),
        btnSaveImg,
        btnSaveTxt,
        lblStatus, lblTime, lblIter,
    )

    split := container.NewHSplit(sidepanel, container.NewPadded(grid))

    w.SetContent(split)
    w.ShowAndRun()
}
```

3. Screenshot Aplikasi

- Contoh 1 : (input-output normal)

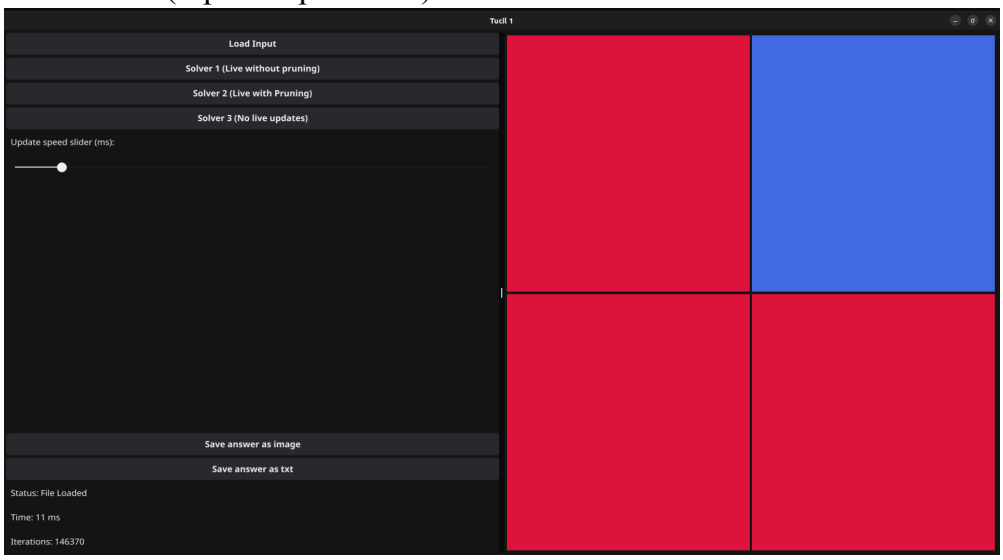


Gambar 1.1 Input dengan solusi

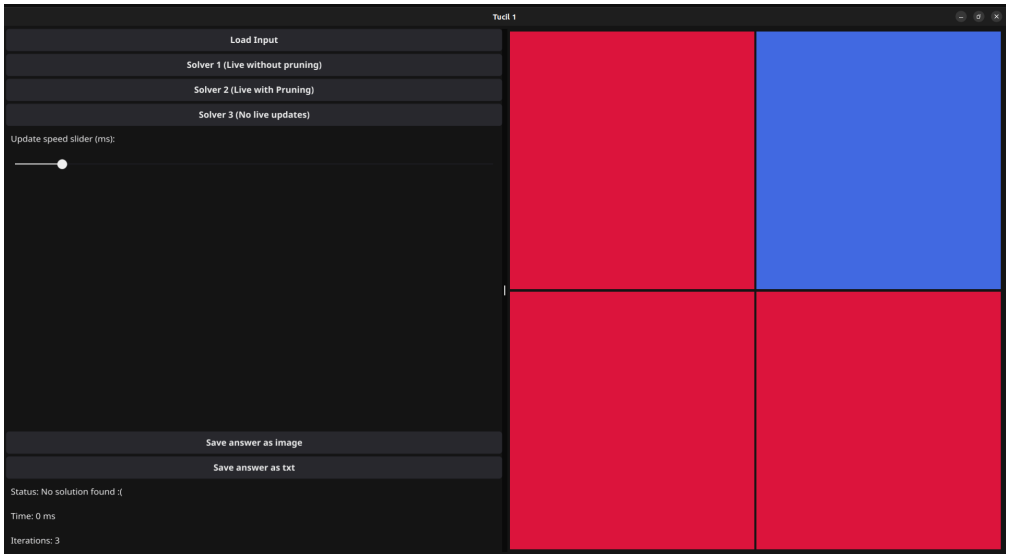


Gambar 1.2 Salah satu solusi input 1.1

- Contoh 2 : (input tanpa solusi)

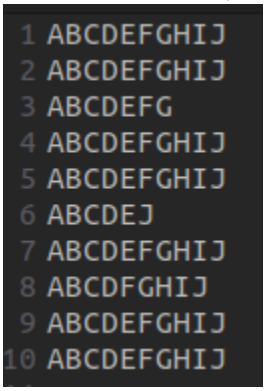


Gambar 2.1 Input tanpa solusi

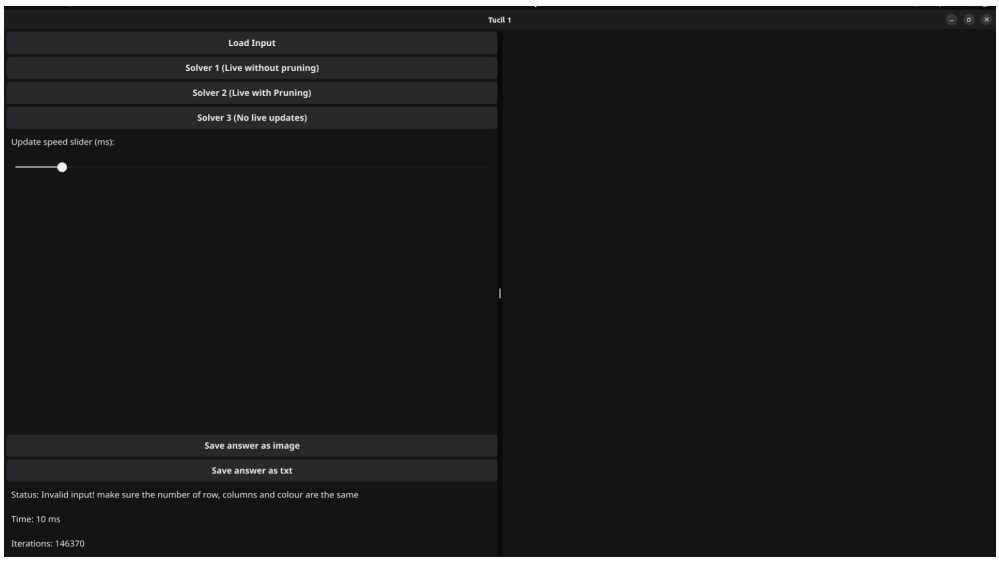


Gambar 2.2 Laporan bahwa tidak terdapat solusi untuk input 2.1

- Contoh 3 : (input yang tidak sesuai format)

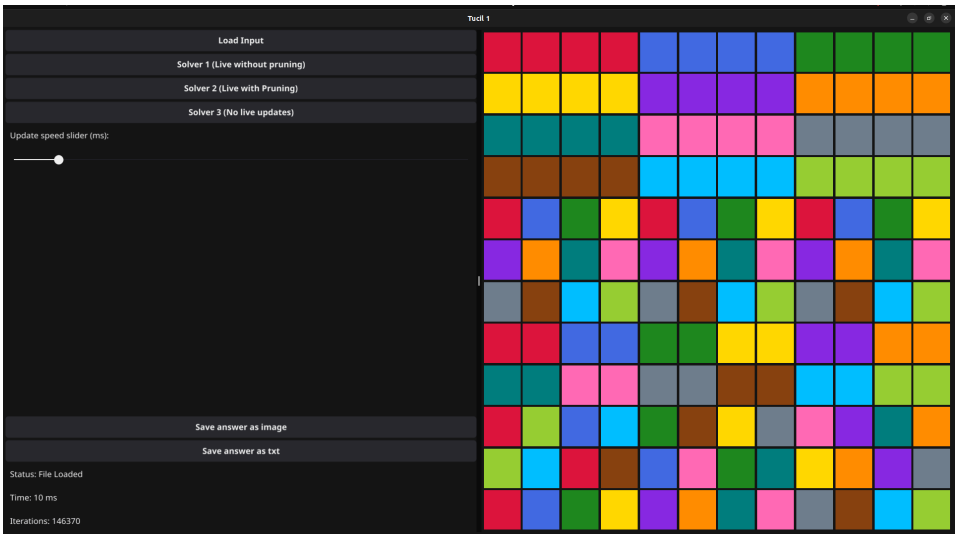


Gambar 3.1 Input tidak sesuai format

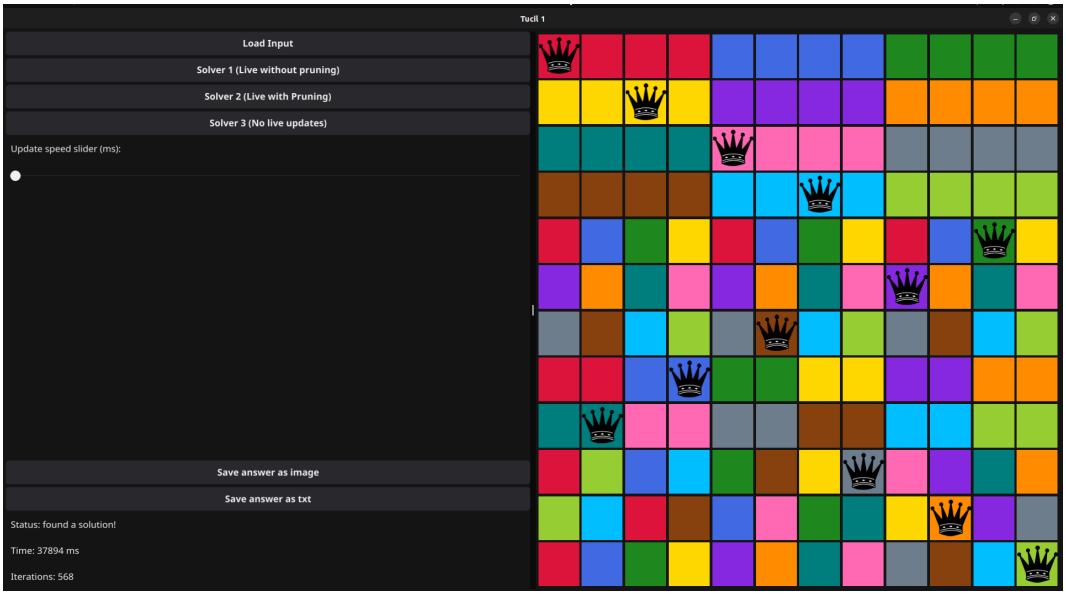


Gambar 3.2 Tampilan input yang tidak sesuai (dengan status input tidak sesuai)

- Contoh 4 : (input besar dengan live update dengan pruning)

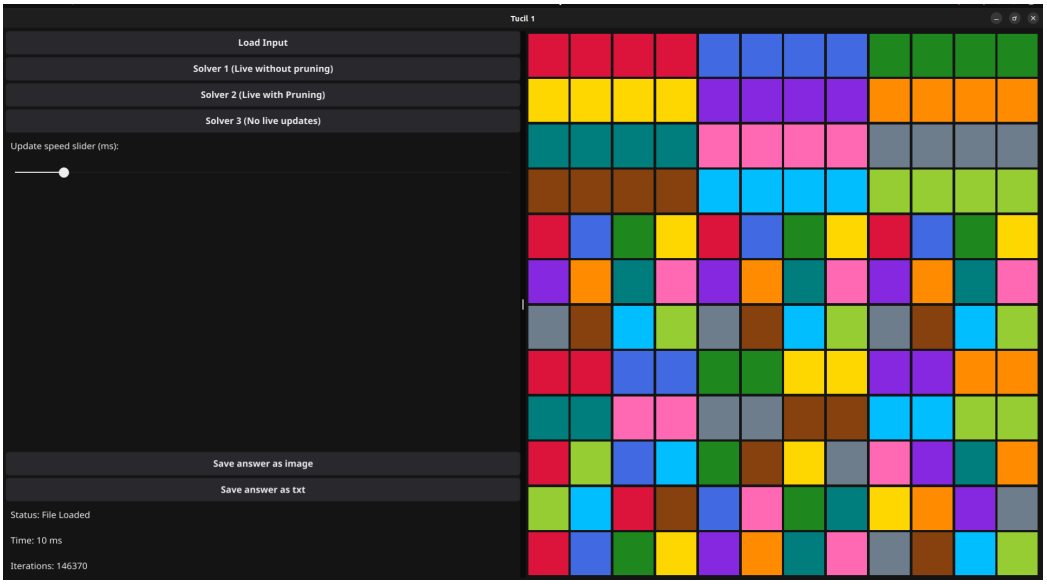


Gambar 4.1 Tampilan input 12x12 sebelum solver1

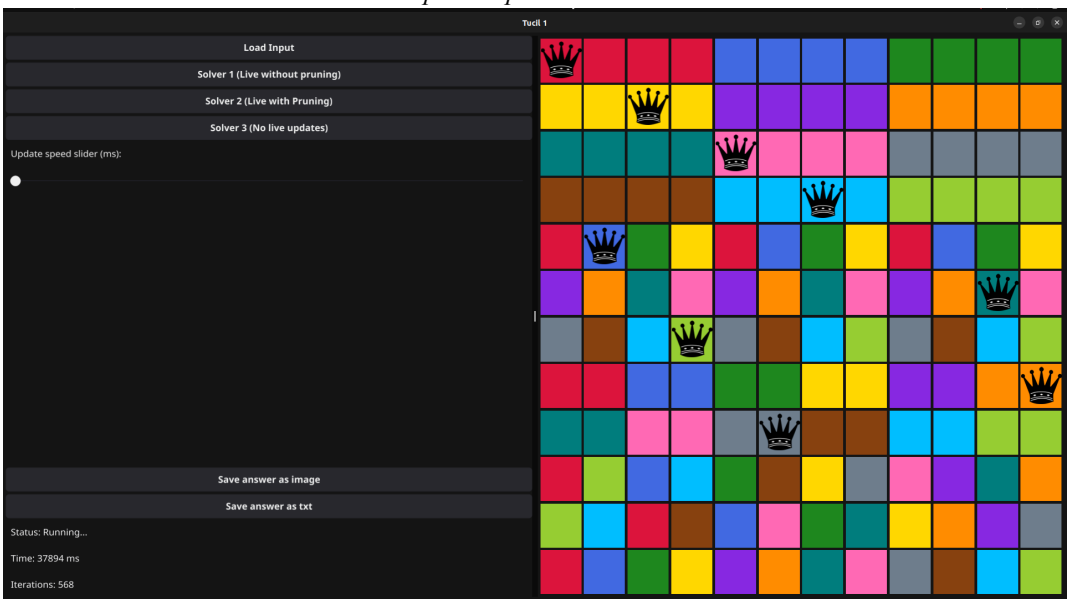


Gambar 4.2 Tampilan akhir solver1 pada input 12x12 (37894 ms)

- Contoh 5 : (input besar dengan live update tanpa pruning)

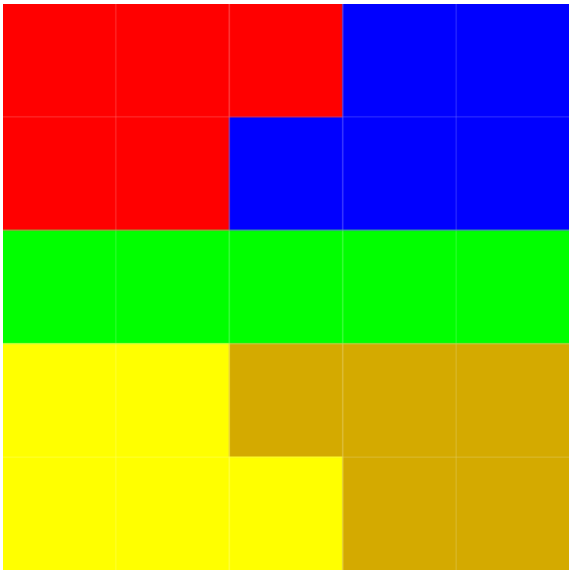


Gambar 5.1 Tampilan input 12x12 sebelum solver2

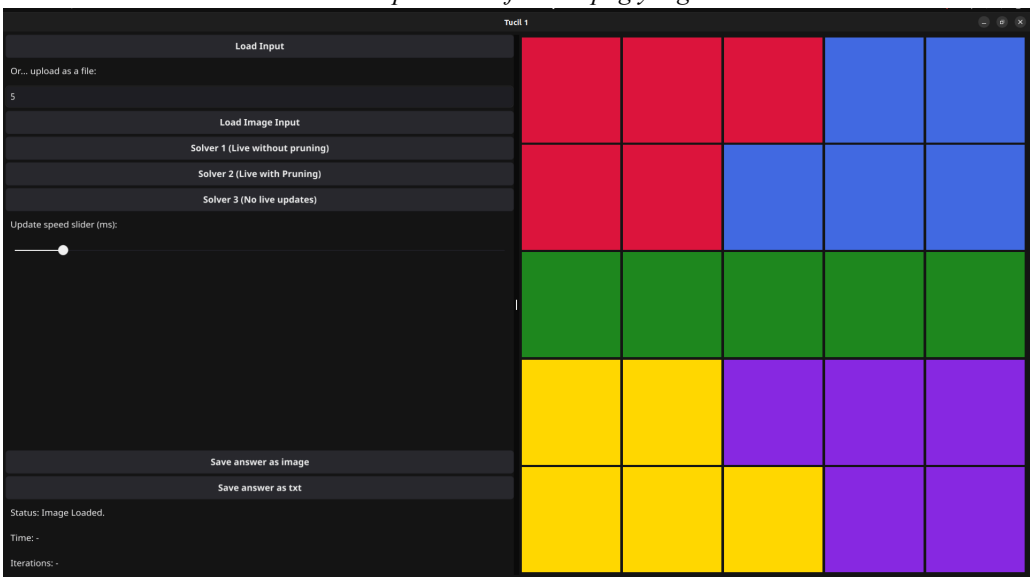


Gambar 5.2 Tampilan live update setelah 30 menit

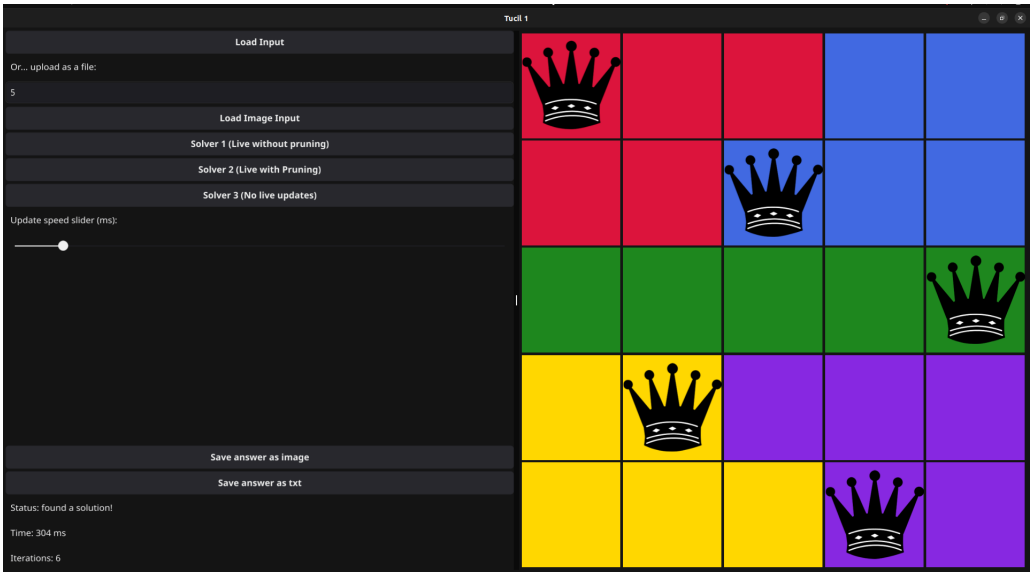
- Contoh 6 : (input dengan gambar dengan N yang tepat)



Gambar 6.1 Input dalam format .png yang diberikan

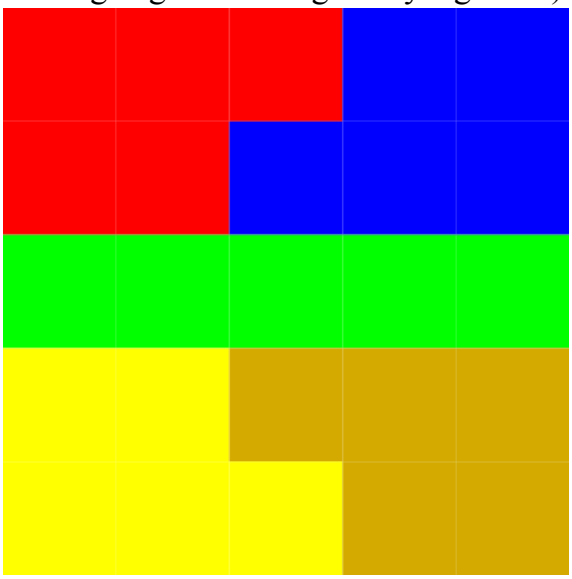


Gambar 6.2 Hasil load dengan input N = 5

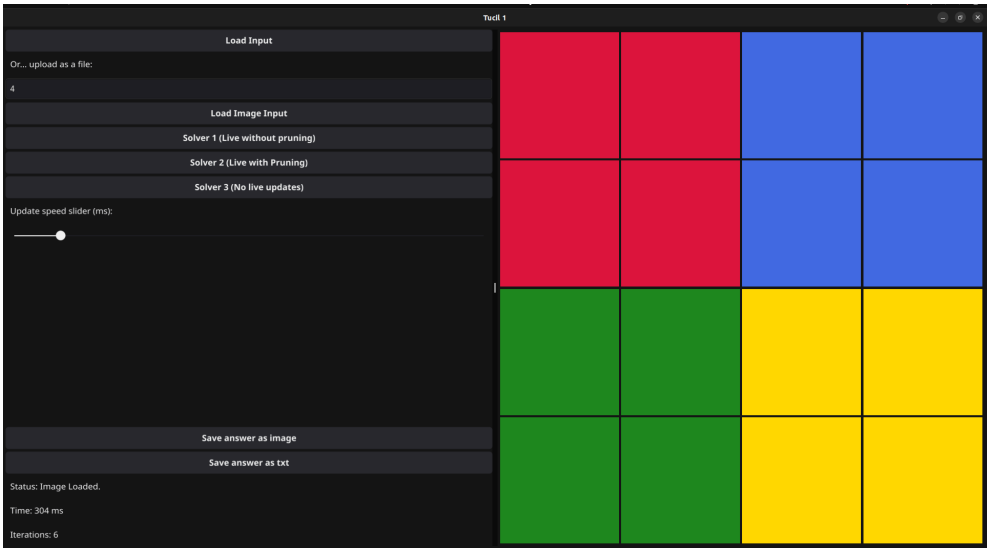


Gambar 6.3 Hasil penyelesaian dengan Solver1

- Contoh 7 : (input dengan gambar dengan N yang salah)



Gambar 7.1 Input dalam format .png yang diberikan



Gambar 7.2 Hasil load yang salah (hal ini menunjukkan kesalahan input nilai N dapat menyebabkan undefined behaviour pada load gambar)

4. Pranala Repositori

https://github.com/YeyThePotatoMan/Tucil1-13524040

5. Pernyataan tidak melakukan kecurangan

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

Kloce Paul William Saragih

6. Lampiran

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