IF2211 Strategi Algoritma

Tucil 1

Disusun oleh :

Muhammad Adam Mirza (18223015)

Dosen Pengampu:

Prof. Dr. Ir. Rinaldi, M.T.

PROGRAM STUDI SISTEM DAN TEKNOLOGI INFORMASI

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

JL. GANESHA 10, BANDUNG 40132

2026

DAFTAR ISI

1. Algoritma *Bruteforce*

```
def permutate(qpos, idx, cb):
    global iter, res
    if idx== len(qpos):
        iter+= 1
        n = len(cb)
        if iter% 1000 == 0:
            util.printboard(cb, qpos)
            print(f"Iterasi: {iter}")
        if valid(cb, qpos):
            res = qpos.copy()
            return True
        return False
    for i in range(idx, len(qpos)):
        qpos[idx],qpos[i] = qpos[i], qpos[idx]
        if permutate(qpos, idx+ 1, cb):
            return True
        qpos[idx],qpos[i] = qpos[i],qpos[idx]
    return False
```

1. Mulai dari array posisi ratu, misal [1,2,0,3]:
   a. Baris 0 → kolom 1
   b. Baris 1 → kolom 2
   c. Baris 2 → kolom 0
   d. Baris 3 → kolom 3
2. Ambil index pertama (idx = 0) sebagai posisi yang akan dicoba dulu.
3. Coba setiap kemungkinan dengan menukar ratu di posisi idx dengan setiap posisi setelahnya, sehingga semua kombinasi posisi untuk baris pertama akan dicoba.
4. Pindah ke baris berikutnya (idx + 1) dan ulangi proses swap untuk

mencoba semua kemungkinan di baris itu.

5. Lanjut rekursif sampai seluruh baris sudah memiliki posisi ratu, artinya satu permutasi lengkap terbentuk.
6. Cek apakah permutasi itu valid (tidak ada ratu yang saling menyerang).
7. Kalau valid, simpan permutasi sebagai solusi dan hentikan pencarian.
8. Kalau tidak valid, kembali ke langkah sebelumnya untuk mencoba kombinasi lain.
9. Terus ulangi sampai ditemukan solusi pertama yang valid atau semua permutasi habis dicoba.

2. Source Code

1) algo.py

```
#reference =
https://www.geeksforgeeks.org/dsa/print-all-possible-permu
tations-of-an-array-vector-without-duplicates-using-backtr
acking/
import util
iter = 0
res = None

def permutate(qpos, idx, cb):
    global iter, res
    if idx== len(qpos):
        iter+= 1
        n = len(cb)
        if iter% 1000 == 0:
            util.printboard(cb, qpos)
            print(f"Iterasi: {iter}")
        if valid(cb, qpos):
            res = qpos.copy()
            return True
        return False
    for i in range(idx, len(qpos)):
        qpos[idx],qpos[i] = qpos[i], qpos[idx]
        if permutate(qpos, idx+ 1, cb):
            return True
        qpos[idx],qpos[i] = qpos[i],qpos[idx]
    return False

def getsolution(cb, qpos):
```

```
    global iter, res
    iter,res =0,None
    permutate(qpos, 0, cb)
    return res, iter


def valid(cb, qpos):
    area = set()
    for i in range(len(qpos)):
        c = qpos[i]
        temp = cb[i][c]
        if temp in area:
            return False
        area.add(temp)
    for i in range(1,len(qpos)):
        c = qpos[i]
        prev = qpos[i- 1]
        if abs(c -prev)<= 1:
            return False
    return True
```

2) main.py

```
import time
import sys
import os
sys.path.append('src')
import algo
import util

def main():
    name = input("Nama file input (input1..input5): ")
    filepath = "input/" + name + ".txt"
    if not os.path.isfile(filepath):
        print("File tidak ditemukan.")
        return
```

```
    cb = []
    with open(filepath, 'r') as f:
        for line in f:
            clean_line = line.strip()
            if clean_line:
                cb.append(list(clean_line))
    n = len(cb)
    qpos = list(range(n))

    start = time.time()
    res, iter = algo.getsolution(cb, qpos)
    end = time.time()
    ms = (end- start) * 1000
    lines = None

    if res:
        lines = util.printboard(cb, res)
        print(f"{ms:.2f} ms")
        print(f"Jumlah iterasi: {iter}")
    else:
        lines = None
        print("Tidak ada solusi.")
        print(f"{ms:.2f} ms")
        print(f"Jumlah iterasi: {iter}")
    os.makedirs("test",exist_ok=True)
    solpath = os.path.join("test", name + "_sol.txt")
    with open(solpath, 'w') as f:
        if lines:
            for line in lines:
                f.write(line + "\n")
        else:
            f.write("Tidak ada solusi.\n")
        f.write(f"{ms:.2f} ms\n")
        f.write(f"Jumlah iterasi: {iter}\n")
    print(f"Solusi disimpan di {solpath}")

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

3) util

```
def printboard(cb, qpos):
    lines = []
    for i in range(len(cb)):
        line = ""
```

```
        for c in range(len(cb[i])):
            if qpos[i] == c:
                line+= "#"
            else:
                line+= cb[i][c]
        print(line)
        lines.append(line)
    return lines
```

3. Tangkapan Layar

1) input 1 and solution 1

input > input1.txt

```
AAABBCCCD
ABBBBCECD
ABBBDCECD
AAABDCCCD
BBBBDDDDD
FGGGDDHDD
FGIGDDHDD
FGIGDDHDD
FGGGDDHHH
```

```
AAABBCC#D
ABBB#CECD
ABBBDC#CD
A#ABDCCCD
BBBBD#DDD
FGG#DDHDD
#GIGDDHDD
FG#GDDHDD
FGGGDDHH#
431.53 ms
Jumlah iterasi: 300387
Solusi disimpan di test\input1_sol.txt
```

2) input 2 and solution 2

input > input2.txt

```
AAAABBBCCC
ADDEEBFGCC
HDDEEBFGII
HDJJJBFGII
HKKJJLLMMM
NKKOOPPMMM
NNQOOPPRRR
SSQQTUUVRR
SSQQTUUVWW
XXYYYZZZWW
```

```
#AAABBBCCC
AD#EEBFGCC
HDDE#BFGII
HDJJJB#GII
H#KJJLLMMM
NKKOO#PMMM
NNQOOPPR#R
SSQ#TUUVRR
SSQQTUU#WW
XXYYYZZZW#
93.88 ms
Jumlah iterasi: 52575
Solusi disimpan di test\input2_sol.txt
```

3) input 3 and solution 3

input > input3.txt

```
1  AAABBBCCC
2  AABBBCCCD
3  EEFFFDDDG
4  EEFHHDDGG
5  EIIIHHGGG
6  JJIIIKKLL
7  JJMMMKKLL
8  NNOOMMPPP
9  NNOOQQPPP
```

```
#AABBBCCC
AA#BBCCCD
EEFF#DDDG
E#FHHDDGG
EII#HHGGG
JJIIIK#LL
JJMMMKKL#
NNOOM#PPP
NNOOQQP#P
11.02 ms
Jumlah iterasi: 6612
Solusi disimpan di test\input3_sol.txt
```

4) input 4 and solution 4

input > input4.txt

```
AAABBBCC
ADDBBEEC
FDDBGEEE
FFDGGGHH
FFIGGGHH
FFIGGAAH
IIIJJAAH
IIIJJJAA
```

```
#AABBBCC
ADD#BEEC
FDDBGE#E
F#DGGGHH
FFIG#GHH
FF#GGAAH
IIIJJAA#
IIIJJ#AA
0.00 ms
Jumlah iterasi: 1924
Solusi disimpan di test\input4_sol.txt
```

5) input 5 and solution 5

input > input5.txt

```
AABBB
AACCC
BCCCC
BBBBD
DDDDE
```

```
Tidak ada solusi.
0.00 ms
Jumlah iterasi: 120
Solusi disimpan di test\input5_sol.txt
```

4. 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		✓

Github Repository: https://github.com/AdmMRZ/Tucil1_18223015

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

Muhammad Adam Mirza