LAPORAN TUGAS KECIL 1

IF2211 STRATEGI ALGORITMA

Disusun oleh :

Muhamad Hasbullah Faris 18223014

PROGRAM TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

TAHUN AJARAN 2025/2026

Daftar Isi

Daftar Isi **2**
Permasalahan **3**
Pendekatan Solusi **3**
Implementasi Solusi **3**
Inisialisasi Permasalahan 3
Eksplorasi Solusi 4
Penyimpanan Solusi 6
Fungsi Print Iterasi 7
Fungsi Validasi Aturan Baris dan Kolom 7
Fungsi Validasi Aturan Warna 8
Fungsi Menyimpan Solusi 8
Pengujian Test Case **9**
Pengujian 1 (Memiliki Solusi) 9
Pengujian 2 (Memiliki Solusi) 9
Pengujian 3 (Memiliki Solusi) 10
Pengujian 4 (Memiliki Solusi) 10
Pengujian 5 (Tidak Memiliki Solusi) 11
Pranala Repository **11**
Pernyataan Kejujuran **11**
Lampiran **11**

Permasalahan

LinkedIn Queens merupakan permainan yang tersedia pada aplikasi LinkedIn. Permainan ini mengharuskan pemain untuk menempatkan queens pada sebuah papan persegi berwarna, sedemikian rupa sehingga terdapat hanya satu queen pada tiap baris, kolom, dan daerah warna.

Pendekatan Solusi

Pendekatan yang digunakan untuk menyelesaikan **LinkedIn Queens** adalah dengan menggunakan algoritma bruteforce, yaitu pendekatan yang akan mencoba semua kemungkinan penempatan queen hingga ditemukan solusi yang memenuhi syarat.

Algoritma bruteforce akan mencoba menempatkan queen satu persatu hingga terdapat n queen pada sebuah papan berukuran n^2, lalu memvalidasi apakah solusi tersebut memenuhi semua aturan. Jika tidak tervalidasi, algoritma akan mencoba konfigurasi solusi lain hingga ditemukan solusi yang memenuhi semua aturan. Pendekatan ini memastikan bahwa solusi yang ditemukan adalah solusi yang valid.

Implementasi Solusi

Inisialisasi Permasalahan

```
file_path = input("Masukkan file permasalahan: ")
file = open(file_path).read()
problem = list(file.replace("\n", ""))

length = len(file.splitlines())
solution = []
solution_found = False
```

Pertama-tama, dilakukan inisialisasi data utama dengan menerima lokasi file .txt input dari pengguna pada variabel file_path. File tersebut kemudian dibaca dan direpresentasikan menjadi list linear problem dengan panjang n^2.

Selanjutnya, length menyimpan ukuran papan ukuran n, yang dihitung dari jumlah baris pada file. Variabel solution disiapkan sebagai tempat menyimpan solusi, jika solusi ditemukan, sedangkan variabel solution_found berfungsi sebagai penanda untuk mengetahui apakah solusi berhasil ditemukan atau tidak.

Eksplorasi Solusi

```
total_cells = length ** 2
idx = list(range(length))
case = 0

start = time.time() * 1000 # Start time

if len(problem) != (length ** 2):
    print("Input tidak valid!")
else:
    while True:
        case += 1
        if case % 1000 == 0:
            print_input(copy, length)

        copy = problem[:]
        for p in idx:
            copy[p] = '#'

        # Validate
        if val_col_row(copy, length) and val_color(problem, copy):
            print("Solusi ditemukan:")
            print_input(copy, length)
            solution = copy
            solution_found = True
            break

        i = length - 1
        while i >= 0 and idx[i] == i + (total_cells - length):
            i -= 1
```

```
        if i < 0:
            break

        idx[i] += 1
        for j in range(i + 1, length):
            idx[j] = idx[j - 1] + 1

end = time.time() * 1000 # End time
```

Sebelum dilakukan eksplorasi, dilakukan inisiasi variabel total_cells menyimpan luas papan yaitu n^2, dan idx adalah daftar indeks awal yang berisi n posisi pertama sebagai kandidat penempatan queen. Variabel case dipakai untuk menghitung berapa banyak konfigurasi yang sudah diuji.

Proses eksplorasi dilakukan dengan mencoba seluruh konfigurasi penempatan queen menggunakan loop while yang dimulai dari idx. Pada setiap iterasi, nilai case bertambah untuk mencatat banyaknya konfigurasi yang telah diperiksa, dibuat juga salinan dari problem dalam variabel copy, lalu p pada idx di-*assign* dengan # sebagai penanda queen.

Konfigurasi ini kemudian divalidasi oleh fungsi val_col_row yang memastikan setiap baris dan kolom tepat memiliki 1 queen, dan fungsi val_color yang memastikan tidak ada queen yang menempati daerah warna yang sama. Jika kedua kondisi terpenuhi, loop akan berhenti dan konfigurasi akan disimpan ke dalam variabel solusi, dan variabel solution_found akan di-set nilainya ke True.

Apabila konfigurasi tidak valid, akan dicoba konfigurasi berikutnya, dimulai dari index terakhir pada idx. Index terakhir ini di-*assign* ke variabel i, lalu diperiksa pada loop while i >= 0 and idx[i] == i + (total_cells - length) untuk mengetahui apakah idx[i] sudah mencapai ujung papan (length). Jika sudah, nilai i dikurangi untuk memeriksa elemen pada index i - 1 (sebelah kiri).

Jika setelahnya nilai i < 0, maka seluruh kombinasi sudah dicoba dan loop berhenti. Jika belum, maka dilakukan pencarian hingga nilai i < 0. idx[i] dinaikkan satu (idx[i] +=

1), lalu semua elemen setelah i di-reset dengan aturan idx[j] = idx[j - 1] + 1 agar tetap berurutan dan membentuk kombinasi terkecil berikutnya.

Penyimpanan Solusi

```
if solution_found == False:
    print("Solusi tidak ditemukan.")

print(f"Banyak kasus yang ditinjau: {case}")
print(f"Waktu pencarian: {(end - start):.2f}ms")

if solution_found == True:
    choice = input("Apakah Anda ingin menyimpan solusi? (Ya/Tidak): ").lower()

    if choice == "ya":
        file_name = input("Masukkan nama file solusi (Contoh: solution.txt): ")
        save_solution(solution, length, file_name)
        print(f"Solusi disimpan pada {file_name}.")
        print("Program selesai.")
    elif choice == "tidak":
        print("Program Selesai.")
    else:
        print("Input tidak valid!")
```

Setelah eksplorasi selesai, program akan melakukan print banyak kasus yang ditinjau dan waktu pencarian (eksplorasi) dengan menghitung selisih variabel end dan start. Apabila solusi ditemukan (ditandai dengan solution_found bernilai True), maka pengguna akan mendapatkan opsi untuk menyimpan solusi. Sebaliknya, apabila solusi tidak ditemukan, maka program akan memberi tahu pengguna bahwa solusi tidak ditemukan.

Fungsi Print Iterasi

```
def print_input(input, length):
    for i in range(length):
```

```
        row = input[length * i: length * (i + 1)]
        print(" ".join(row))
    print()
```

Fungsi print_input bertujuan untuk melakukan print terhadap konfigurasi solusi. Fungsi ini menerima parameter berupa variabel input, yaitu konfigurasi solusi yang ingin di print dan length, yaitu panjang baris dari konfigurasi solusi. Fungsi ini akan melakukan slicing list input sepanjang ukuran length (baris), lalu melakukan *print* untuk masing-masing barisnya.

Fungsi Validasi Aturan Baris dan Kolom

```
def val_col_row(solution, length):
    row = [0] * length
    col = [0] * length
    for i in range(len(solution)):
        if solution[i] == "#":
            row[i // length] += 1
            col[i % length] += 1
    return all(x == 1 for x in row) and all(x == 1 for x in col)
```

Fungsi val_col_row bertujuan untuk melakukan validasi terhadap aturan permainan yang mengharuskan hanya ada 1 queen pada setiap baris dan kolom papan permainan. Fungsi ini menerima parameter berupa variabel solution, yaitu konfigurasi solusi yang ingin divalidasi dan length, yaitu panjang baris dari konfigurasi solusi. Fungsi ini akan membuat list kosong row dan cow yang berisi nilai 0 sepanjang ukuran length. Validasi dilakukan dengan melakukan iterasi pengecekan untuk setiap elemen pada solution. Apabila ditemukan solution[i] dengan nilai “#”, maka posisi baris dan kolom tersebut akan ditandai dengan menambahkan nilai row[i // length] dan col[i % length]. Apabila setelah iterasi selesai semua nilai dari row dan col adalah 1, maka konfigurasi solusi valid dan fungsi akan mengembalikan nilai True.

Fungsi Validasi Aturan Warna

```
def val_color(problem, solution):
    color = set()
    for i in range(len(solution)):
        if solution[i] == "#":
            if problem[i] in color:
                return False
            color.add(problem[i])
    return True
```

Fungsi val_color bertujuan untuk melakukan validasi terhadap aturan permainan yang mengharuskan hanya ada 1 queen pada setiap zona warna. Fungsi ini menerima parameter berupa variabel problem, yaitu permasalahan yang diberikan oleh pengguna dan solution, yaitu konfigurasi solusi yang ingin divalidasi. Fungsi ini akan membuat set kosong color. Validasi akan dilakukan dengan melakukan iterasi pengecekan pada setiap elemen pada solution. Apabila ditemukan solution[i] dengan nilai “#”, maka akan ditambahkan elemen dari problem dengan posisi yang sama dengan solution yang sedang diperiksa (problem[i]) ke dalam set color. Apabila sudah ada elemen yang sama, maka 1 daerah warna sudah ditempati 2 kali, dan fungsi akan mengembalikan nilai False. Jika sebaliknya, maka fungsi akan mengembalikan nilai True.

Fungsi Menyimpan Solusi

```
def save_solution(solution, length, filename):
    f = open(filename, "w")
    for i in range(length):
        row = solution[length * i: length * (i + 1)]
        f.write("".join(row) + "\n")
    f.close()
```

Fungsi save_solution bertujuan untuk menyimpan solusi kedalam sebuah file .txt. Fungsi ini menerima parameter berupa variabel solution, yaitu konfigurasi solusi yang ingin disimpan, length, yaitu panjang baris dari konfigurasi solusi, dan filename yaitu nama file solusi. Sama seperti fungsi print_input, fungsi ini melakukan slicing terhadap

solution sepanjang length dan disimpan pada variabel row, lalu menuliskan row satu persatu ke dalam file solusi.

Pengujian Test Case

Pengujian 1 (Memiliki Solusi)

Input	Output
AAABBB AAABBB CCCDDD CCCDDD EEEFFF EEEFFF	# A A B B B A A A # B B C # # # D # C C C D D D E E E F F F E E E F F F # A A B B B A A A # B B C # C D D D C C # D # D E E E F F F # E E F F F Solusi ditemukan: # A A B B B A A A # B B C # C D D D C C C D # D E E # F F F E E E F F # Banyak kasus yang ditinjau: 251215 Waktu pencarian: 1319.80ms Apakah Anda ingin menyimpan solusi? (Ya/Tidak):
Banyak kasus yang ditinjau: 251215	
Waktu pencarian: 1319.80 ms	

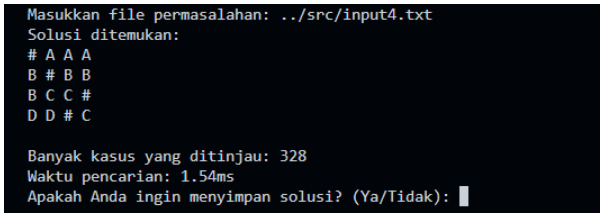
Pengujian 2 (Memiliki Solusi)

Input	Output
AAAAAA ABBBBA ACCCBD ACCBBD AECCBD AEEFFF	# A A A A A A B # B B A A C # C B # A C C # B D # E C C B D A E E F F F # A A A A A A B # B B A A C C # # D A C C B # D # E C C B D A E E F F F Solusi ditemukan: # A A A A A A B # B B A A C C # B D A C C B B # A # C C B D A E E F # F Banyak kasus yang ditinjau: 238791 Waktu pencarian: 1303.03ms Apakah Anda ingin menyimpan solusi? (Ya/Tidak):
Banyak kasus yang ditinjau: 238791	
Waktu pencarian: 1303.03ms	

Pengujian 3 (Memiliki Solusi)

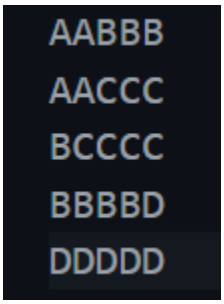
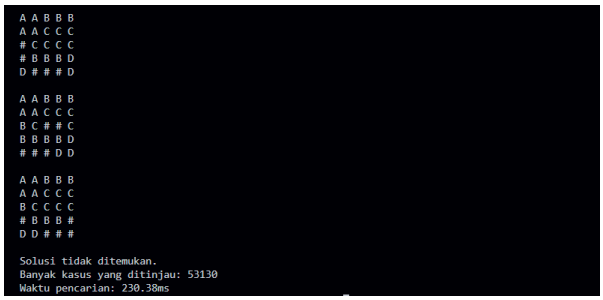
Input	Output
AAAAAAA BBBBBBB BCBBDDD CCBBDEE FCBBDDD FCBBDGD CCCBDDD	
Banyak kasus yang ditinjau: 20098741	
Waktu pencarian: 140262.88ms	

Pengujian 4 (Memiliki Solusi)

Input	Output
AAAA BBBB BCCC DDDC	
Banyak kasus yang ditinjau: 328	
Waktu pencarian: 1.54ms	

Pengujian 5 (Tidak Memiliki Solusi)

Input	Output

	
Banyak kasus yang ditinjau: 53130	
Waktu pencarian: 230.38ms	

Pranala Repository

Repository dapat diakses melalui pranala berikut: github.com/hsbu/Tucil1_18223014

Pernyataan Kejujuran

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

Lampiran

No	Poin	Ya	Tidak
1	Program berhasil di kompilasi tanpa kesalahan	√	
2	Program berhasil dijalankan	√	

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		√