IF2211 Strategi Algoritma

Tugas Kecil 1: Penyelesaian Permainan Queens LinkedIn

Disusun oleh

Ni Made Sekar Jelita (18223101)

Kelas K1

FAKULTAS SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2026

DAFTAR ISI

DAFTAR ISI.....1
1. Algoritma Brute Force..... 2
2. Penjelasan Source Program..... 2
3. Hasil Program..... 8
4. Pranala Repository..... 10
5. Pernyataan..... 11
Lampiran..... 12

1. Algoritma Brute Force

Algoritma brute force adalah algoritma yang mencari solusi secara langsung tanpa memanfaatkan heuristik. Algoritma ini mencoba segala kemungkinan solusi dan sangat bergantung pada *computing power*. Secara garis besar, berikut adalah karakteristik brute force.

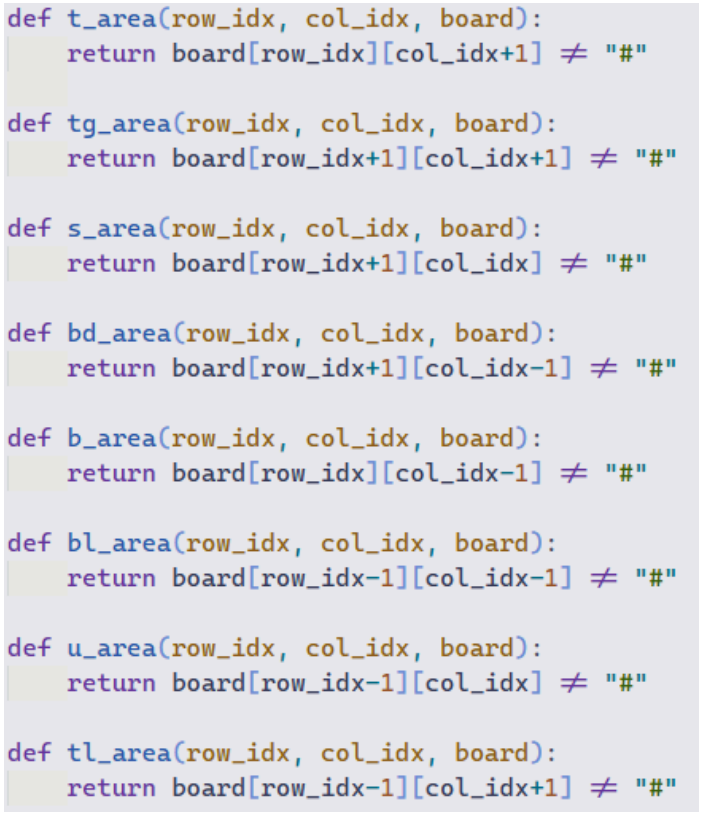
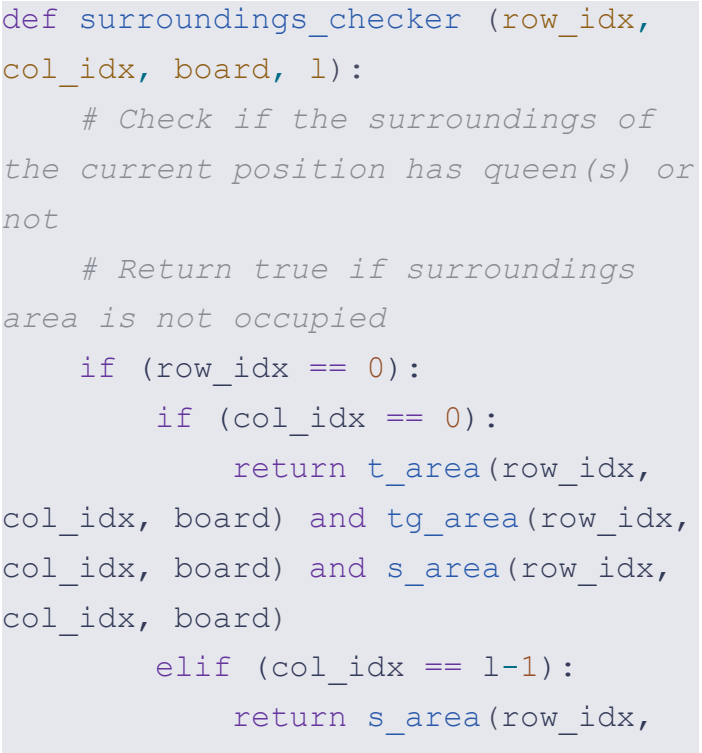
a. Tidak cerdas
b. Memerlukan usaha yang besar (*computing power*)
c. Sederhana atau mudah dipahami
d. Mudah diimplementasikan
e. Menjamin menemukan solusi persoalan

2. Penjelasan *Source* Program

Function/potongan program	Penjelasan

`def column_checker(column, board, l):`
`  for i in range(l):`
`  if (board[i][column] == "#"):`
`  return False`
`  return True`
	Function untuk memeriksa apakah column pada idx column sudah memiliki queens atau belum. Mengembalikan True jika belum.

`def row_checker(row, board, l):`
`  for i in range(l):`
`  if (board[row][i] == "#"):`
`  return False`
`  return True`
	Function untuk memeriksa apakah row pada idx row sudah memiliki queens atau belum. Mengembalikan True jika belum.

	Function-function untuk memeriksa apakah area sesuai kode (tg: tenggara) sudah memiliki queens atau belum. Mengembalikan True jika belum
	Function yang memanggil function-function sebelumnya, mengembalikan true jika surroundings area belum ditempati queens

```
col_idx, board) and bd_area(row_idx,
col_idx, board) and b_area(row_idx,
col_idx, board)
        else:
            return b_area(row_idx,
col_idx, board) and bd_area(row_idx,
col_idx, board) and s_area(row_idx,
col_idx, board) and tg_area(row_idx,
col_idx, board) and t_area(row_idx,
col_idx, board)
    elif (row_idx == l-1):
        if (col_idx == 0):
            return u_area(row_idx,
col_idx, board) and tl_area(row_idx,
col_idx, board) and t_area(row_idx,
col_idx, board)
        elif (col_idx == l-1):
            return u_area(row_idx,
col_idx, board) and bl_area(row_idx,
col_idx, board) and b_area(row_idx,
col_idx, board)
        else:
            return b_area(row_idx,
col_idx, board) and bl_area(row_idx,
col_idx, board) and u_area(row_idx,
col_idx, board) and tl_area(row_idx,
col_idx, board) and t_area(row_idx,
col_idx, board)
    else:
        if (col_idx == 0):
            return u_area(row_idx,
col_idx, board) and tl_area(row_idx,
```

col_idx, board) and t_area(row_idx, col_idx, board) and tg_area(row_idx, col_idx, board) and s_area(row_idx, col_idx, board)
elif (col_idx == l-1):
return u_area(row_idx, col_idx, board) and bl_area(row_idx, col_idx, board) and b_area(row_idx, col_idx, board) and bd_area(row_idx, col_idx, board) and s_area(row_idx, col_idx, board)
else:
return bl_area(row_idx, col_idx, board) and u_area(row_idx, col_idx, board) and tl_area(row_idx, col_idx, board) and t_area(row_idx, col_idx, board) and tg_area(row_idx, col_idx, board) and s_area(row_idx, col_idx, board) and bd_area(row_idx, col_idx, board) and b_area(row_idx, col_idx, board) and bl_area(row_idx, col_idx, board)	

def check_queen(color, colors_used, row_idx, column_idx, board, l):
if (colors_used[color] == True):
return False
return column_checker(column_idx, board, l) and row_checker(row_idx, board, l)
	Function yang menggabungkan pemeriksaan queen di kolom, baris, dan warna. Mengembalikan True jika queen memenuhi syarat

`and surroundings_checker(row_idx, column_idx, board, l)`	

`l = len(chicken)`
 	
`palette = set()`
`for row in chicken:`
`for color in row:`
`palette.add(color)`
 	
`colors_used = {warna: False for warna in palette}`
 	
`init_board = copy.deepcopy(chicken)`
 	
`iterazi = 0`
`row = 0`
`col = 0`
`queen_throne_col = [-1] * l`
	Warna-warna yang ada pada board, ditaruh pada set palette. Setelah itu, pemeriksaan apakah warna tersebut sudah memiliki queen atau belum dilakukan dengan menggunakan dictionary colors_used, True jika queen sudah ditaruh.
	
	Queen_throne_col digunakan untuk menandakan posisi queen yang sudah dipasang ada di kolom berapa

`for j in range(col, l):`
`iterazi += 1`
`current_color = init_board[row][j]`
`if check_queen(current_color, colors_used, row, j, chicken, l):`
`chicken[row][j] = "#"`
 	
`colors_used[current_color] = True`
	Iterasi dari row indeks 0 hingga sepanjang board.
	
	Pada for loop, dilakukan pengecekan apakah queen memenuhi syarat untuk ditaruh atau tidak. Jika iya,
	- posisi tersebut diberi mark “#”
	- Dictionary colors_used diperbarui menjadi True pada warna terkait
	- queen_throne_col diperbarui dengan indeks j (kolom) saat ini
	- Break loop supaya langsung ke kolom selanjutnya
	
	Jika ratu sudah berhasil dipasang (throned),

```
queen_throne_col[row] = j
                throned = True
                break
        if throned:
            row += 1
            col = 0
        else:
            row -= 1
            if row < 0:
                print("Solusi tidak
ditemukan")
                print("\n")

print(r"_\|/__\|/__\|/__\|/__\|/__\|
/__\|/__\|/__\|/__\|/__\|/__\|/_")
                print(r"_\|/
__\|/_")
                print(r"_\|/
CORONATION INTERRUPTED
__\|/_")
                print(r"_\|/
__\|/_")

print(r"_\|/__\|/__\|/__\|/__\|/__\|
/__\|/__\|/__\|/__\|/__\|/__\|/_")
                print("\n")
                return

            # start lagi dari tempat
queen di iterasi terakhir
            prev_col =
```

tambah indeks row (+1) untuk melanjutkan ke baris selanjutnya

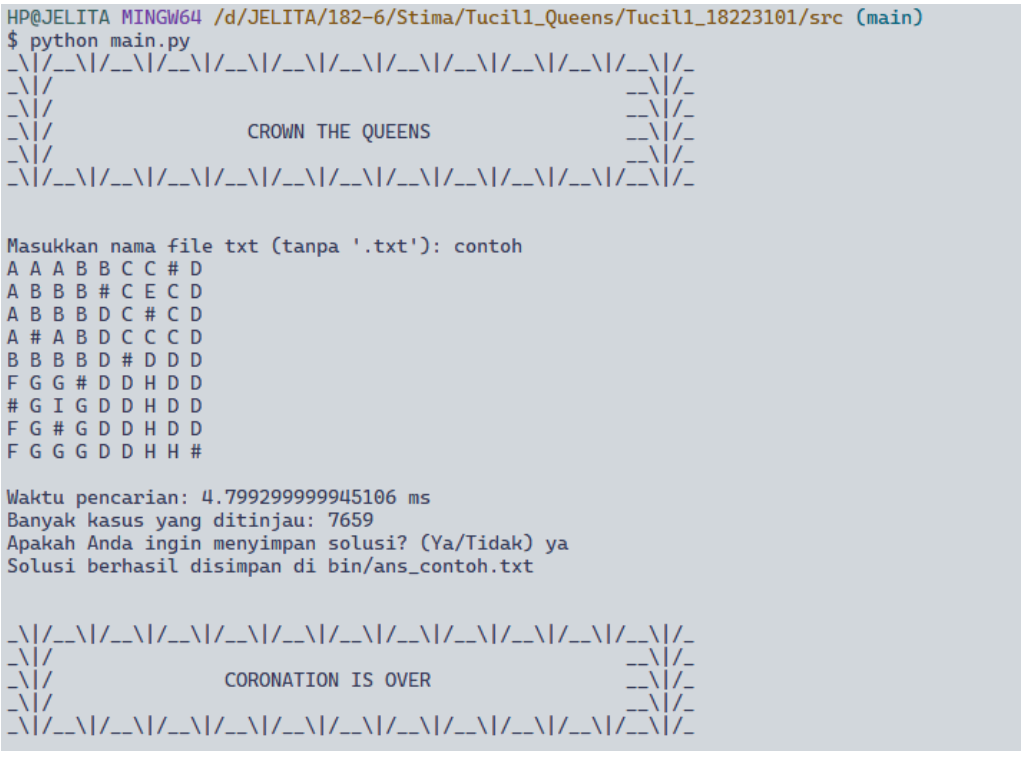
Jika belum,
(di sini dilakukan *backtracking*)
Posisi dikembalikan ke indeks pada baris sebelumnya (row -=1)

Prev_col menyimpan queen yang terakhir kali ditaruh
Prev_color menyimpan warna asli posisi yang ditempati queen

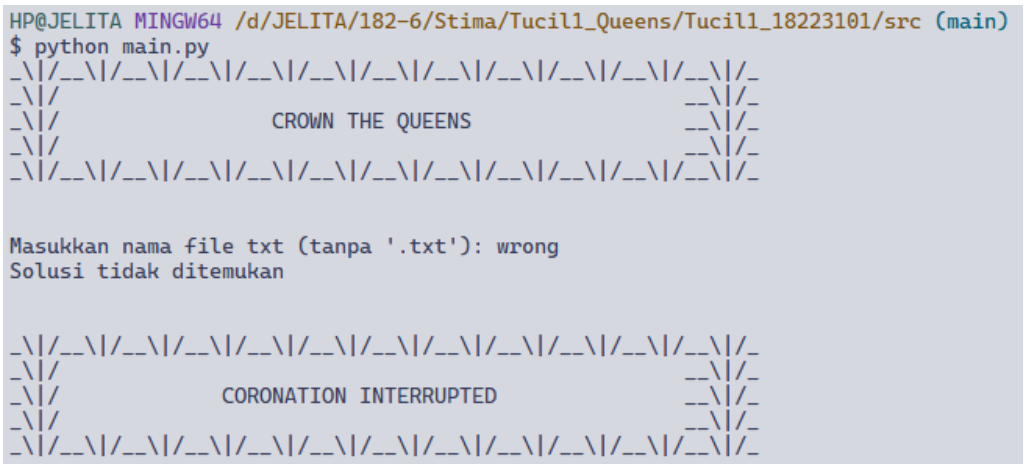
Kemudian, mark (“#”) diganti kembali dengan warna sebelumnya dan while loop diulang dari awal dengan row yang sama (karena belum berhasil menempatkan queen di row tersebut.)

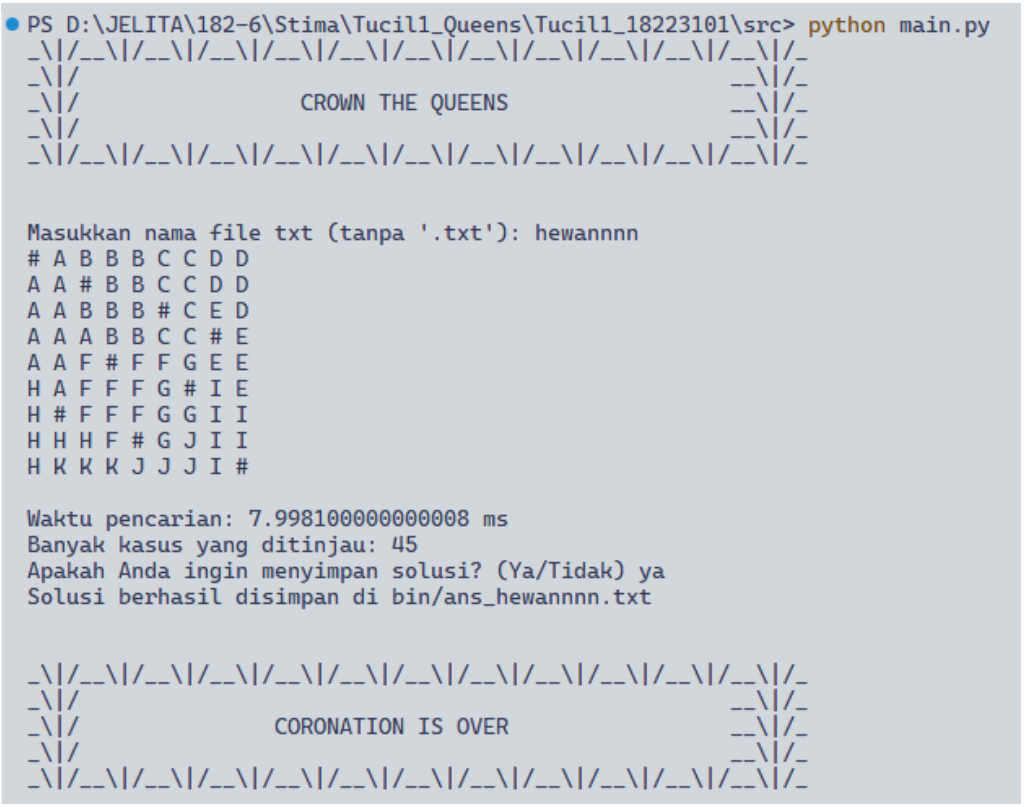
```
queen_throne_col[row]
            prev_color =
init_board[row][prev_col]
            chicken[row][prev_col] =
prev_color
            colors_used[prev_color]
= False


            col = prev_col + 1
```

3. Hasil Program

No	File .txt	Hasil screenshot
1.	Contoh.txt (test case dari spesifikasi	

2.	dinobob4.txt (community level)	HP@JELITA MINGW64 /d/JELITA/182-6/Stima/Tucil1_Queens/Tucil1_18223101/src (main) $ python main.py _|/__|/__|/__|/__|/__|/__|/__|/__|/__|/__|/__|/_ _|/ __|/_ _|/ CROWN THE QUEENS __|/_ _|/ __|/_ _|/__|/__|/__|/__|/__|/__|/__|/__|/__|/__|/__|/_ Masukkan nama file txt (tanpa '.txt'): dinobob4 A A A B # C A A # B B C D A B B C # D # F F C D D E E # D D # D D D D D Waktu pencarian: 1.1622999999998385 ms Banyak kasus yang ditinjau: 633 Apakah Anda ingin menyimpan solusi? (Ya/Tidak) ya Solusi berhasil disimpan di bin/ans_dinobob4.txt _|/__|/__|/__|/__|/__|/__|/__|/__|/__|/__|/__|/_ _|/ __|/_ _|/ CORONATION IS OVER __|/_ _|/ __|/_ _|/__|/__|/__|/__|/__|/__|/__|/__|/__|/__|/__|/_
3.	queens_1802.txt (LinkedIn level 18022026)	HP@JELITA MINGW64 /d/JELITA/182-6/Stima/Tucil1_Queens/Tucil1_18223101/src (main) $ python main.py _|/__|/__|/__|/__|/__|/__|/__|/__|/__|/__|/__|/_ _|/ __|/_ _|/ CROWN THE QUEENS __|/_ _|/ __|/_ _|/__|/__|/__|/__|/__|/__|/__|/__|/__|/__|/__|/_ Masukkan nama file txt (tanpa '.txt'): queens_1802 A A A # A B B A # A A B B D A A A E # F D E E # E F F D E G E F F # D # G F F F D D G D D D D D # Waktu pencarian: 0.8347999999998024 ms Banyak kasus yang ditinjau: 749 Apakah Anda ingin menyimpan solusi? (Ya/Tidak) ya Solusi berhasil disimpan di bin/ans_queens_1802.txt _|/__|/__|/__|/__|/__|/__|/__|/__|/__|/__|/__|/_ _|/ __|/_ _|/ CORONATION IS OVER __|/_ _|/ __|/_ _|/__|/__|/__|/__|/__|/__|/__|/__|/__|/__|/__|/_

4.	korokira.txt (community level)	
5.	wrong.txt	

6.	Hewannnn.txt (community level)	

4. Pranala Repository

Repository Github	https://github.com/ielita/Tucil1_18223101.git
Google Drive	Deliverable

5. Pernyataan

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

Ni Made Sekar Jelita Parameswari

Lampiran

[1] “Brute Force Algorithms Explained,” freeCodeCamp.org. Accessed: Feb. 18, 2026. [Online]. Available: https://www.freecodecamp.org/news/brute-force-algorithms-explained/
[2] “Python Tutorial.” [Online]. Available: http://www.w3schools.com/python/
[3] “The Python Standard Library — Python 2.7.18 documentation.” [Online]. Available: https://docs.python.org/2/library/