KECERDASAN BUATAN F

PENYELESAIAN CSP UNTUK KASUS MAP COLORING

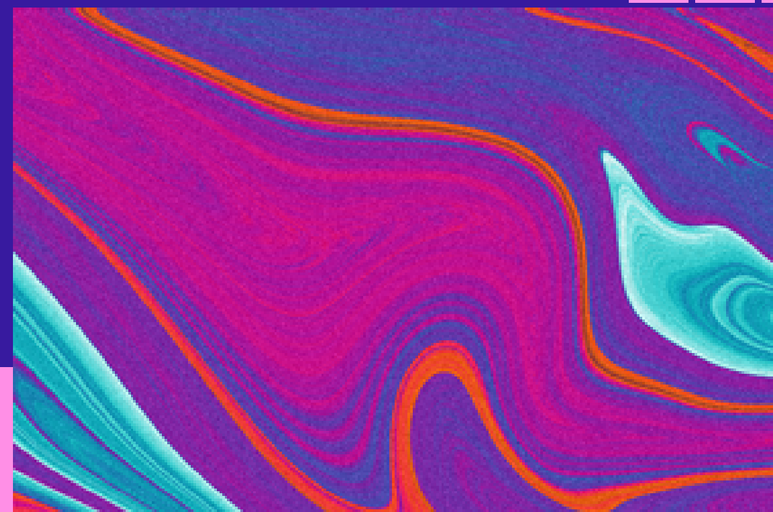
Najma UlyaAgustina-5025211239

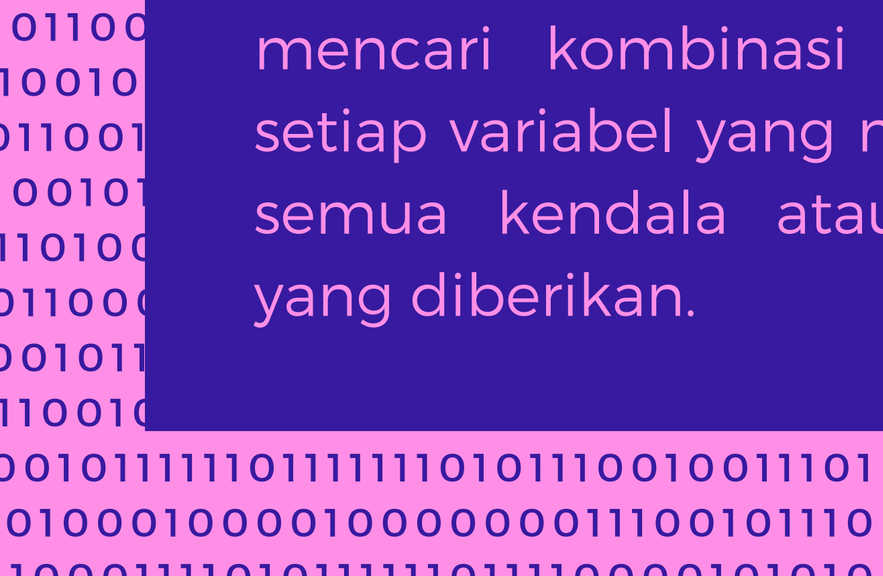
CSP

Constraint Satisfaction Problem

CSP adalah sebuah jenis masalah dalam bidang kecerdasan buatan (artificial intelligence) yang melibatkan pencarian solusi yang memenuhi sejumlah kendala atau batasan tertentu.

Tujuan dari CSP adalah mencari kombinasi nilai dari setiap variabel yang memenuhi semua kendala atau batasan yang diberikan.

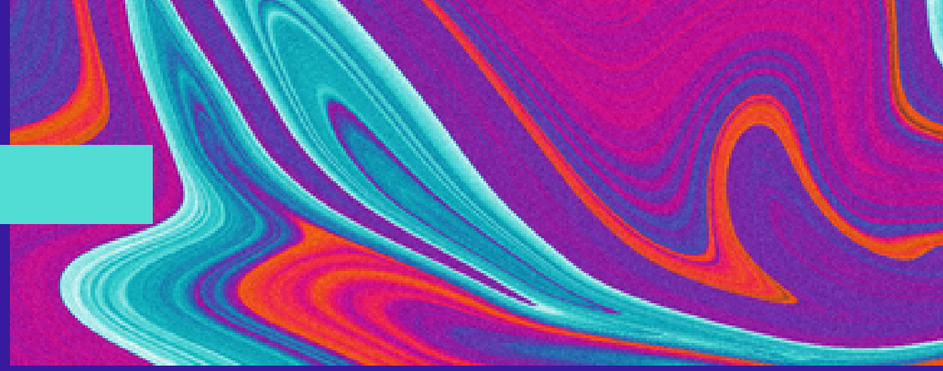


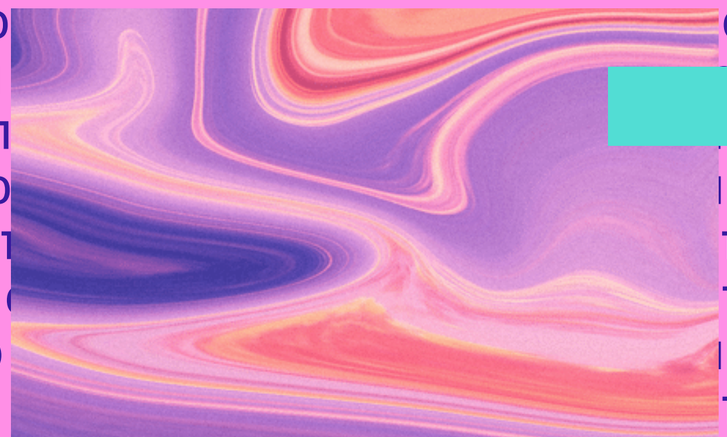


CSP pada Color Maping

Contoh kasus CSP adalah permasalahan pemberian warna pada peta. Dalam permasalahan ini, setiap daerah pada peta harus diberi warna yang berbeda dengan daerah sekitarnya.

Variabel dalam masalah ini adalah daerah-daerah pada peta yang harus diwarnai, sedangkan **kendala** adalah bahwa dua daerah yang berbatasan tidak boleh diberi warna yang sama.





Tujuan Masalah

Tujuan dari masalah ini adalah mencari kombinasi warna untuk setiap daerah pada peta yang memenuhi semua kendala atau batasan yang diberikan.

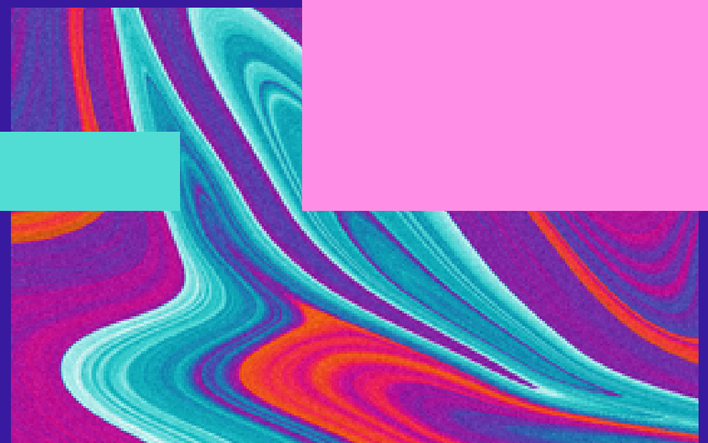
Implementasi

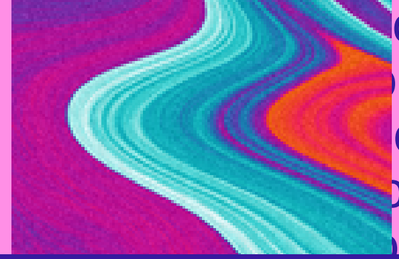
Kecerdasan-Buatan > mapcoloring.py > ...

```python
# Inisialisasi graf yang merepresentasikan peta
graph = {
    'WA': ['NT', 'SA'],
    'NT': ['WA', 'SA', 'Q'],
    'SA': ['WA', 'NT', 'Q', 'NSW', 'V'],
    'Q': ['NT', 'SA', 'NSW'],
    'NSW': ['Q', 'SA', 'V'],
    'V': ['SA', 'NSW']
}

# Inisialisasi variabel
variables = ['WA', 'NT', 'SA', 'Q', 'NSW', 'V']

# Inisialisasi domain nilai untuk setiap variabel
domain = {
    'WA': ['red', 'green', 'blue'],
    'NT': ['red', 'green', 'blue'],
    'SA': ['red', 'green', 'blue'],
    'Q': ['red', 'green', 'blue'],
    'NSW': ['red', 'green', 'blue'],
    'V': ['red', 'green', 'blue']
}
```





```
24  # Fungsi untuk mengecek apakah nilai yang diberikan pada variabel konflik dengan variabel tetangga
25  def is_consistent(var, value, assignment):
26      for neighbor in graph[var]:
27          if neighbor in assignment and assignment[neighbor] == value:
28              return False
29      return True
30
31  # Fungsi rekursif untuk mencari solusi
32  def backtrack(assignment):
33      # Jika assignment sudah memenuhi semua variabel, return assignment
34      if len(assignment) == len(variables):
35          return assignment
36
37      # Pilih variabel yang belum memiliki nilai
38      unassigned_vars = [var for var in variables if var not in assignment]
39      var = unassigned_vars[0]
```

```
    # Pilih variabel yang belum memiliki nilai
    unassigned_vars = [var for var in variables if var not in assignment]
    var = unassigned_vars[0]

    # Coba semua nilai yang mungkin untuk variabel tersebut
    for value in domain[var]:
        if is_consistent(var, value, assignment):
            # Jika nilai memenuhi kendala, tambahkan ke assignment dan coba variabel berikutnya
            assignment[var] = value
            result = backtrack(assignment)
            if result is not None:
                return result
            # Jika solusi tidak ditemukan, kembalikan variabel ke keadaan sebelumnya dan coba nilai beriku
            del assignment[var]

    # Jika tidak ada nilai yang memenuhi kendala, kembalikan None
    return None

# Cetak solusi
solution = backtrack({})
print(solution)
```

TERIMAKASIH!