Pembangunan Perangkat Lunak dan Penyelesaian Permainan Colored Queens

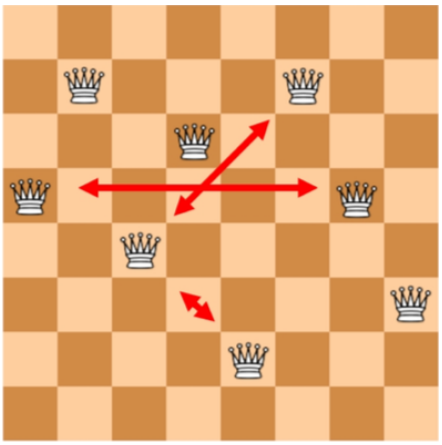
Arlo Dante Hananvyasa–6182201010

1 Data Tugas Akhir

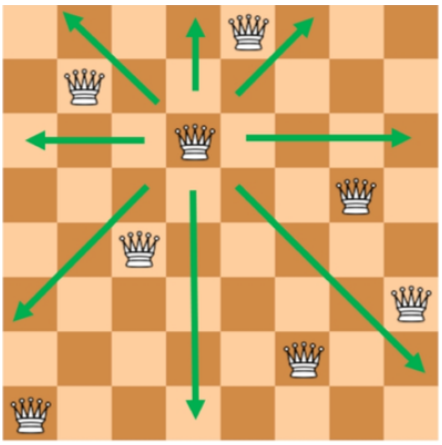
Pembimbing utama/tunggal: **Husnul Hakim, M.T.**
Pembimbing pendamping: -
Kode Topik : **HUH5902ACS**
Topik ini sudah dikerjakan selama : **1** semester
Pengambilan pertama kali topik ini pada : Semester **40 - Genap 15/16**
Pengambilan pertama kali topik ini di kuliah : **Tugas Akhir 1**
Tipe Laporan : **B -** Dokumen untuk reviewer pada presentasi dan **review Tugas Akhir 1**

2 Latar Belakang

Masalah n-queens merupakan salah satu permasalahan klasik dalam ilmu komputer yang telah dipelajari secara ekstensif sejak abad ke-19. Dalam bentuk standarnya, masalah n-queens memerlukan penempatan n buah bidak menteri pada papan catur berukuran $n \times n$ sedemikian rupa sehingga tidak ada menteri yang dapat menyerang satu sama lain secara horizontal, vertikal, maupun diagonal. Sebagai contoh, pada papan berukuran 8×8, terdapat 92 solusi valid yang memenuhi seluruh kendala tersebut. Masalah ini tidak hanya menarik dari segi teoretis, tetapi juga memiliki aplikasi praktis dalam berbagai bidang seperti penjadwalan, alokasi sumber daya, dan desain sirkuit terpadu, sehingga menjadikannya salah satu tolok ukur penting dalam penelitian algoritma pencarian dan pemodelan berbasis kendala.



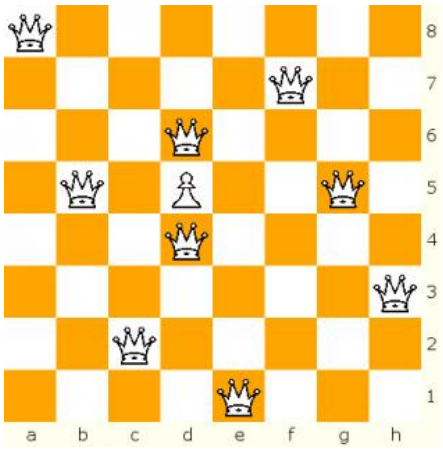
Gambar 1(a) Contoh solusi salah masalah N-Queens



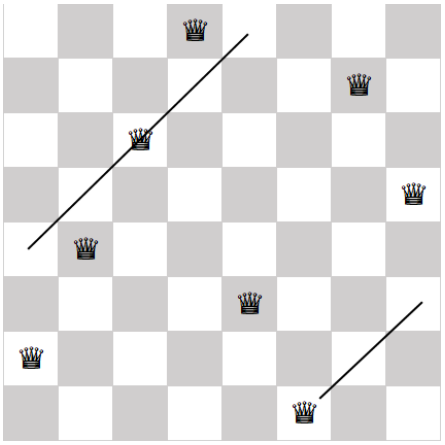
Gambar 1(b) Contoh valid permasalahan N-Queens

Seiring perkembangan penelitian, muncul berbagai variasi dari masalah n-queens tradisional yang menawarkan kompleksitas dan tantangan komputasional yang lebih tinggi. Salah satu variasi yang telah diteliti adalah *N+k Queens Problem*, di mana k buah bidak pion ditempatkan sebagai penghalang sehingga memungkinkan penempatan $N + k$ bidak menteri pada papan berukuran $N \times N$. Variasi lainnya adalah *Toroidal N-Queens*, di mana papan catur dibentuk menjadi torus dengan menghubungkan sisi-sisi yang berlawanan, sehingga menteri dapat "melingkar" dari satu sisi ke sisi lainnya. Peningkatan kompleksitas pada variasi-variasi ini terlihat jelas; misalnya pada *N+k Queens Problem* dengan $N = 8$ dan $k = 2$, jumlah kemungkinan

konfigurasi yang harus dieksplorasi meningkat secara signifikan dibandingkan masalah standar karena adanya kendala tambahan berupa posisi pion yang tidak dapat dilanggar. Variasi-variasi ini menunjukkan bahwa menambahkan satu atau dua aturan baru saja dapat memperbesar ruang pencarian secara drastis dan mengubah struktur solusi problem secara fundamental.

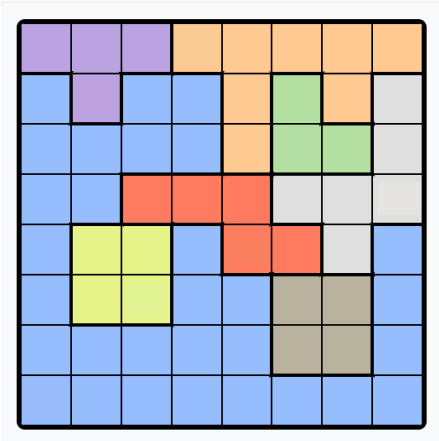


Gambar 2(a) Contoh solusi salah masalah *8+1 Queens*
Sumber: `http://www.npluskqueens.info/background.html`

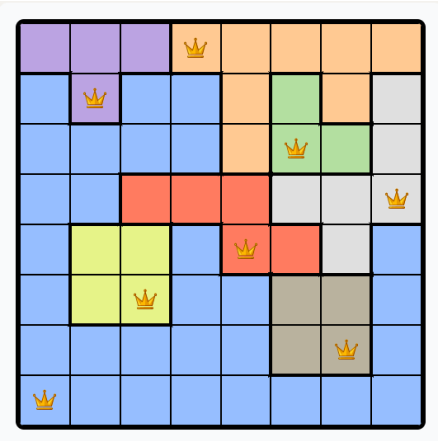


Gambar 2(b) Contoh penyerangan yang dapat terjadi pada masalah *Toroidal Queens*
Sumber: `https://www.johndcook.com/blog/2021/08/18/queens-on-a-donut/`

Penelitian ini berfokus pada varian *Colored Queens*, sebuah permasalahan yang hingga saat ini belum memiliki literatur akademis yang memadai dan belum diteliti secara formal dalam publikasi ilmiah. Berbeda dengan masalah n-queens tradisional, permainan *Colored Queens* memiliki aturan yang lebih kompleks: papan permainan dibagi menjadi beberapa sektor berwarna, setiap sektor harus berisi tepat satu bidak menteri, dan tidak ada menteri yang boleh bersebelahan secara langsung, baik horizontal, vertikal, maupun diagonal. Perbedaan fundamental lainnya adalah bahwa bidak menteri pada *Colored Queens* hanya dapat menyerang secara horizontal dan vertikal, sehingga lebih dari satu bidak dapat ditempatkan pada satu garis diagonal yang sama. Kompleksitas komputasional varian ini lebih tinggi dibandingkan permasalahan n-queens tradisional karena terdapat tiga lapis kendala yang harus dipenuhi secara simultan: kendala partisi warna (setiap warna tepat satu menteri), kendala *adjacency* (tidak boleh bersebelahan), dan kendala serangan (horizontal dan vertikal). Sebagai ilustrasi, pada papan berukuran 6×6 dengan 6 sektor warna, ruang pencarian solusi menjadi jauh lebih sempit dan bergantung pada struktur pembagian sektor, sehingga kompleksitasnya umumnya lebih tinggi dibandingkan 6-queens standar yang memiliki ruang solusi yang lebih teratur. Ketidakteraturan struktur sektor warna juga membuat heuristik dan simetri yang biasa dimanfaatkan pada n-queens standar tidak lagi berlaku, sehingga *Colored Queens* merupakan permasalah yang jauh lebih sulit.



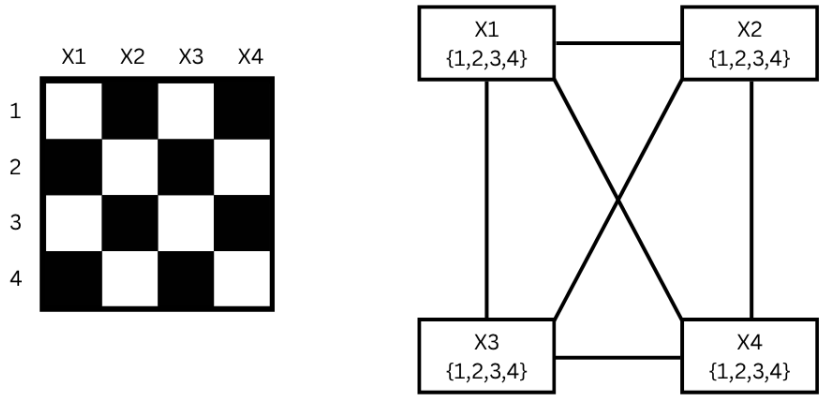
Gambar 3(a) Contoh kondisi awal permainan Colored Queens



Gambar 3(b) Solusi valid permainan Colored Queens

Masalah n-queens dan seluruh variasinya, termasuk *Colored Queens*, tergolong ke dalam kategori *Constraint Satisfaction Problem* (CSP). CSP adalah jenis permasalahan yang melibatkan pencarian solusi dengan memberikan nilai pada sejumlah variabel sedemikian rupa sehingga memenuhi seperangkat batasan atau kendala tertentu. Secara formal, sebuah CSP didefinisikan oleh tiga komponen: himpunan variabel $X = X1, X2, ..., Xn$, himpunan domain $D = D1, D2, ..., Dn$ yang berisi nilai-nilai yang mungkin untuk setiap variabel, dan himpunan kendala C yang membatasi kombinasi nilai yang dapat diberikan pada variabel-variabel tersebut. Dalam konteks *Colored Queens*, variabel-variabelnya adalah posisi menteri untuk setiap sektor warna, domain untuk setiap variabel adalah sel-sel yang tersedia pada sektor tersebut, dan kendala-kendalanya mencakup aturan tidak bersebelahan serta tidak saling menyerang. CSP memiliki aplikasi luas dalam penjadwalan, perencanaan, konfigurasi produk, desain jaringan, serta berbagai sistem pengambilan keputusan berbasis kendala.

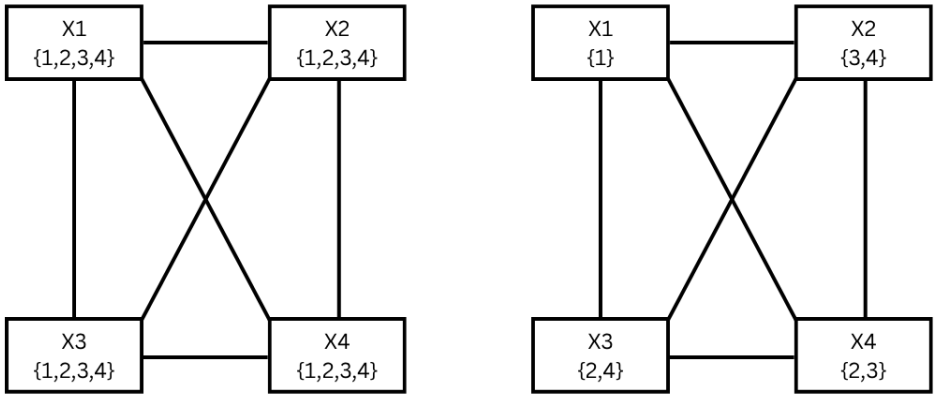
Penyelesaian permasalahan CSP, terutama yang memiliki ruang pencarian besar dan kompleks seperti *Colored Queens*, dapat didekati melalui perspektif optimasi. Optimasi adalah proses pencarian solusi terbaik dari sekumpulan alternatif yang memungkinkan, biasanya dengan memaksimalkan atau meminimalkan suatu fungsi objektif tertentu. Dalam konteks CSP, fungsi objektif dapat berupa jumlah kendala yang dilanggar, sehingga solusi optimal adalah konfigurasi yang tidak melanggar kendala sama sekali (nilai objektif nol). Algoritma optimasi umumnya bekerja dengan menggabungkan strategi eksplorasi dan eksploitasi terhadap ruang solusi: eksplorasi memungkinkan algoritma menjelajahi area solusi baru, sedangkan eksploitasi berfokus pada penyempurnaan solusi yang sudah dianggap menjanjikan. Keseimbangan antara keduanya sangat penting, karena terlalu banyak eksploitasi berpotensi membuat algoritma terjebak pada solusi lokal, yaitu kondisi ketika algoritma menemukan konfigurasi yang tampak optimal di sekitar posisinya saat ini tetapi sebenarnya bukan solusi terbaik secara global. Perspektif optimasi menjadi relevan karena banyak permasalahan CSP tergolong *NP-hard*, yaitu kelas masalah yang tidak diketahui memiliki algoritma yang dapat diselesaikan secara efisien dalam waktu polinomial. Waktu polinomial berarti waktu komputasi yang meningkat secara wajar terhadap ukuran input (misalnya $O(n^2)$ atau $O(n^3)$), sedangkan masalah NP-hard umumnya membutuhkan waktu eksponensial yang meningkat jauh lebih cepat sehingga tidak praktis untuk diselesaikan secara eksak pada ukuran besar, sehingga metode optimasi menjadi alternatif yang lebih realistis.



Gambar 4 Contoh permasalahan *4-Queens* yang dimodelkan sebagai CSP, dimana variabel X1 s.d. X4 adalah baris bidak menteri pada setiap kolomnya dengan nilai kemungkinan 1,2,3,4 dan kendala yang menghubungakn setiap variabel berupa aturan penyerangan bidak menteri pada permainan catur

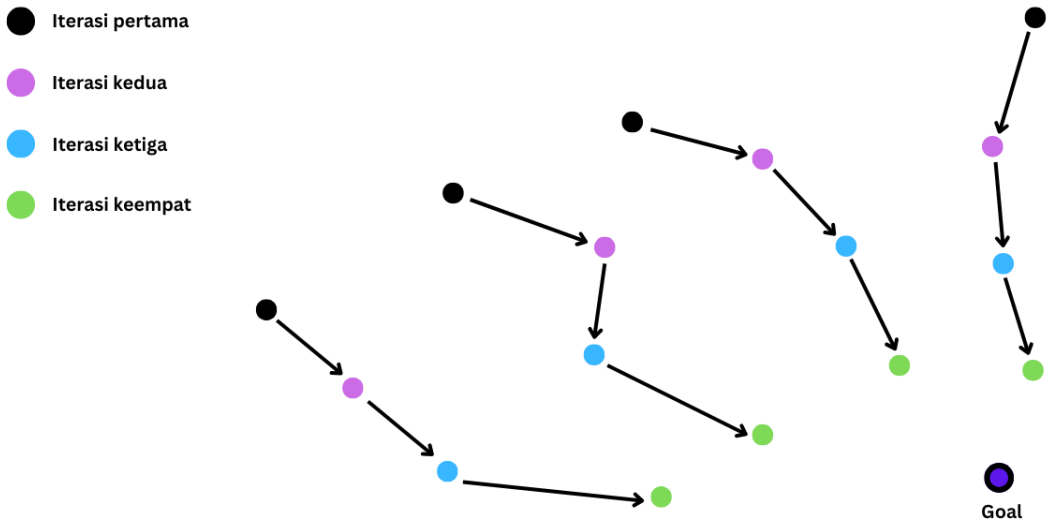
Untuk menyelesaikan permasalahan CSP seperti *Colored Queens*, salah satu pendekatan klasik yang terbukti efektif adalah algoritma *backtracking*. *Backtracking* bekerja dengan membangun solusi secara bertahap melalui pencarian mendalam, di mana setiap variabel diberi nilai satu per satu sambil memeriksa konsistensi dengan kendala yang ada. Ketika algoritma menemui situasi di mana tidak ada nilai yang valid untuk variabel berikutnya (*dead end*), algoritma akan mundur (*backtrack*) ke langkah sebelumnya dan mencoba alternatif lain. Kekuatan *backtracking* terletak pada sifatnya yang sistematis dan kemampuannya untuk menjamin bahwa solusi akan ditemukan jika memang ada. Namun, pada masalah dengan ruang pencarian yang berkembang secara eksponensial seperti *Colored Queens*, *backtracking* murni dapat menjadi sangat tidak efisien karena harus mengeksplorasi sejumlah besar kemungkinan konfigurasi sebelum menemukan solusi yang valid. Oleh karena itu, diperlukan modifikasi dan teknik tambahan untuk memangkas ruang pencarian agar algoritma tetap praktis digunakan pada instansi masalah *Colored Queens* yang berukuran besar seperti papan 20×20 dan 30×30.

Untuk meningkatkan efisiensi algoritma backtracking, penelitian ini mengintegrasikan teknik *Maintaining Arc Consistency* (MAC) yang mengimplementasikan algoritma *Arc Consistency 3* (AC-3). AC-3 adalah teknik propagasi kendala yang berfungsi untuk menyempitkan ruang pencarian dengan mengeliminasi nilai-nilai dalam domain yang tidak mungkin menghasilkan solusi valid berdasarkan kendala antar variabel. Algoritma ini bekerja dengan memeriksa konsistensi antara pasangan variabel dan secara iteratif menghapus nilai-nilai yang tidak memiliki pasangan yang konsisten pada variabel lain. Dengan menjalankan AC-3 pada tahap preprocessing sebelum pencarian dimulai serta setiap kali backtracking membuat assignment baru, banyak konfigurasi yang tidak valid dapat dieliminasi lebih awal sebelum algoritma membuang waktu mengeksplorasinya. Pada problem seperti *Colored Queens* yang memiliki ketergantungan antar variabel sangat kuat akibat kendala *adjacency* dan partisi warna, propagasi kendala memberikan dampak reduksi ruang solusi yang signifikan sehingga dapat mengurangi jumlah langkah *backtracking* yang diperlukan secara drastis dan memperpendek waktu yang dibutuhkan untuk mencari solusi yang valid.



Gambar 5 Contoh ruang solusi *4-Queens* yang dimodelkan sebagai CSP, sebelum(kiri) dan sesudah(kanan) menjalankan AC-3 pada saat menaruh bidak menteri di posisi 1 di kolom X1

Sebagai alternatif dari pendekatan deterministik, penelitian ini juga mengeksplorasi penggunaan *Particle Swarm Optimization* (PSO), sebuah algoritma metaheuristik yang terinspirasi dari perilaku kolektif kawanan burung atau ikan dalam mencari makanan. Metaheuristik adalah strategi pencarian tingkat tinggi yang memandu proses eksplorasi ruang solusi tanpa menjamin menemukan solusi optimal, namun seringkali dapat menemukan solusi yang cukup baik dalam waktu yang lebih singkat dibandingkan metode eksak seperti backtracking. PSO bekerja dengan mensimulasikan sekumpulan partikel yang bergerak dalam ruang solusi, di mana setiap partikel menyesuaikan posisinya berdasarkan pengalaman terbaiknya sendiri dan pengalaman terbaik kawanan. Kelebihan PSO terletak pada kemampuannya untuk melakukan eksplorasi ruang solusi secara paralel dan menghindari jebakan solusi lokal dalam beberapa kasus. Namun, karena PSO dirancang untuk ruang solusi kontinu, penerapannya pada *Colored Queens* yang bersifat diskret memerlukan adaptasi khusus. Adaptasi tersebut mencakup representasi setiap partikel sebagai susunan posisi menteri untuk setiap sektor warna pada papan. Konsep velocity dalam PSO juga dimodifikasi menjadi bilangan desimal yang merepresentasikan probabilitas seberapa besar kemungkinan posisi menteri pada suatu sektor akan beralih mengikuti konfigurasi Neighbourhood Best, serta prosedur perbaikan (*repair mechanisms*) untuk menangani solusi yang melanggar kendala hard constraint seperti aturan satu menteri per warna.. Adaptasi ini penting agar PSO tetap relevan dan dapat memberikan performa yang kompetitif pada problem bersifat kombinatorial.



Gambar 6 Contoh pergerakan partikel-partikle ke arah solusi optimal dengan setiap iterasi
Sumber: Modifikasi dari `https://www.researchgate.net/figure/sualization-of-the-PSO-Algorithm_fig3_334363118`

Dari segi pengembangan perangkat lunak, penelitian ini tidak hanya berfokus pada aspek algoritmik, tetapi juga pada penyediaan antarmuka pengguna yang intuitif dan edukatif. Akan dikembangkan sebuah aplikasi berbasis web yang memungkinkan pengguna untuk berinteraksi langsung dengan permainan *Colored Queens*, memilih tingkat kesulitan berdasarkan ukuran papan, serta memungkinkan pengguna untuk menyelesaikan *puzzle* secara manual. Aplikasi ini akan dilengkapi sistem peringatan visual ketika pengguna menempatkan menteri pada posisi yang melanggar kendala, sehingga membantu pemahaman terhadap aturan permainan. Selain mode permainan manual, aplikasi juga akan menyediakan mode demonstrasi yang menampilkan proses pencarian solusi oleh algoritma backtracking dan PSO secara visual, lengkap dengan metrik performa seperti waktu eksekusi, jumlah iterasi, dan perbandingan efektivitas kedua metode. Pendekatan ini diharapkan dapat mengurangi kesenjangan antara teori algoritma dan aplikasi praktis, serta memberikan alat pembelajaran interaktif bagi pengguna yang tertarik mempelajari teknik penyelesaian CSP.

3 Rumusan Masalah

- Bagaimana cara membangun perangkat lunak permainan *Colored Queens*?
- Bagaimana membangun solusi permainan *Colored Queens* menggunakan teknik *Backtracking* dan *Particle Swarm Optimization* (PSO)?
- Bagaimana membangun solver untuk permainan colored queen yang mengimplementasikan *Backtracking* dan PSO yang dapat diintegrasikan dengan perangkat lunak yang dibangun?
- Bagiamana kinerja dari solver yang dibangun dalam mencari solusi permainan *Colored Queens*?

4 Tujuan

- Membangun perangkat lunak permainan *Colored Queens*.
- Mempelajari cara membangun solusi permainan Colored Queen menggunakan teknik *Backtracking* dan *Particle Swarm Optimization*.
- Membangun solver untuk permainan colored queen yang mengimplementasikan *Backtracking* dan PSO yang dapat diintegrasikan dengan perangkat lunak yang dibangun.

- Melakukan pegujian untuk mengukur kinerja dari solver yang dibangun dalam mencari solusi permainan *Colored Queens*

5 Detail Perkembangan Pengerjaan Tugas Akhir

Detail bagian pekerjaan skripsi sesuai dengan rencana kerja/laporan perkembangan terkahir :

1. **Melakukan studi literatur terkait permasalahan n-queens dan variannya, Constraint Satisfaction Problem (CSP), algoritma pencarian Backtracking, teknik metaheuristik Particle Swarm Optimization, serta metode propagasi kendala AC-3).**
   Status : Ada sejak rencana kerja skripsi.
   Hasil :

   (a) **Permasalahan N-Queens**

   (b) **Berbagai Varian N-Queens**

   (c) ***Constraint Satisfaction Problems* (CSP)**

   (d) **Algoritma *Backtracking***

   (e) ***Maintaining Arc Consistency* dengan algoritma *Arc Consistency 3* (AC-3)**

   (f) **Algoritma *Particle Swarm Optimization* (PSO)**

2. **Mengumpulkan dan menyusun berbagai skenario permasalahan Colored Queens yang akan digunakan sebagai basis pengujian algoritma serta sebagai pilihan tingkat kesulitan bagi pengguna.**
   Status : Ada sejak rencana kerja skripsi.
   Hasil :

   Sebagai bagian dari tahap pengumpulan dan penyusunan skenario permasalahan Colored Queens, saya membangun sebuah web scraper Python yang menggunakan library Selenium untuk mengekstraksi beragam konfigurasi papan dari situs Play Queens Game. Data ini diperlukan sebagai dasar pengujian untuk algoritma Backtracking dan Particle Swarm Optimization (PSO), sekaligus menjadi kumpulan level yang dapat dipilih pengguna berdasarkan tingkat kesulitan. Skrip yang dibuat mengotomatisasi proses pengambilan puzzle untuk berbagai ukuran papan—mulai dari 7×7 hingga 11×11—dengan jumlah level yang bervariasi pada setiap ukuran. Setiap puzzle diakses melalui URL tertentu, kemudian Selenium membaca atribut setiap sel (baris, kolom, dan warna) untuk membentuk representasi papan yang terstruktur.

   Hasil ekstraksi disimpan dalam bentuk berkas JSON yang menyimpan posisi baris, kolom, serta warna dari setiap kotak di dalam papan, agat mudah diproses kembali oleh *solver* maupun antarmuka pengguna. Pendekatan ini memastikan bahwa dataset berisi skenario yang beragam, valid, dan sesuai dengan mekanisme permainan asli. Dengan adanya kumpulan skenario ini, pengujian algoritma dapat dilakukan pada berbagai tingkat kesulitan, dan pengguna aplikasi dapat memilih puzzle berdasarkan ukuran papan.

   Selain puzzle yang diambil dari situs daring tersebut, dua papan vaild tambahan berukuran 20×20 dan 30×30 telah dibuat untuk pengujian algoritma yang akan diimplementasikan pada tugas akhir ini. Pembuatan papan-papan baru ini dimulai dengan sebuah solusi valid permasalahan N-Queens lalu menambahan sektor-sektor warna sedemikian rupa agar hanya terdapat satu bidak menteri di dalam satu sektor warna.

   Cuplikan file JSON:

```
[
{
    "row": 0, //posisi baris
    "col": 0, //posisi kolom
    "color": "rgba(253, 224, 71, 1)" //warna
},
...
```

Cuplikan fungsi inti program Python:

```python
for cell in cells: # untuk setiap cell/kotak dalam papan:
row = int(cell.get_attribute("data-row")) # ambil posisi baris
col = int(cell.get_attribute("data-col")) # ambil posisi kolom
color = cell.value_of_css_property("background-color") # ambil warna kotak tersebut
# menambahkan file JSON sesuai format
board.append({
    "row": row,
    "col": col,
    "color": color
})
```

Keseluruhan Program Python:

```python
from selenium import webdriver
from selenium.webdriver.common.by import By
import chromedriver_autoinstaller
import json
import time
import os

# Auto-install ChromeDriver
chromedriver_autoinstaller.install()

# Base URL
BASE_URL = "https://www.playqueensgame.com/puzzles/{size}x{size}/{level}"

# Define board sizes and number of levels
board_levels = {
    7: 50,
    8: 130,
    9: 110,
    10: 60,
    11: 50
}

# Create output folder
os.makedirs("boards", exist_ok=True)

# Launch browser
driver = webdriver.Chrome()

for size, max_level in board_levels.items():
for level in range(1, max_level + 1):
url = BASE_URL.format(size=size, level=level)
driver.get(url)
time.sleep(2)  # allow page to load

# Extract cells
cells = driver.find_elements(By.CSS_SELECTOR, "div[data-row][data-col]")
board = []
for cell in cells:
row = int(cell.get_attribute("data-row"))
col = int(cell.get_attribute("data-col"))
color = cell.value_of_css_property("background-color")
board.append({
    "row": row,
    "col": col,
    "color": color
})

# Sort for consistency
board.sort(key=lambda c: (c["row"], c["col"]))

# Save JSON
filename = f"boards/{size}x{size}_level{level}.json"
with open(filename, "w") as f:
json.dump(board, f, indent=2)

print(f"Saved_{filename}")

driver.quit()
print("All_boards_extracted!")
```

Papan 20×20 valid (setiap warna dilambangkan sebagai sebuah karakter yang dimulai dari kode ASCII 'A'):

```
J J J J J J J G G G G G E E E E E I I I I
J J J J J J J G G G G G E E E E E I I I I
```

```
J J J J J J J G G G G E E E E E E I I I I
D D D J J J J G G E E E E E E R R I I I I
D D D D D D D D D G H E R R R R R R I I I I
D D D D D D D U U H H H H P P R R R O I I
D D D D D U U U H H H H P P P P P O O I
D D D D U U U U H H H H H P P O O O O O O
D D D D S U U U U H H H H P P O O O O O O
M D S S S U U U U U H H H P K K K O O O
M M S S S S U U U F H H H K K K K K K K
M S S S S S S F F F L H K K K K K K K K
M M M S S S F F F F L L L K K K K K K K
M M M S S T F F F L L L L L K K K K C C
M B B B T T L L L L L L L L L K K C C C
B B B B T T T L L L L L L L A A C C C C
B B B T T T T T N L L L L A A A C C C C
B B B T T T T N N N N N A A A A C C C C
B B B T T T N N N N N N A A A A A C C C
B B B T T T N N N N N N A A A A A A C C
```

Papan 30×30 valid setiap warna dilambangkan sebagai sebuah karakter yang dimulai dari kode ASCII 'A':

```
I I I I I I I ^ ^ ^ ^ F F F W W W W W W V V V V V V V V V V
I I I I I I ^ ^ ^ ^ ^ F F F F W W W W V V V V V V V V V V V
I I I I _ _ ^ ^ ^ ^ ^ F F F F F W W V V V V V V V V V V V V
I I I I _ _ _ ^ ^ ^ ^ F F F F F W V V V V V V V V V V V V Z
I I I _ _ _ _ ^ ^ ^ ^ ^ F F O O O O V V V V V V V V V Z Z Z
I I _ _ _ _ _ _ ^ ^ ^ O O O O O O O O V V V V V V Z Z Z Z Z
I _ _ _ _ _ _ _ ^ ^ D [ [ O O C C C C C C V V V V Z Z Z Z Z Z
I _ _ _ _ _ _ _ ^ ^ D [ [ [ C C C C C C C C V R R Z Z Z Z Z Z
_ _ _ _ _ _ _ D D D D [ [ [ C C C C C C C C C R R R Z Z Z Z Z Z
\ _ _ G G G G G D D D D [ [ C C C C U U R R R R Z Z Z Z Z Z Z
\ \ G G G G G G G D D D [ [ [ C C C U U R R R R Z Z Z Z Z Z Z
\ \ G G G G G G G D D D [ [ [ C C U U U R R R R R P Z Z Z Z Z
\ \ \ \ G G G G G G D D [ [ [ U U U U U R R R R R P P Z Z Z Z
\ \ \ \ \ \ G G G G G D D [ U U U U U U U U R R R P P P Z Z Z Z
\ \ \ \ \ \ \ G G G G D D [ [ U U U U U U U U R R P P P P P P P
\ \ \ \ \ \ \ G G G B D D [ [ U U U U U U U U R R P P P ] ] ] ]
\ \ \ \ \ \ \ \ B B B T T T T U Y Y U Y U S S L L L ] ] ] ] ]
M M \ \ \ \ \ \ B B B B T T T Y Y Y Y Y S S S L L L L ] ] ] ]
M M M M M M M B B B B T T T T Y Y Y Y Y S S S S L L L ] ] ] ]
M M M M M M B B B B T T T T Y Y Y Y Y Y S S S L L L ] ] H H
M M M M M B B B B B T T T T Y Y Y Y Y Y S S S L L L ] H H H
M M M X X B B B B B T T T T T Y Y Y Y Y Y Y A A L L H H H H H
M M X X X X B B B B T T T T T T Y Y Y Y Y Y A A A H H H H H H
X X X X X X B B B T T T T T T N N Y Y Y Y A A A A H H H H H H
X X X X X X X B B T T T T T N N N N Y A A A A A A A H H H H
X X X X X X X J B T T T T T N N N N N A A A A A A A A K K K
X X X X E E J J J J T T N N N N N N N N N A A A A A A K K K K
E E E E E E E J J J J N N N N N N N N N N A A A A A K K K K
E E E E E E E J J J J J N N N N N N N N N A A A A A K K K K
E E E E E J J J J J J J N N N N N N N N N A A A A A K K K K
```

3. **Melakukan pemodelan masalah Colored Queens ke dalam bentuk CSP agar dapat diproses oleh algoritma pencarian.**
   Status : Ada sejak rencana kerja skripsi.
   Hasil :

   Pemodelan permainan Colored Queens sebagai *Constraint Satisfaction Problem* (CSP) dilakukan untuk memastikan bahwa aturan permainan dapat diproses secara konsisten oleh algoritma Backtracking maupun Particle Swarm Optimization (PSO). Pemodelan ini dilakukan dengan mengidentifikasi tiga komponen utama dalam CSP, yaitu variabel, domain atau ruang solusi, dan kendala sebagai berikut:

   (a) **Variabel**
   Setiap warna pada papan diperlakukan sebagai sebuah variabel. Masing-masing variabel harus ditempatkan tepat satu menteri pada salah satu sel yang termasuk ke dalam sektor warna tersebut.

   (b) **Ruang Solusi**
   Ruang solusi untuk setiap variabel merupakan himpunan seluruh sel yang memiliki warna yang sesuai dengan variabel tersebut. Dengan demikian, setiap variabel memiliki domain yang berbeda tergantung pada posisi warna di papan.

   (c) **Kendala (Constraints)**
   Aturan permainan diterjemahkan menjadi sejumlah kendala yang harus dipenuhi secara bersamaan:

- Tidak boleh ada dua menteri yang bersebelahan secara horizontal, vertikal, maupun diagonal.
- Setiap warna hanya boleh berisi satu menteri.
- Serangan diagonal tidak dihitung, sehingga dua menteri dapat berada pada garis diagonal yang sama selama tidak bersebelahan.
- Serangan horizontal dan vertikal tetap berlaku, sehingga dua menteri tidak boleh berada pada baris atau kolom yang sama.

Contoh sederhana:

Misalkan papan 4x4 dengan 4 warna berbeda (R, B, G, Y) dan setiap warna membentuk sektor yang tersebar di papan:

```
1  R B B B
2  R R B Y
3  G Y Y Y
4  G G G Y
```

Di sini:

- Variabel: R, B, G, Y
- Domain variabel: Semua sel dengan warna yang sesuai
  - R: {(1,1), (2,1), (2,2)}
  - B: {(1,2), (1,3), (1,4), (2,3)}
  - G: {(3,1), (4,1), (4,2), (4,3)}
  - Y: {(2,4), (3,2), (3,3), (3,4), (4,4)}
- Tujuan: Menempatkan satu menteri per warna tanpa melanggar kendala

Contoh solusi yang memenuhi kendala:

```
1  - - Q -
2  Q - - -
3  - - - Q
4  - Q - -
```

Keterangan:

- 'Q' menandakan posisi menteri pada papan.
- Setiap menteri berada di sel dengan warna yang berbeda (satu per variabel/warna).
- Tidak ada dua menteri yang bersebelahan atau berada pada baris/kolom yang sama.

Dengan pemodelan CSP tersebut, seluruh aturan permainan dapat dinyatakan secara terstruktur sehingga solver dapat melakukan pemeriksaan dan pencarian solusi dengan cara yang konsisten dan sistematis.

4. **Melakukan analisis kebutuhan perangkat lunak, baik fungsional maupun non-fungsional, termasuk kebutuhan solver dan antarmuka pengguna.**
   Status : Ada sejak rencana kerja skripsi.
   Hasil :

   Melakukan analisis kebutuhan perangkat lunak, yang meliputi:

   - Kebutuhan fungsional, seperti:

  - Fitur permainan Colored Queens.
  - Fungsi solver untuk menyelesaikan permainan.
  - Peringatan visual saat terjadi pelanggaran kendala.
  - Fitur petunjuk untuk pemain yang membutuhkan bantuan.
  - Mode permainan, mode pengujian algoritma, dan visualisasi algoritma.
- Kebutuhan non-fungsional, seperti:
  - Performa: Waktu respons sistem dan efisiensi komputasi solver
  - Usability: Antarmuka intuitif dan mudah dipelajari
  - Visualisasi: Representasi grafis yang jelas untuk proses pencarian solusi
  - Skalabilitas: Kemampuan menangani berbagai ukuran papan permainan
- Kebutuhan khusus terkait solver:
  - Integrasi algoritma Backtracking dengan dan tanpa AC-3 dan PSO.
  - Mekanisme perbandingan performa antar algoritma (Waktu dan ketepatan solusi)
  - Visualisasi langkah-langkah pencarian solusi secara real-time
- Kebutuhan antarmuka pengguna:
- Antarmuka pengguna yang intuitif, meliputi:
  - Tampilan jelas dan konsisten, sehingga pengguna mudah mengenali fungsi setiap tombol atau menu.
  - Interaksi sederhana, dengan langkah-langkah minimum untuk melakukan tindakan utama (misal menempatkan menteri, memulai ulang permainan).
  - Umpan balik visual langsung, seperti peringatan warna atau highlight saat terjadi pelanggaran kendala.
  - Navigasi mudah, dengan akses cepat ke mode permainan dan mode pengujian algoritma.
  - Informasi penting terlihat tanpa perlu membuka banyak jendela atau menu.

5. **Merancang arsitektur sistem serta antarmuka pengguna untuk aplikasi Colored Queens Solver.**
   Status : Ada sejak rencana kerja skripsi.
   Hasil :

6. **Menyusun dokumentasi tugas akhir untuk tahap TA 1.**
   Status : Ada sejak rencana kerja skripsi.
   Hasil :

6 Pencapaian Rencana Kerja

Langkah-langkah kerja yang berhasil diselesaikan dalam Tugas Akhir 1 ini adalah sebagai berikut:

1. Melakukan studi literatur terkait permasalahan n-queens dan variannya, Constraint Satisfaction Problem (CSP), algoritma pencarian Backtracking, teknik metaheuristik Particle Swarm Optimization, serta metode propagasi kendala AC-3.

2. Mengumpulkan dan menyusun berbagai skenario permasalahan Colored Queens yang akan digunakan sebagai basis pengujian algoritma serta sebagai pilihan tingkat kesulitan bagi pengguna.

3. Melakukan pemodelan masalah Colored Queens ke dalam bentuk CSP agar dapat diproses oleh algoritma pencarian.

4. Melakukan analisis kebutuhan perangkat lunak, baik fungsional maupun non-fungsional, termasuk kebutuhan solver dan antarmuka pengguna.
5. Merancang arsitektur sistem serta antarmuka pengguna untuk aplikasi Colored Queens Solver.
6. Menyusun dokumentasi tugas akhir untuk tahap TA 1.

7 Kendala yang Dihadapi

Kendala - kendala yang dihadapi selama mengerjakan Tugas Akhir :

- Terlalu banyak melakukan prokratinasi
- Terlalu banyak godaan berupa hiburan (game, film, dll)
- Tugas Akhir diambil bersamaan dengan kuliah ASD karena selama 5 semester pertama kuliah tersebut sangat dihindari dan tidak diambil, dan selama 4 semester terakhir kuliah tersebut selalu mendapat nilai E
- Mengalami kesulitan pada saat sudah mulai membuat program komputer karena selama ini selalu dibantu teman

Bandung, 19/11/2025

Arlo Dante Hananvyasa

Menyetujui,

Nama: Husnul Hakim, M.T.
Pembimbing Tunggal