```
from simpleai.search import CspProblem, backtrack


# Constraint function: two connected nodes must have different colors
def constraint_func(names, values):
return values[0] != values[1]


if __name__ == "__main__":

# Variables (nodes)
names = ('Ma', 'Ju', 'St', 'Am', 'Br',
'Jo', 'De', 'Al', 'Mi', 'Ke')

# Domain (colors)
colors = dict((name, ['red', 'green', 'blue', 'gray']) for name in names)

# Constraints (edges)
constraints = [

(('Ma', 'Ju'), constraint_func),
(('Ma', 'St'), constraint_func),

(('Ju', 'St'), constraint_func),
(('Ju', 'Am'), constraint_func),
(('Ju', 'De'), constraint_func),
(('Ju', 'Br'), constraint_func),

(('St', 'Am'), constraint_func),
(('St', 'Al'), constraint_func),
(('St', 'Mi'), constraint_func),

(('Am', 'Mi'), constraint_func),
(('Am', 'Jo'), constraint_func),
(('Am', 'De'), constraint_func),

(('Br', 'De'), constraint_func),
(('Br', 'Ke'), constraint_func),

(('Jo', 'Mi'), constraint_func),
(('Jo', 'Am'), constraint_func),
(('Jo', 'De'), constraint_func),
(('Jo', 'Ke'), constraint_func),

(('De', 'Ke'), constraint_func),
]

# Create CSP problem
problem = CspProblem(names, colors, constraints)

# Solve using backtracking
output = backtrack(problem)

# Print solution
print("\nColor mapping:\n")
for k, v in output.items():
```

```
print(k, "==>", v)
```

output:

```
Color mapping:

Ma ==> red
Ju ==> green
St ==> blue
Am ==> red
Br ==> red
Jo ==> green
De ==> blue
Al ==> red
Mi ==> gray
Ke ==> gray
```