

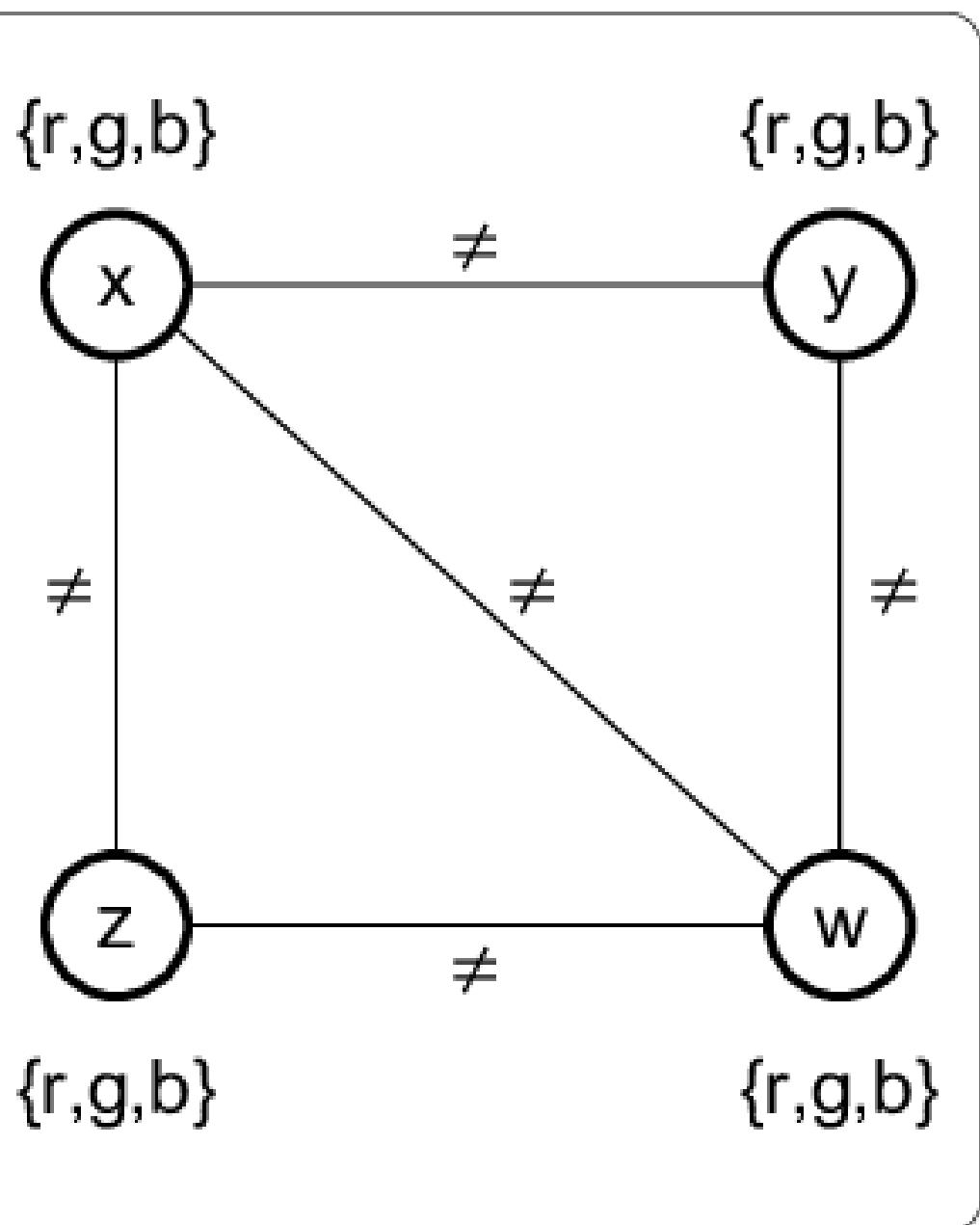
Tugas Individu 3

Menyelesaikan CSP untuk kasus Map Coloring

T I M O T H Y H O S I A B U D I A N T O
5 0 2 5 2 1 1 0 9 8



Map Coloring & CSP



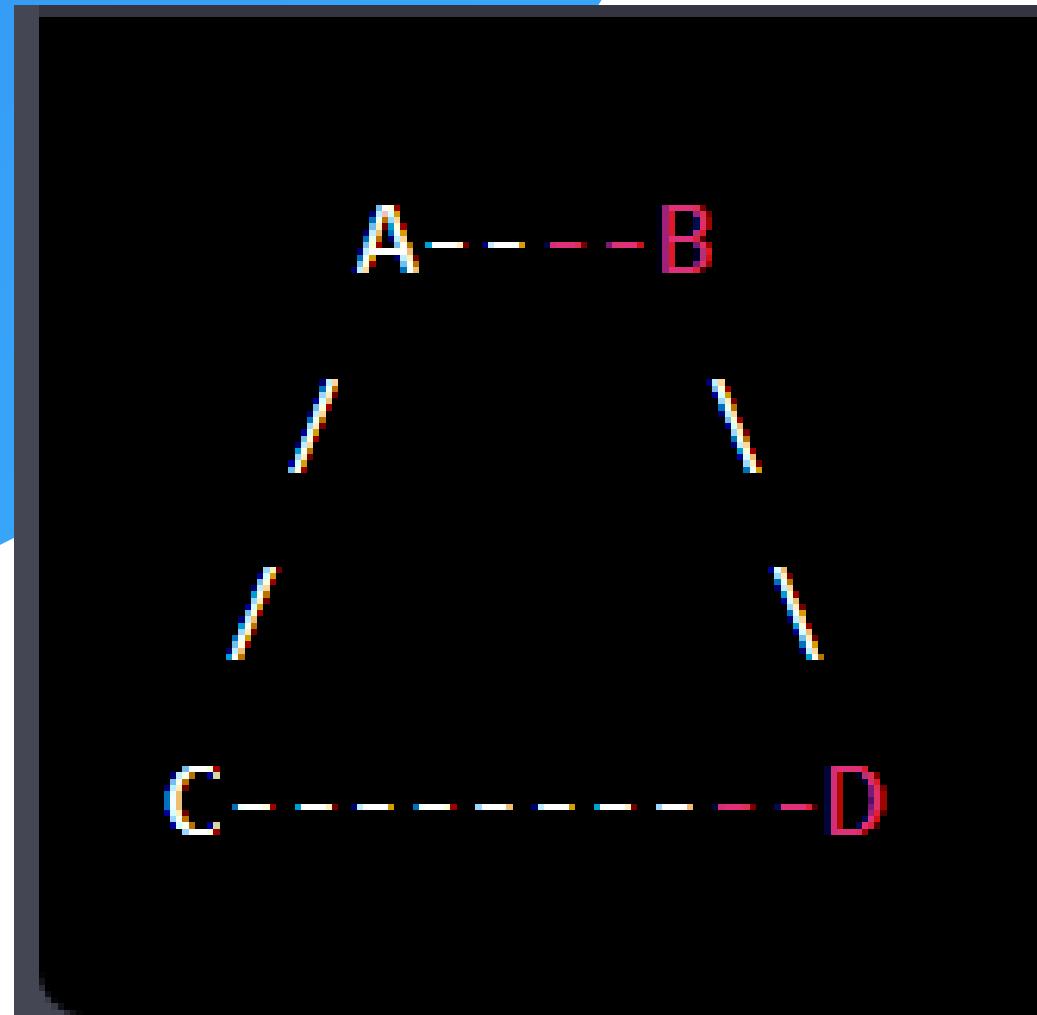
Map coloring adalah masalah matematika yang melibatkan pemberian warna pada daerah-daerah di dalam sebuah peta sehingga setiap daerah yang saling bersebelahan tidak memiliki warna yang sama. Tujuan dari map coloring adalah untuk menentukan jumlah warna minimum yang diperlukan untuk mewarnai semua daerah pada peta dengan syarat yang diberikan.

CSP (Constraint Satisfaction Problem) adalah model masalah di mana objek-objek harus ditempatkan atau diatur sedemikian rupa sehingga memenuhi sejumlah batasan atau keterikatan tertentu (constraint). Tujuan dari CSP adalah untuk menemukan solusi yang memenuhi semua keterikatan atau constraint tersebut. Map coloring adalah salah satu contoh dari CSP, di mana keterikatan antara daerah-di dalam peta adalah bahwa setiap daerah yang bersebelahan harus memiliki warna yang berbeda.

Soal

**Buatlah program untuk menyelesaikan
map coloring problem (csp)**

IMPLEMENTASI KODE



Program ini akan mencoba untuk mewarnai setiap daerah di peta dengan warna merah, hijau, dan biru, sehingga setiap daerah yang bersebelahan tidak memiliki warna yang sama.

IMPLEMENTASI KODE

```
1 from constraint import *
2
3 # Inisialisasi solver CSP
4 problem = Problem()
5
6 # Daftar daerah pada peta
7 areas = ['A', 'B', 'C', 'D']
8
9 # Setiap daerah harus diberi warna merah, hijau, atau biru
10 colors = ['red', 'green', 'blue']
11
12 # Tambahkan variabel-variabel ke solver CSP
13 for area in areas:
14     problem.addVariable(area, colors)
```

Program ini menggunakan library python-constraint yang menyediakan solvers untuk menyelesaikan masalah CSP. Pertama-tama, kita inisialisasi solver CSP menggunakan Problem(). Kemudian, kita menentukan daftar variabel-variabel pada masalah kita, yaitu daftar daerah pada peta, dan setiap variabel harus diberi nilai dari daftar warna yang tersedia.

```
16 # Setiap daerah yang bersebelahan tidak boleh memiliki warna yang sama
17 problem.addConstraint(lambda a, b: a != b, ('A', 'B'))
18 problem.addConstraint(lambda a, b: a != b, ('A', 'C'))
19 problem.addConstraint(lambda a, b: a != b, ('B', 'D'))
20 problem.addConstraint(lambda a, b: a != b, ('C', 'D'))
21
22 # Solusi dari CSP
23 solutions = problem.getSolutions()
24
25 # Cetak hasil solusi
26 for solution in solutions:
27     print(solution)
```

IMPLEMENTASI KODE

Selanjutnya, kita menambahkan keterikatan atau constraint antara variabel-variabel. Dalam kasus ini, setiap daerah yang bersebelahan tidak boleh memiliki warna yang sama. Kita menambahkan constraint ini ke solver CSP menggunakan `addConstraint()`.

Akhirnya, kita meminta solver untuk mencari solusi dari masalah CSP yang telah didefinisikan. Solusi-solusi yang ditemukan disimpan dalam variabel `solutions`. Kita dapat mencetak hasil solusi menggunakan loop `for`.

```
{'A': 'green', 'B': 'blue', 'C': 'blue', 'D': 'green'}  
{'A': 'green', 'B': 'blue', 'C': 'red', 'D': 'green'}  
{'A': 'green', 'B': 'red', 'C': 'blue', 'D': 'green'}  
{'A': 'green', 'B': 'red', 'C': 'red', 'D': 'blue'}  
{'A': 'green', 'B': 'red', 'C': 'red', 'D': 'green'}  
{'A': 'red', 'B': 'green', 'C': 'green', 'D': 'red'}  
{'A': 'red', 'B': 'green', 'C': 'green', 'D': 'blue'}  
{'A': 'red', 'B': 'green', 'C': 'blue', 'D': 'red'}  
{'A': 'red', 'B': 'blue', 'C': 'green', 'D': 'red'}  
{'A': 'red', 'B': 'blue', 'C': 'blue', 'D': 'green'}  
{'A': 'red', 'B': 'blue', 'C': 'blue', 'D': 'red'}
```

IMPLEMENTASI KODE

Program akan mengoutputkan kemungkinan kombinasi warna di tiap daerah



STUDIO SHODWE

Thank You

CUKUP SEKIAN