Elective Course
Course Code: CS4103
Autumn 2025-26

Lecture #19

Artificial Intelligence for Data Science

Week-6: CONSTRAINT SATISFACTION PROBLEM (CSP) [Part-V]

Problem Solving and Exploring ***python-constraint***

Course Instructor:

Dr. Monidipa Das

Assistant Professor

Department of Computational and Data Sciences

Indian Institute of Science Education and Research Kolkata, India 741246

Street Puzzle

1	2	3	4	5

N_i = {English, Spaniard, Japanese, Italian, Norwegian}
C_i = {Red, Green, White, Yellow, Blue}
D_i = {Tea, Coffee, Milk, Fruit-juice, Water}
J_i = {Painter, Sculptor, Diplomat, Violinist, Doctor}
A_i = {Dog, Snails, Fox, Horse, Zebra}

The Englishman lives in the Red house
The Spaniard has a Dog
The Japanese is a Painter
The Italian drinks Tea
The Norwegian lives in the first house on the left
The owner of the Green house drinks Coffee
The Green house is on the right of the White house
The Sculptor breeds Snails
The Diplomat lives in the Yellow house
The owner of the middle house drinks Milk
The Norwegian lives next door to the Blue house
The Violinist drinks Fruit juice
The Fox is in the house next to the Doctor's
The Horse is next to the Diplomat's

Who owns the Zebra?
Who drinks Water?

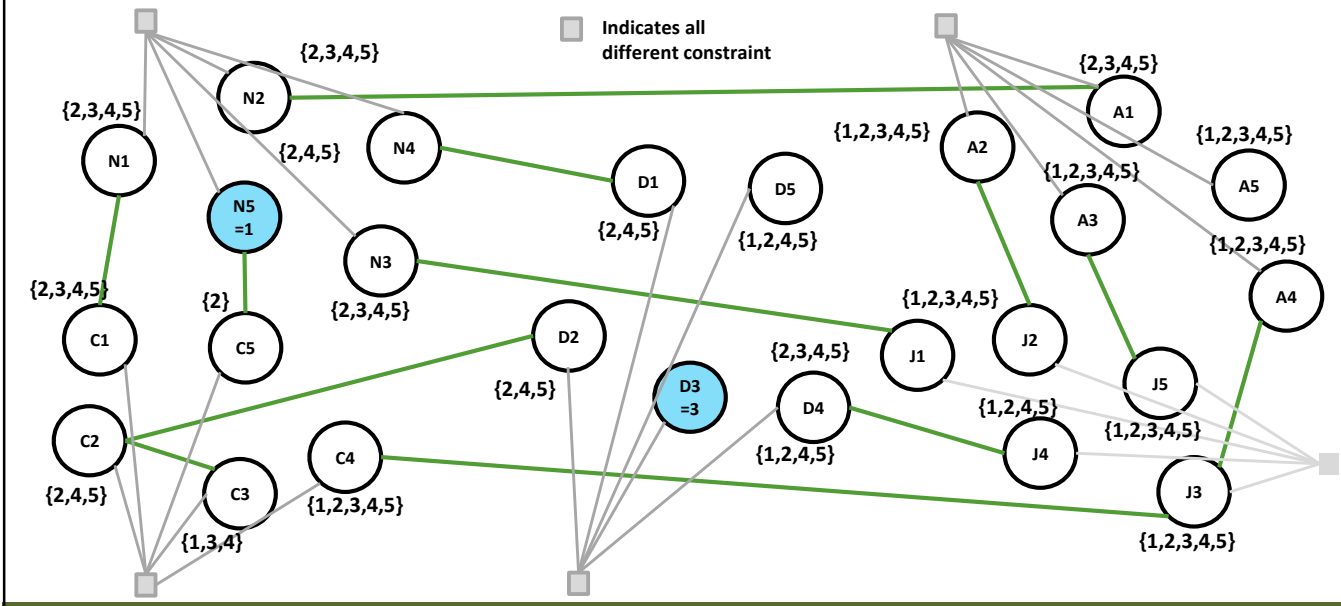
Street Puzzle (Solution)

1	2	3	4	5

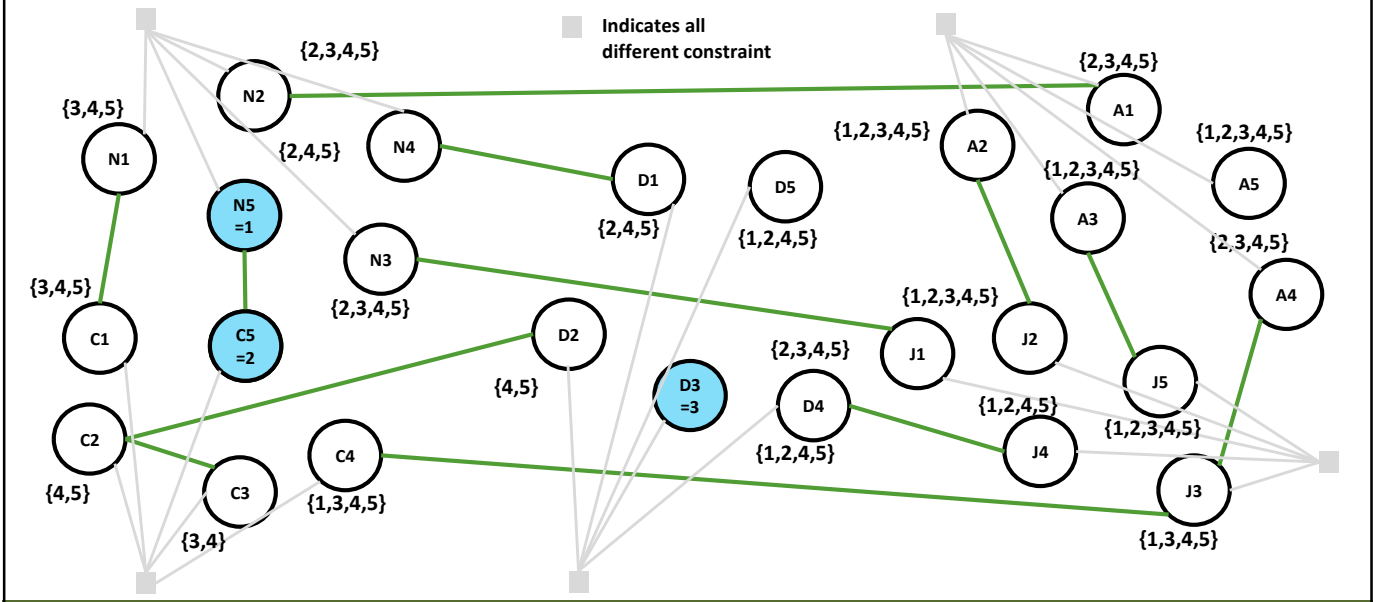
N_i = {English, Spaniard, Japanese, Italian, Norwegian}
C_i = {Red, Green, White, Yellow, Blue}
D_i = {Tea, Coffee, Milk, Fruit-juice, Water}
J_i = {Painter, Sculptor, Diplomat, Violinist, Doctor}
A_i = {Dog, Snails, Fox, Horse, Zebra}

1) The Englishman lives in the Red house → N1=C1
2) The Spaniard has a Dog → N2=A1
3) The Japanese is a Painter → N3=J1
4) The Italian drinks Tea → N4=D1
5) The Norwegian lives in the first house on the left → N5=H1
6) The owner of the Green house drinks Coffee → C2=D2
7) The Green house is on the right of the White house → C3=Hx ; C2=Hx+1
8) The Sculptor breeds Snails → J2=A2
9) The Diplomat lives in the Yellow house → J3=C4
10) The owner of the middle house drinks Milk → D3=H3
11) The Norwegian lives next door to the Blue house → C5=Hx ; N5=Hx±1
12) The Violinist drinks Fruit juice → J4=D4
13) The Fox is in the house next to the Doctor's → A3=Hx ; J5=Hx±1
14) The Horse is next to the Diplomat's → A4=Hx ; J3=Hx±1

Street Puzzle (Solution)



Street Puzzle (Solution)



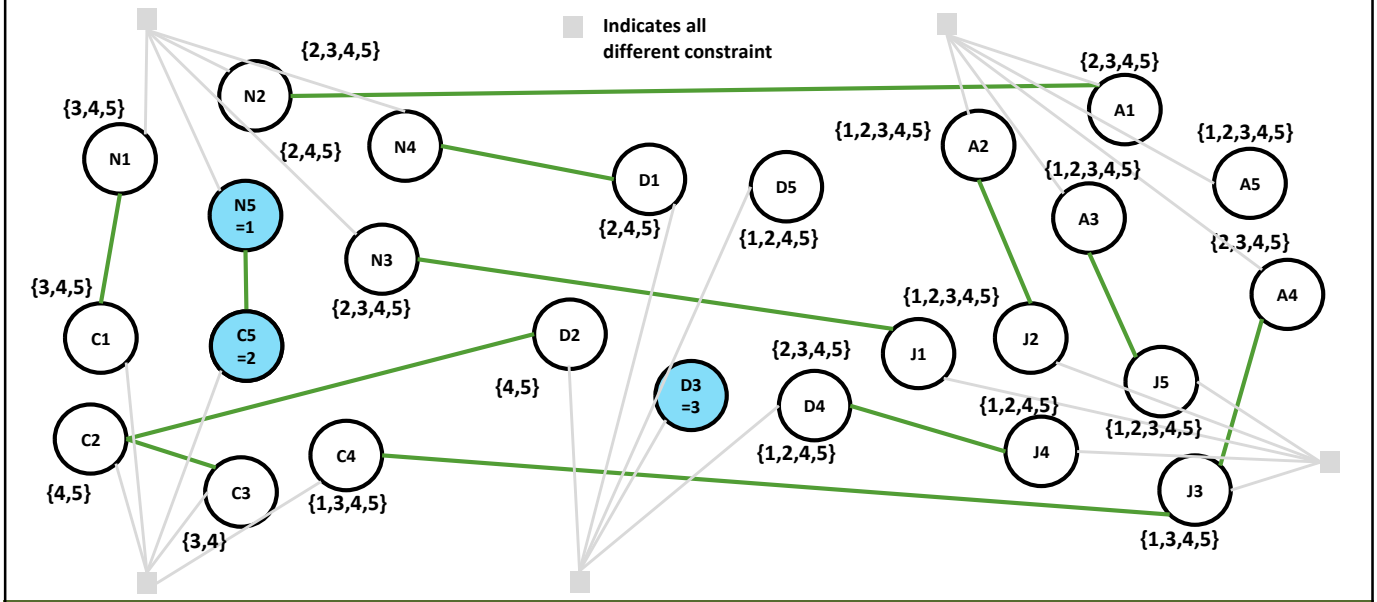
Street Puzzle (Solution)

1	2	3	4	5

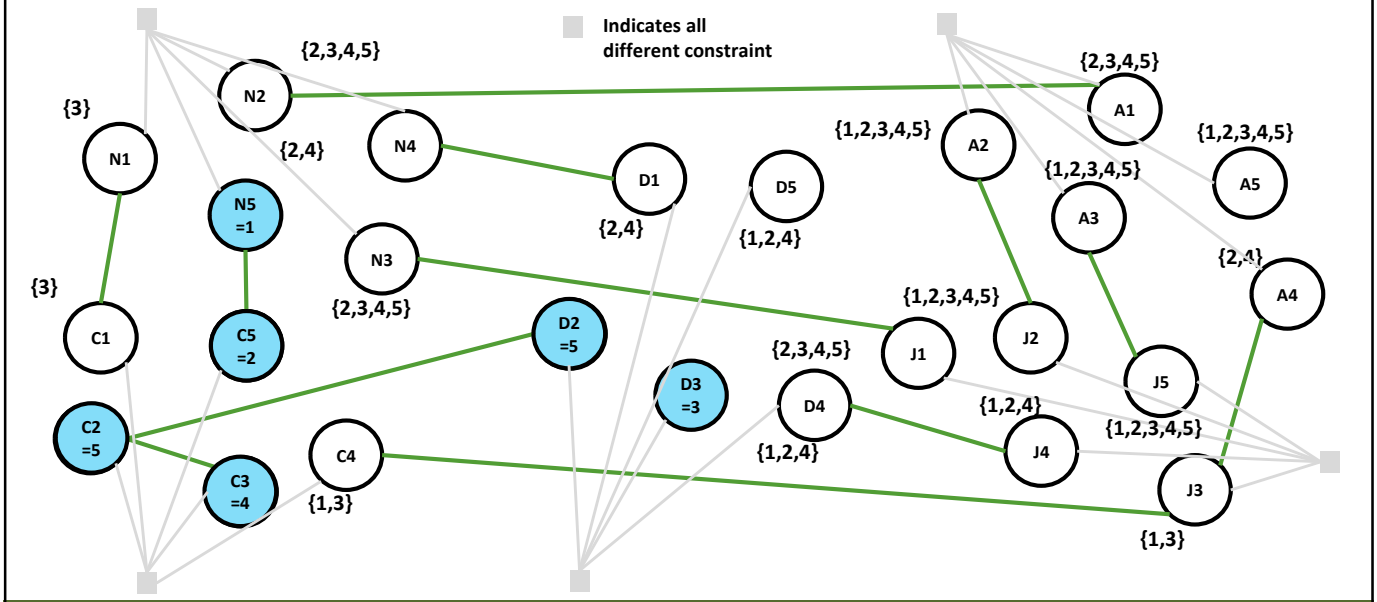
N_i = {English, Spaniard, Japanese, Italian, Norwegian}
C_i = {Red, Green, White, Yellow, Blue}
D_i = {Tea, Coffee, Milk, Fruit-juice, Water}
J_i = {Painter, Sculptor, Diplomat, Violinist, Doctor}
A_i = {Dog, Snails, Fox, Horse, Zebra}

1) The Englishman lives in the Red house → N1=C1
2) The Spaniard has a Dog → N2=A1
3) The Japanese is a Painter → N3=J1
4) The Italian drinks Tea → N4=D1
5) The Norwegian lives in the first house on the left → N5=H1
6) The owner of the Green house drinks Coffee → C2=D2
7) The Green house is on the right of the White house → C3=Hx ; C2=Hx+1
8) The Sculptor breeds Snails → J2=A2
9) The Diplomat lives in the Yellow house → J3=C4
10) The owner of the middle house drinks Milk → D3=H3
11) The Norwegian lives next door to the Blue house → C5=Hx ; N5=Hx±1
12) The Violinist drinks Fruit juice → J4=D4
13) The Fox is in the house next to the Doctor's → A3=Hx ; J5=Hx±1
14) The Horse is next to the Diplomat's → A4=Hx ; J3=Hx±1

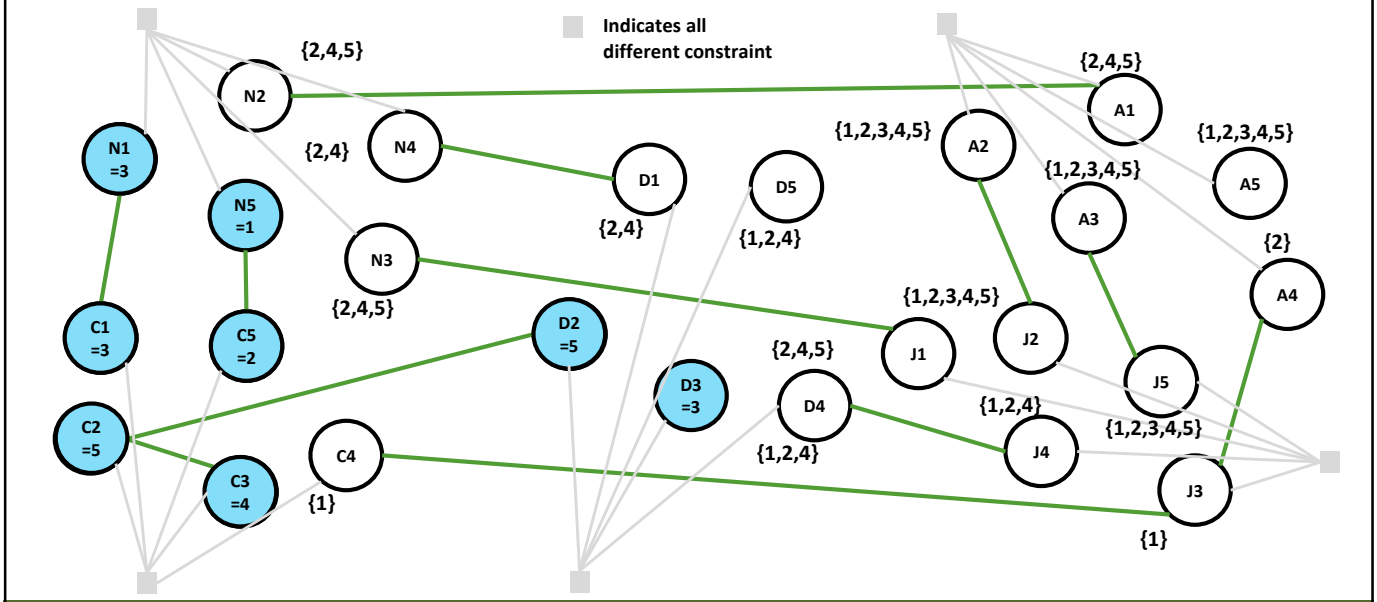
Street Puzzle (Solution)



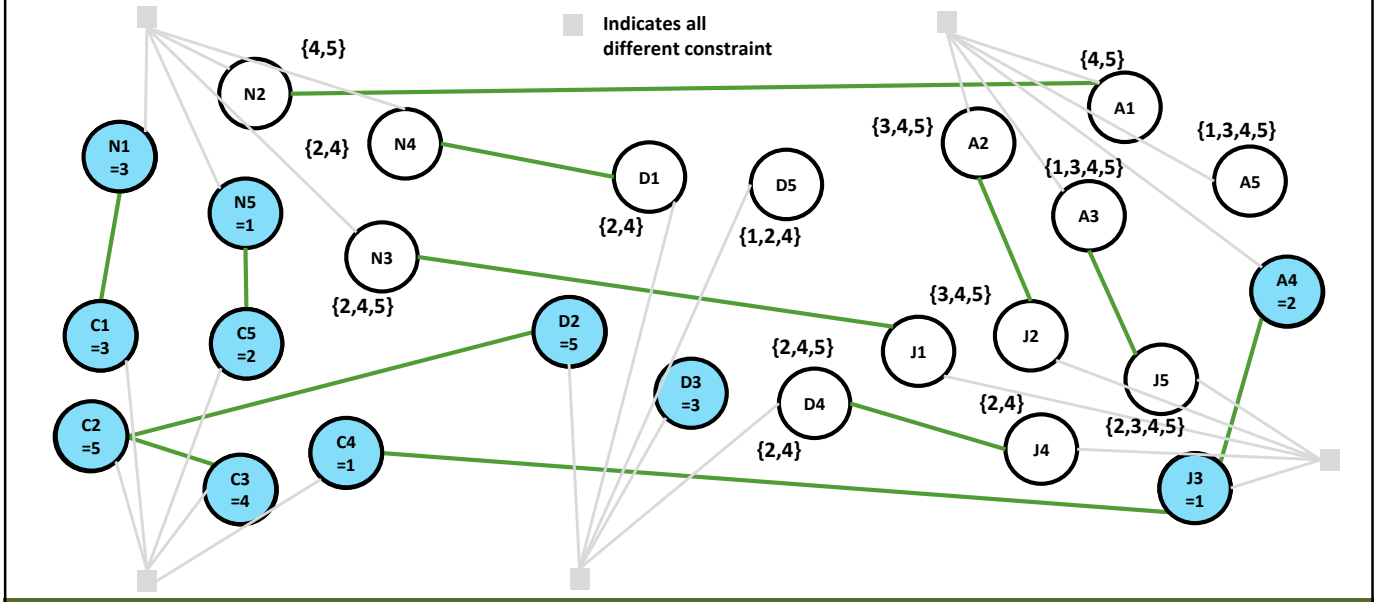
Street Puzzle (Solution)



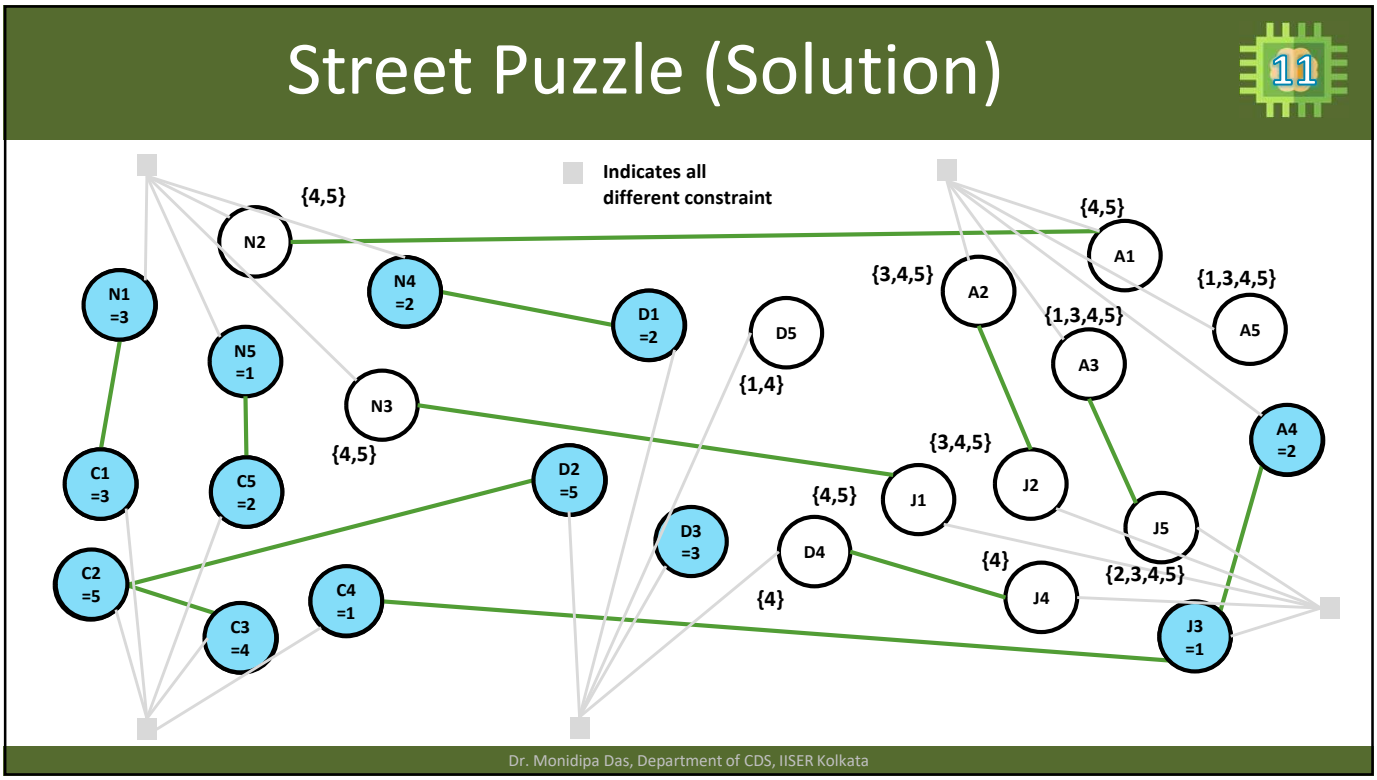
Street Puzzle (Solution)



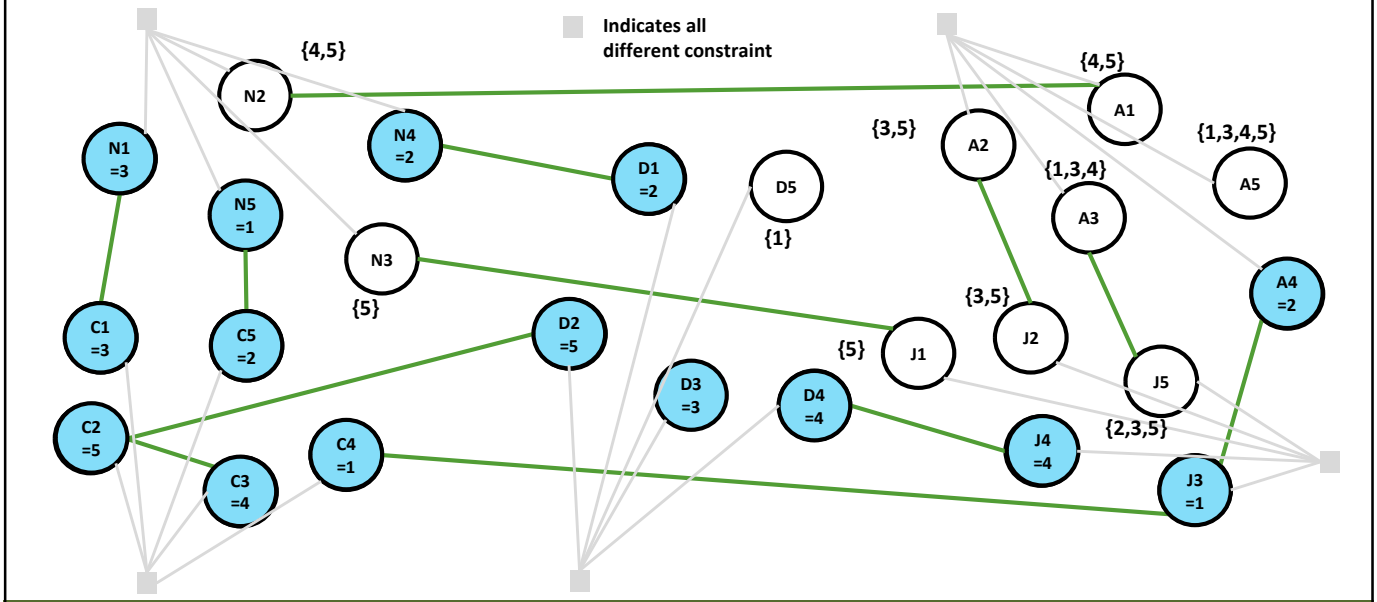
Street Puzzle (Solution)



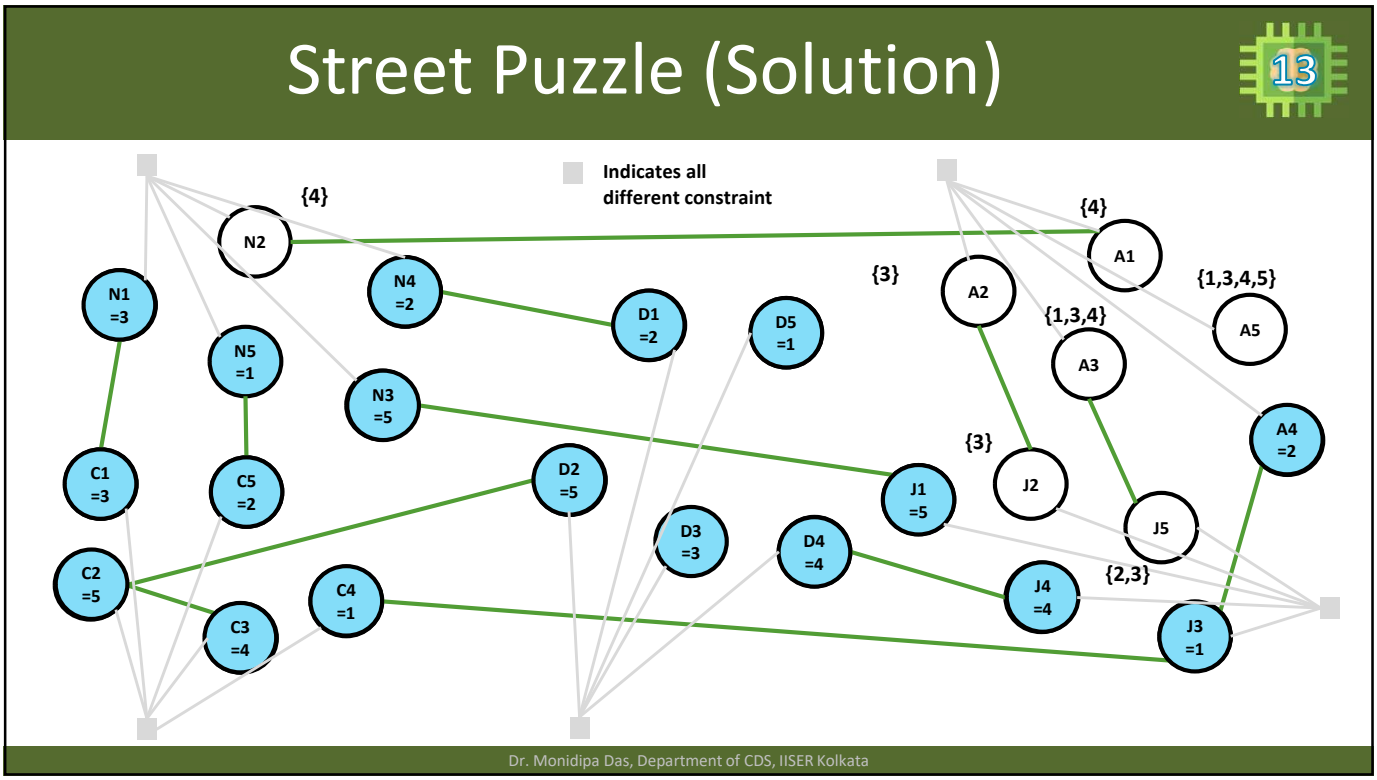
Street Puzzle (Solution)



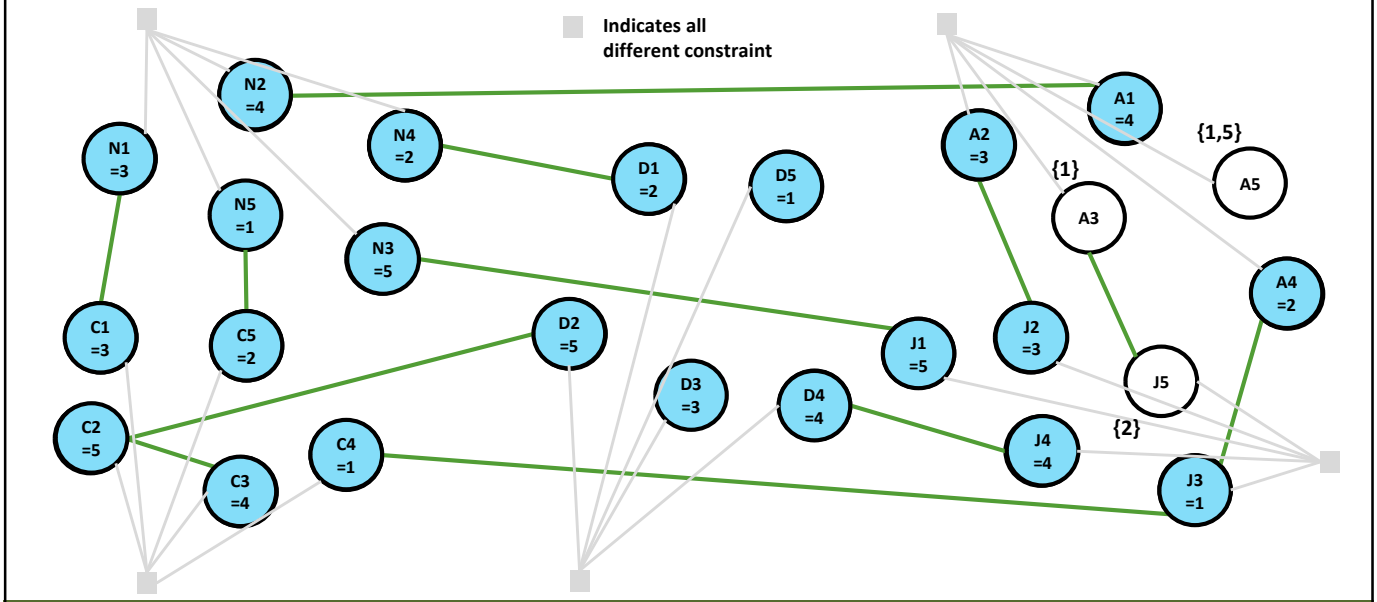
Street Puzzle (Solution)



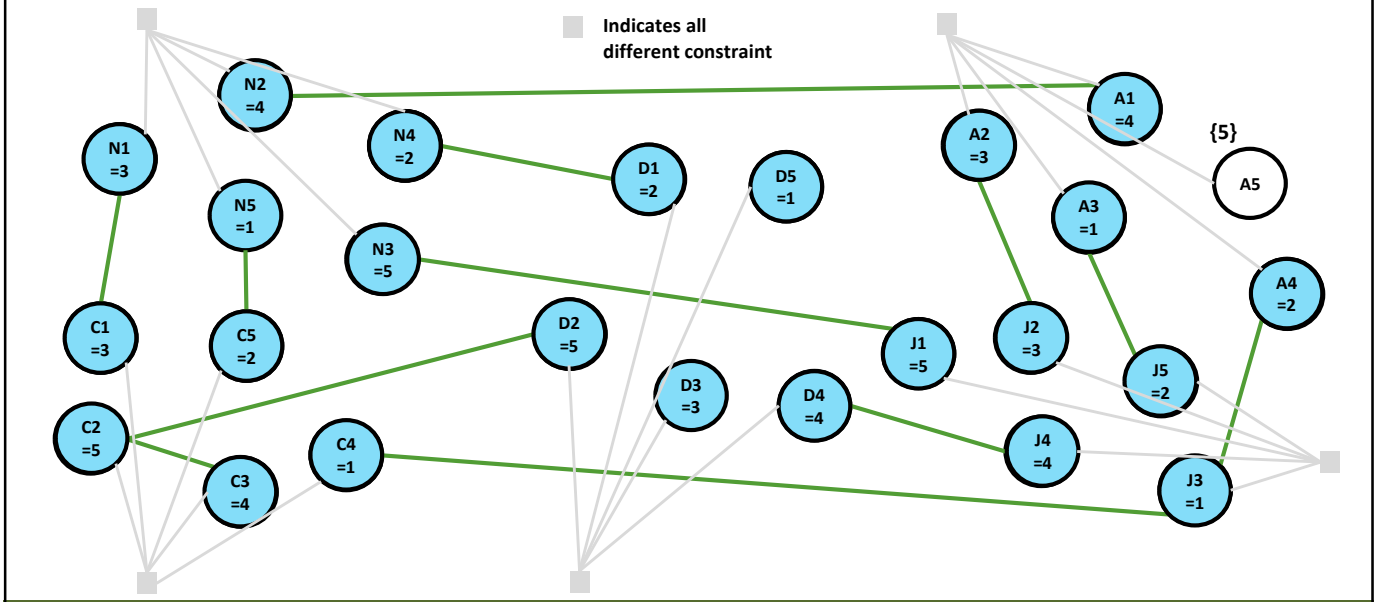
Street Puzzle (Solution)



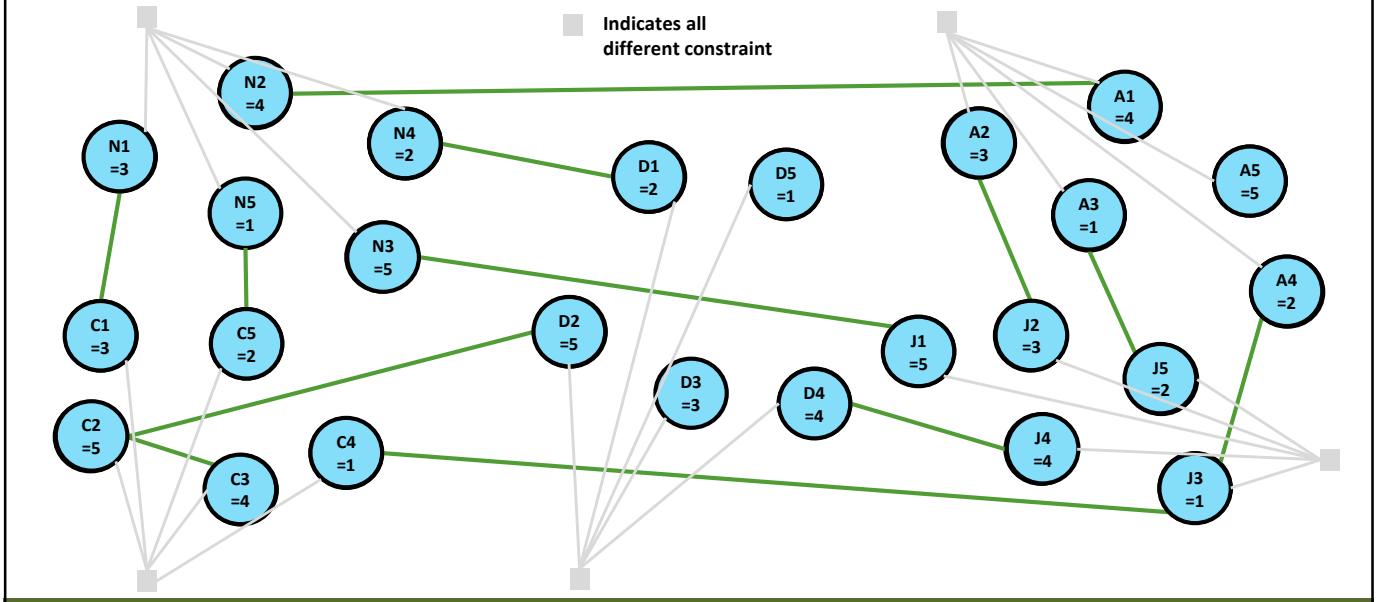
Street Puzzle (Solution)



Street Puzzle (Solution)

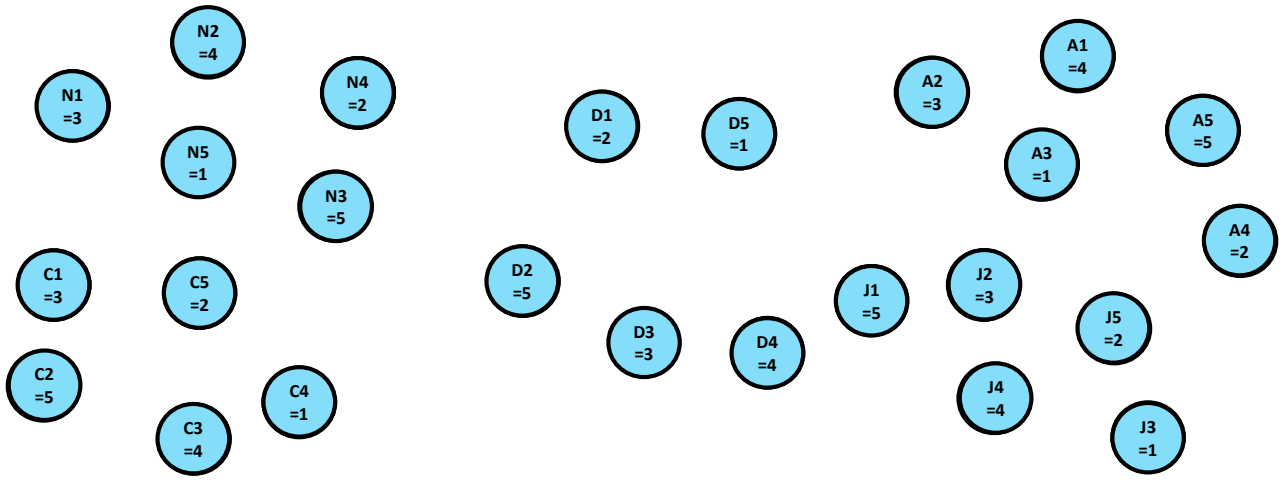


Street Puzzle (Solution)



Street Puzzle (Solution)

17



Street Puzzle (Solution)

18

Who owns the Zebra?
Who drinks Water?

N5 =1	N4 =2	N1 =3	N2 =4	N3 =5
C4 =1	C5 =2	C1 =3	C3 =4	C2 =5
D5 =1	D1 =2	D3 =3	D4 =4	D2 =5
J3 =1	J5 =2	J2 =3	J4 =4	J1 =5
A3 =1	A4 =2	A2 =3	A1 =4	A5 =5

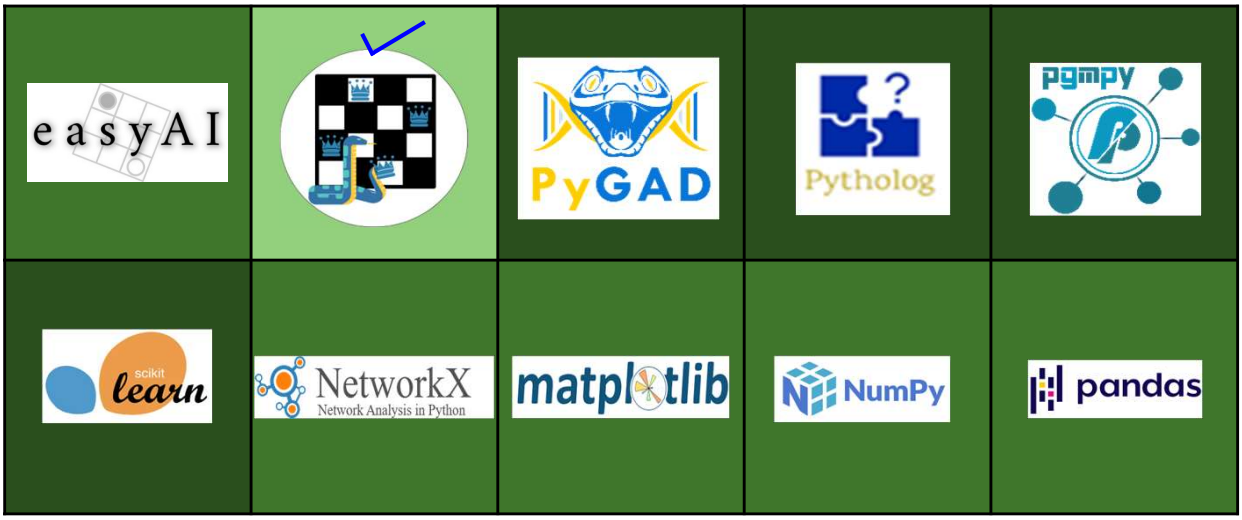
The **Norwegian** person stays in the 1st (left most) house which is of color **yellow**. He is a **diplomat** and prefers **Water**. He owns the animal **Fox.**

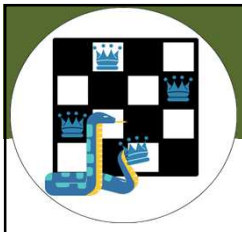
The **Japanese** person stays in the 5th (right most) house which is of color **green**. He is a **painter** and prefers **coffee**. He owns the animal **Zebra.**

N_i = {English, Spaniard, Japanese, Italian, Norwegian}
C_i = {Red, Green, White, Yellow, Blue}
D_i = {Tea, Coffee, Milk, Fruit-juice, Water}
J_i = {Painter, Sculptor, Diplomat, Violinist, Doctor}
A_i = {Dog, Snails, Fox, Horse, Zebra}

A Few Python Libraries for AI

19





python-constraint

20

pip install python-constraint

Example: Solving Street Puzzle

```
from constraint import Problem, AllDifferentConstraint

def solve():
   problem = Problem()
   for i in range(1, 6):
      problem.addVariable("N%d" % i,["English", "Spaniard", "Japanese", "Italian", "Norwegian"])
      problem.addVariable("C%d" % i,["Red", "Green", "White", "Yellow", "Blue"])
      problem.addVariable("D%d" % i,["Tea", "Coffee", "Milk", "Fruit-juice", "Water"])
      problem.addVariable("J%d" % i,["Painter", "Sculptor", "Diplomat", "Violinist", "Doctor"])
      problem.addVariable("A%d" % i,["Dog", "Snail", "Fox", "Horse", "Zebra"])

   problem.addConstraint(AllDifferentConstraint(), ["N%d" % i for i in range(1, 6)])
   problem.addConstraint(AllDifferentConstraint(), ["C%d" % i for i in range(1, 6)])
   problem.addConstraint(AllDifferentConstraint(), ["D%d" % i for i in range(1, 6)])
   problem.addConstraint(AllDifferentConstraint(), ["J%d" % i for i in range(1, 6)])
   problem.addConstraint(AllDifferentConstraint(), ["A%d" % i for i in range(1, 6)])
```

Solving Street Puzzle using python-constraint


```
for i in range(1, 6):
        # Constraint 1
        problem.addConstraint(
            lambda N, C: N!="English" or C=="Red",("N%d" % i, "C%d" % i))

        # Constraint 2
        problem.addConstraint(
            lambda N, A: N!="Spaniard" or A=="Dog",("N%d" % i, "A%d" % i))

        # Constraint 3
        problem.addConstraint(
            lambda N, J: N!="Japanese" or J=="Painter",("N%d" % i, "J%d" % i))

        # Constraint 4
        problem.addConstraint(
            lambda N, D: N!="Italian" or D=="Tea",("N%d" % i, "D%d" % i))

        # Constraint 5
        if i==1:
            problem.addConstraint(lambda N: N =="Norwegian",("N%d" % i,))
```


Solving Street Puzzle using python-constraint


```
        # Constraint 6
        problem.addConstraint(
            lambda D, C: D!="Coffee" or C=="Green",("D%d" % i, "C%d" % i))

        # Constraint 7
        if i < 5:
            problem.addConstraint(
                lambda CA, CB: CA!="green" or CB=="white",("C%d" % i, "C%d" % (i + 1)))
        else:
            problem.addConstraint(lambda C: C != "green", ("C%d" % i,))

        # Constraint 8
        problem.addConstraint(
            lambda J, A: J != "Sculptor" or A == "Snail",("J%d" % i, "A%d" % i))

        # Constraint 9
        problem.addConstraint(
            lambda J, C: J != "Diplomat" or C == "Yellow",("J%d" % i, "C%d" % i))

        # Constraint 10
        if i==3:
            problem.addConstraint(lambda D: D=="Milk",("D%d" % i,))
```

Solving Street Puzzle using python-constraint


```
# Constraint 11
if 1<i<5:
    problem.addConstraint(
        lambda N, CA, CB: N!="Norwegian" or CA=="Blue" or CB=="Blue",
        ("N%d" % i, "C%d" % (i+1),"C%d" % (i-1))
    )
elif i==1:
    problem.addConstraint(
        lambda N, C: N!="Norwegian" or C=="Blue",
        ("N%d" % i, "C%d" % (i+1))
    )
elif i==5:
    problem.addConstraint(
        lambda N, C: N!="Norwegian" or C=="Blue",
        ("N%d" % i, "C%d" % (i-1))
    )
# Constraint 12
problem.addConstraint(
    lambda J, D: J != "Violinist" or D == "Fruit-juice",
    ("J%d" % i, "D%d" % i)
)
```


Solving Street Puzzle using python-constraint

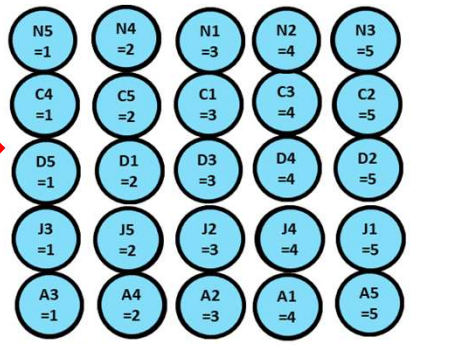

```
# Constraint 13
if 1 < i < 5:
    problem.addConstraint(
        lambda A, JA, JB: A!="Fox" or JA=="Doctor" or JB=="Doctor",
        ("A%d" % i, "J%d" % (i+1), "J%d" % (i-1)))
elif i==1:
    problem.addConstraint(lambda A, J: A!="Fox" or J=="Doctor",("A%d" % i, "J%d" % (i+1)))
elif i==5:
    problem.addConstraint(lambda A, J: A!="Fox" or J=="Doctor",("A%d" % i, "J%d" % (i-1)))

# Constraint 14
if 1 < i < 5:
    problem.addConstraint(
        lambda A, JA, JB: A!="Horse" or JA=="Diplomat" or JB=="Diplomat" ,
        ("A%d" % i, "J%d" % (i+1), "J%d" % (i-1)))
elif i==1:
    problem.addConstraint(
        lambda A, J: A!="Horse" or J=="Diplomat",("A%d" % i, "J%d" % (i+1)))
elif i==5:
    problem.addConstraint(
        lambda A, J: A!="Horse" or J=="Diplomat",("A%d" % i, "J%d" % (i-1)))
```

Solving Street Puzzle using python-constraint


```
    solutions = problem.getSolutions()
    return solutions

def showSolution(solution):
    for i in range(1, 6):
        print("House %d" % i)
        print("--------")
        print("Nationality: %s" % solution["N%d" % i])
        print("Color: %s" % solution["C%d" % i])
        print("Drink: %s" % solution["D%d" % i])
        print("Job: %s" % solution["J%d" % i])
        print("Animal: %s" % solution["A%d" % i])
        print("")

def main():
    solutions = solve()
    print("Found %d solution(s)!" % len(solutions))
    print("")
    # for solution in solutions:
    #     showSolution(solution)
    #     input()
    showSolution(solutions[5])
```


Solving Street Puzzle using python-constraint


```
In [29]: runfile('F:/CS4103/code/street.py', wdir='F:/CS4103/code')
Found 31 solution(s)!

House 1
--------
Nationality: Norwegian
Color: Yellow
Drink: Water
Job: Diplomat
Animal: Fox

House 2
--------
Nationality: Italian
Color: Blue
Drink: Tea
Job: Doctor
Animal: Horse

House 3
--------
Nationality: English
Color: Red
Drink: Milk
Job: Sculptor
Animal: Snail

House 4
--------
Nationality: Spaniard
Color: White
Drink: Fruit-juice
Job: Violinist
Animal: Dog

House 5
--------
Nationality: Japanese
Color: Green
Drink: Coffee
Job: Painter
Animal: Zebra
```



N_i = {English, Spaniard, Japanese, Italian, Norwegian}
C_i = {Red, Green, White, Yellow, Blue}
D_i = {Tea, Coffee, Milk, Fruit-juice, Water}
J_i = {Painter, Sculptor, Diplomat, Violinist, Doctor}
A_i = {Dog, Snails, Fox, Horse, Zebra}

Solving BASE + BALL = GAMES using python-constraint


```
from constraint import Problem, AllDifferentConstraint, NotInSetConstraint


def solve():
    problem = Problem()
    problem.addVariables("baselgm", range(10))
    problem.addVariables("wxyz", range(2))
    problem.addConstraint(lambda e, l, s, w : e + l == s + 10*w, "elsw")
    problem.addConstraint(lambda w, x, l, s, e : s + l + w == e + 10*x, "wxlse")
    problem.addConstraint(lambda y, a, m, x: 2*a + x == m + 10*y, "yamx")
    problem.addConstraint(lambda z,y,b,a: 2*b + y == a + 10*z, "zyba")
    problem.addConstraint(lambda g,z: g == z, "gz")
    problem.addConstraint(NotInSetConstraint([0]), "gb")
    problem.addConstraint(AllDifferentConstraint(),"baselgm")
    solutions = problem.getSolutions()
    return solutions
```


Solving BASE + BALL = GAMES using python-constraint

OR

```
def solve():
    problem = Problem()

    problem.addVariables("baselgm", range(10))

    problem.addConstraint(lambda b,a,s,e,l,g,m : 2*1000*b + 2*100*a + 10*s +
                e + 10* l + l== 10000*g + 1000*a + 100*m + 10*e + s,"baselgm")
    problem.addConstraint(NotInSetConstraint([0]), "gb")
    problem.addConstraint(AllDifferentConstraint(),"baselgm")

    solutions = problem.getSolutions()
    return solutions
```

Solving BASE + BALL = GAMES using python-constraint


```
def main():
    solutions = solve()
    print("BASE + BALL = GAMES")
    for s in solutions:
        #print(s)
        print("%(b)d%(a)d%(s)d%(e)d + %(b)d%(a)d%(l)d%(l)d =
                                        %(g)d%(a)d%(m)d%(e)d%(s)d" % s)


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

```
In [31]: runfile('F:/CS4103/code/baseball_m.py', wdir='F:/CS4103/code')
BASE + BALL = GAMES
7483 + 7455 = 14938
```


Questions?