

Deskripsi Singkat Tugas (dari Spesifikasi)

Queens adalah gim logika yang tersedia pada situs jejaring profesional LinkedIn. Tujuan dari gim ini adalah menempatkan queens pada sebuah papan persegi berwarna sehingga terdapat hanya satu queen pada tiap baris, kolom, dan daerah warna. Selain itu, satu queen tidak dapat ditempatkan bersebelahan dengan queens lainnya, termasuk secara diagonal.

Tugas anda adalah membuat program yang dapat menemukan satu solusi penempatan queens pada suatu papan berwarna yang diberikan, atau menampilkan bahwa tidak ada solusi yang valid. Program melakukan pencarian solusi menggunakan algoritma brute force.

Penjelasan Algoritma

Algoritma brute force yang dipakai menggunakan bahasa python melalui kombinasi posisi queens di setiap wilayah.

function load_board(filename: file) → list of list of char

KAMUS LOKAL

f : SEQFILE of string

line : string

ALGORITMA

assign(f, "filename")

with open(f): {langsung close setelah selesai}

→ [list(line.strip()) for line in f]

procedure save_board(input board: list of list of char, input queens: list of (integer, integer))

KAMUS LOKAL

f : SEQFILE of string

line : string

n, r, c : integer

ALGORITMA

assign(f, "test/output.txt")

rewrite(f)

with open(f): {langsung close setelah selesai}

n ← len(board)

for r in range(n) do

line ← ""

for c in range(len(board[r])) do

if ((r, c) in queens) do

line ← line + "#"

else

line ← board[r][c]

write(f, line)

function build_regions(board: list of list of char) → dictionary

{key: char ; values: list of (integer, integer)}

KAMUS LOKAL

regions : dictionary {key: char ; values: list of (integer, integer)}

n, r, c : integer

line : string

ALGORITMA

regions \leftarrow defaultdict(list) {membuat value default kalau key belum ada}

n \leftarrow len(board)

for r in range(n) do

for c in range(len(board[r])) do

 regions[board[r][c]] \leftarrow (r, c)

\rightarrow regions

function is_valid(positions: list of (integer, integer)) \rightarrow boolean

KAMUS LOKAL

i, j, r1, c1, r2, c2 : integer

line : string

ALGORITMA

for i in range(len(positions)) do

 r1, c1 \leftarrow positions[i]

for j in range(i+1, len(positions)) do

 r2, c2 \leftarrow positions[j]

if (r1 = r2) then

\rightarrow False

if (c1 = c2) then

\rightarrow False

if (abs(r1 - r2) <= 1) and (abs(c1 - c2) <= 1) then

\rightarrow False

\rightarrow True

procedure print_board(input board: list of list of char, input queens: list of (integer, integer))

KAMUS LOKAL

line : string

n, r, c : integer

ALGORITMA

n \leftarrow len(board)

for r in range(n) do

 line \leftarrow ""

for c in range(len(board[r])) do

if ((r, c) in queens) do

 line \leftarrow line + "#"

else

 line \leftarrow board[r][c]

output(line)

output()

procedure main()

{Prosedur program utama}

KAMUS LOKAL

board : list of list of char

regions : dictionary {key: char ; values: list of (integer, integer)}

region_lists : list of list of (integer, integer)

iteration : integer

solution, combination : list of (integer, integer)

start, last_print_time, now, end : real

ALGORITMA

if len(sys.argv != 2) then {format menjalankan program}

output("Format input salah")

→ {akhiri prosedur}

board ← load_board(sys.argv[1])

regions ← build_regions(board)

region_lists ← list(regions.values())

iteration ← 0

solution ← NULL

start ← time.perf_counter()

last_print_time ← start

for combination in product(*region_lists) do

{product menghasilkan semua kombinasi dalam region lists}

iteration ← iteration + 1

now ← time.perf_counter()

if (now - last_print_time) >= 0.3 then

output("Kombinasi ke-", iteration)

print_board(board, combination)

last_print_time ← now

if is_valid(combination) then

solution ← combination

break

end ← time.perf_counter()

output("=====")

if solution then {tidak NULL}

output("Solusi ditemukan:")

print_board(board, solution)

save_board(board, solution)

else

output("Solusi tidak ditemukan!")

output("Total kombinasi yang dicek:", iteration)

output("Waktu pencarian:", round((end - start) * 1000, 3), "ms")

{pembulatan 3 angka di belakang koma}

Pada awal prosedur main, program akan membaca input mulai program, jika tidak sesuai format, maka akan diberitahu melalui terminal bahwa format input salah. Jika input sudah benar, program akan membuat list dari semua posisi kemungkinan queens yang dipisahkan sesuai dengan wilayahnya. Cara mendapatkan list posisi tersebut adalah dengan mengambil tiap karakter yang dipisahkan per baris (berbentuk list of list of char) dari file input, kemudian diubah menjadi bentuk dictionary (key adalah wilayah, sedangkan values berupa list posisi wilayah tersebut), dan akhirnya diubah menjadi bentuk list of list char yang berurutan sesuai wilayahnya (misal, list pertama dari list adalah wilayah A, list yang kedua adalah wilayah B, dan seterusnya).

Menggunakan fungsi product dari itertools, akan menciptakan kombinasi posisi queens dari masing-masing wilayah (ini adalah fase brute force nya, yaitu berdasarkan kombinasi per wilayah). Tiap kombinasi posisi queens tersebut akan diperiksa satu per satu apakah memenuhi aturan permainan queens (tidak satu baris, kolom, dan berdekatan), selain itu, program akan terus menghitung iterasi kombinasi dan waktu yang dipakai, tiap 0.3 detik, program akan print sedang mencoba kombinasi ke berapa serta tampilan kombinasinya di dalam terminal. Jika ditemukan kombinasi yang memenuhi aturan permainan queens tersebut, maka program akan membuat atau memperbarui file output.txt dengan hasil solusi tersebut. Jika tidak, maka program tidak akan memunculkan solusi apa-apa. Kedua jenis hasil ini akan dibarengi dengan jumlah kombinasi yang telah dilakukan serta waktu pencarian yang dipakai di dalam terminal.

Sebagai tambahan kecil, program sudah diatur agar bisa mengambil bentuk ukuran matriks yang beragam. Namun karena queens ditandai menggunakan '#' dalam program, disarankan untuk tidak menggunakan karakter tersebut dalam input wilayah untuk mencegah error.

Source Program

```
from collections import defaultdict
from itertools import product
import time
import sys

def load_board(filename):
    with open(filename, 'r') as f:
        return [list(line.strip()) for line in f]

def save_board(board, queens):
    with open("test/output.txt", "w") as f:
        n = len(board)
        for r in range(n):
            line = ""
            for c in range(len(board[r])):
```

```

        if (r, c) in queens:
            line += "#"
        else:
            line += board[r][c]
        f.write(line.strip() + "\n")

def build_regions(board):
    regions = defaultdict(list)
    n = len(board)
    for r in range(n):
        for c in range(len(board[r])):
            regions[board[r][c]].append((r, c))
    return regions

def is_valid(positions):
    for i in range(len(positions)):
        r1, c1 = positions[i]
        for j in range(i + 1, len(positions)):
            r2, c2 = positions[j]
            if r1 == r2:
                return False
            if c1 == c2:
                return False
            if (abs(r1 - r2) <= 1) and (abs(c1 - c2) <= 1):
                return False
    return True

def print_board(board, queens):
    n = len(board)
    for r in range(n):
        line = ""
        for c in range(len(board[r])):
            if (r, c) in queens:
                line += "#"
            else:
                line += board[r][c]
        print(line)
    print()

```

```

def main():
    if len(sys.argv) != 2:
        print("Format input salah")
        return

    board = load_board(sys.argv[1])
    regions = build_regions(board)
    region_lists = list(regions.values())

    iteration = 0
    solution = None

    start = time.perf_counter()
    last_print_time = start

    for combination in product(*region_lists):
        iteration += 1
        now = time.perf_counter()

        if now - last_print_time >= 0.3:
            print(f"\nKombinasi ke-{iteration}")
            print_board(board, combination)
            last_print_time = now

        if is_valid(combination):
            solution = combination
            break

    end = time.perf_counter()

    print("\n=====")
    if solution:
        print("Solusi ditemukan:")
        print_board(board, solution)
        save_board(board, solution)
    else:
        print("Solusi tidak ditemukan!")

    print("Total kombinasi yang dicek:", iteration)
    print("Waktu pencarian:", round((end - start) * 1000, 3), "ms")

```

```
if __name__ == "__main__":  
    main()
```

Contoh Input dan Output (input.txt, output.txt jika ada solusi, dan terminal)

1. Testcase 1

test > ≡ input.txt	test > ≡ output.txt
1 AAAD	1 AA#D
2 BACD	2 ■ #ACD
3 CCCC	3 CCC#
4 DDDD	4 D#DD

```
=====
Solusi ditemukan:
AA#D
#ACD
CCC#
D#DD

Total kombinasi yang dicek: 80
Waktu pencarian: 0.03 ms
```

2. Testcase 2

test > ≡ input.txt	
1 AABCCD	
2 AEBCCD	
3 EEBFDD	
4 GGBFDD	
5 GGHFII	
6 HHHFII	

=====
Solusi tidak ditemukan!
Total kombinasi yang dicek: 221184
Waktu pencarian: 57.498 ms

3. Testcase 3

<pre>test > ≡ input.txt 1 AABBBCCD 2 AEBBBCCD 3 AEBFFCGD 4 HEEFFCGD 5 HEEFFGGD 6 HIIJGGGD 7 HIIJKKKL 8 HIJJKKKL</pre>	<pre>Kombinasi ke-303916336 AABBBCCD A#BBBCCD #E##FCGD HEEFF#G# HEEFFG#D H#IJGGGD #II#K#KL HIJJKKK# ===== Solusi tidak ditemukan! Total kombinasi yang dicek: 304819200 Waktu pencarian: 96177.848 ms</pre>
--	--

4. Testcase 4

<pre>test > ≡ input.txt 1 AAAAAAA 2 AABACCA 3 ABBDDCC 4 ABEEDDC 5 AAEECC 6 AAFFGGC 7 AAFFGG</pre>	<pre>test > ≡ output.txt 1 ■ #AAAAAA 2 AABAC#A 3 ABB#DCC 4 A#EEDDC 5 AAEE#CC 6 AA#FFGC 7 AAFFG#</pre>
---	---


```

=====
Solusi ditemukan:
#AAAAAA
AABAC#A
ABB#DCC
A#EEDDC
AAAE#CC
AA#FGGC
AAAFFG#

Total kombinasi yang dicek: 6452
Waktu pencarian: 4.018 ms

```

5. Testcase 5

test > ≡ input.txt	test > ≡ output.txt
1 AAABBCCCD	1 AAABBCC#D
2 AB BBBCECD	2 AB BB#CECD
3 AB BBDCECD	3 AB BBD C#CD
4 AAABDCCCD	4 A#ABDCCCD
5 BBBBDDDDD	5 BBBBD#DDD
6 FGGGDDHDD	6 FGG#DDHDD
7 FGIGDDHDD	7 #GIGDDHDD
8 FGIGDDHDD	8 FG#GDDHDD
9 FGGGDDHHH	9 FGGGDDHH#

Kombinasi ke-21207291	=====
AAABBCCCD	Solusi ditemukan:
ABB#B#ECD	AAABBCC#D
ABBBDC#CD	ABBB#CECD
A#ABDCCCD	ABBBDC#CD
BBBBDDDDD	A#ABDCCCD
FGGGDDHDD	BBBBD#DDD
FG###D#DD	FGG#DDHDD
FGIGDDHDD	#GIGDDHDD
#GGGDDHHH	FG#GDDHDD
	FGGGDDHH#
	Total kombinasi yang dicek: 21415356
	Waktu pencarian: 6692.161 ms

Link Repositori

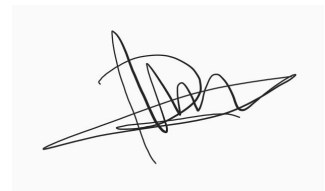
Github: https://github.com/RND815/Tucil1_13524111

Lampiran

No	Poin	Ya	Tidak
1	Program berhasil di kompilasi tanpa kesalahan	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2	Program berhasil dijalankan	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3	Solusi yang diberikan program benar dan mematuhi aturan permainan	<input checked="" type="checkbox"/>	<input type="checkbox"/>
4	Program dapat membaca masukan berkas .txt serta menyimpan solusi dalam berkas .txt	<input checked="" type="checkbox"/>	<input type="checkbox"/>
5	Program memiliki Graphical User Interface (GUI)	<input type="checkbox"/>	<input checked="" type="checkbox"/>
6	Program dapat menyimpan solusi dalam bentuk file gambar	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Pernyataan

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



Reynard Anderson Wijaya