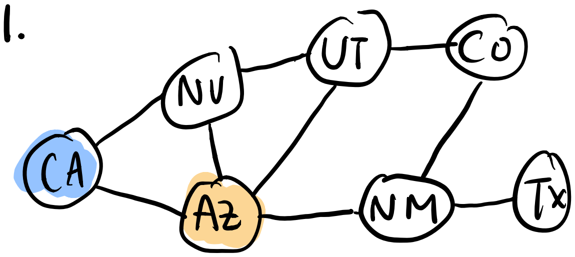
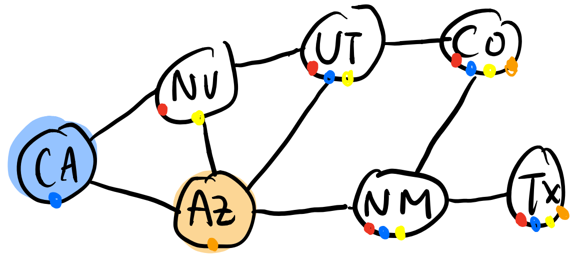
Jinqi Cheng Assignment5 PartI

1.



	CA	NV	Az	UT	NM	Co	TX
initial domains	RBYO	RBYO	RBYO	RBYO	RBYO	RBYO	RBYO
After CA = B	B	RYO	RYO	RBYO	RBYO	RBYO	RBYO
After AZ = O	B	RY	O	RBY	RBY	RBYO	RBYO

2.



	CA	NV	Az	UT	NM	Co	TX
initial domains	RBYO	RBYO	RBYO	RBYO	RBYO	RBYO	RBYO
After CA = B	B	RYO	RYO	RBYO	RBYO	RBYO	RBYO
After AZ = O	B	RY	O	RBY	RBY	RBYO	RBYO

After CA = B

Quene = (CA, NV) (CA, AZ),

NV = RYO, AZ = RYO

After Az=O

Queue = (Az, NV), (Az, UT), (Az, NM)

NV=RY, UT=RBY, NM=RBY

3.

	CA	NV	AZ	UT	NM	CO	TX
After CA=B	B	{CA=B}	{CA=B}	∅	∅	∅	∅
After Az=O	B	{CA=B, Az=O}	O	{Az=O}	{AZ=O}	∅	∅

4.

a. Assume we have 2 classes: Student and Course

```
class Student {
    self.name=" "        # Student's name
    self.courses=[ ]     # Student's course list
    self.workday=[ ]     # Days this student work

class Course {
    self.name=" "        # Course name
    self.time=" "        # Start & end time of the course
    self.day=" "         # day of the course
    self.capacity=30     # Capacity of the course
    self.size=0          # Current size of the course
```

CSP Specifications:

X = {Student1, Student2, ..., Student n} # instances of Student

D_x = {All student names, course lists, workdays} # instance variables of student

Y = {Course 1, Course2, Course3, Course 4} # instances of Course, total 4.

D_y = {courses name, time, day, capacity, size} # instance variables of student

```
C = {
  for ∃ student in X:        # the only overlapping courses are 1 and 3
     if 1 in student.courses and 3 in student.courses:
          return False
  for ∃ course in Y:         # in case the class is full
     if course.size > course.capacity:
          return False

  for ∃ student in X:
       for ∃ course in Y:    # in case students need to work
           if student.workday overlaps course.day:
                return False
```

b. Create an encapsulated variable U

Cartisan product U = X × Y

We can binarize the constraint by multiplying X and Y.