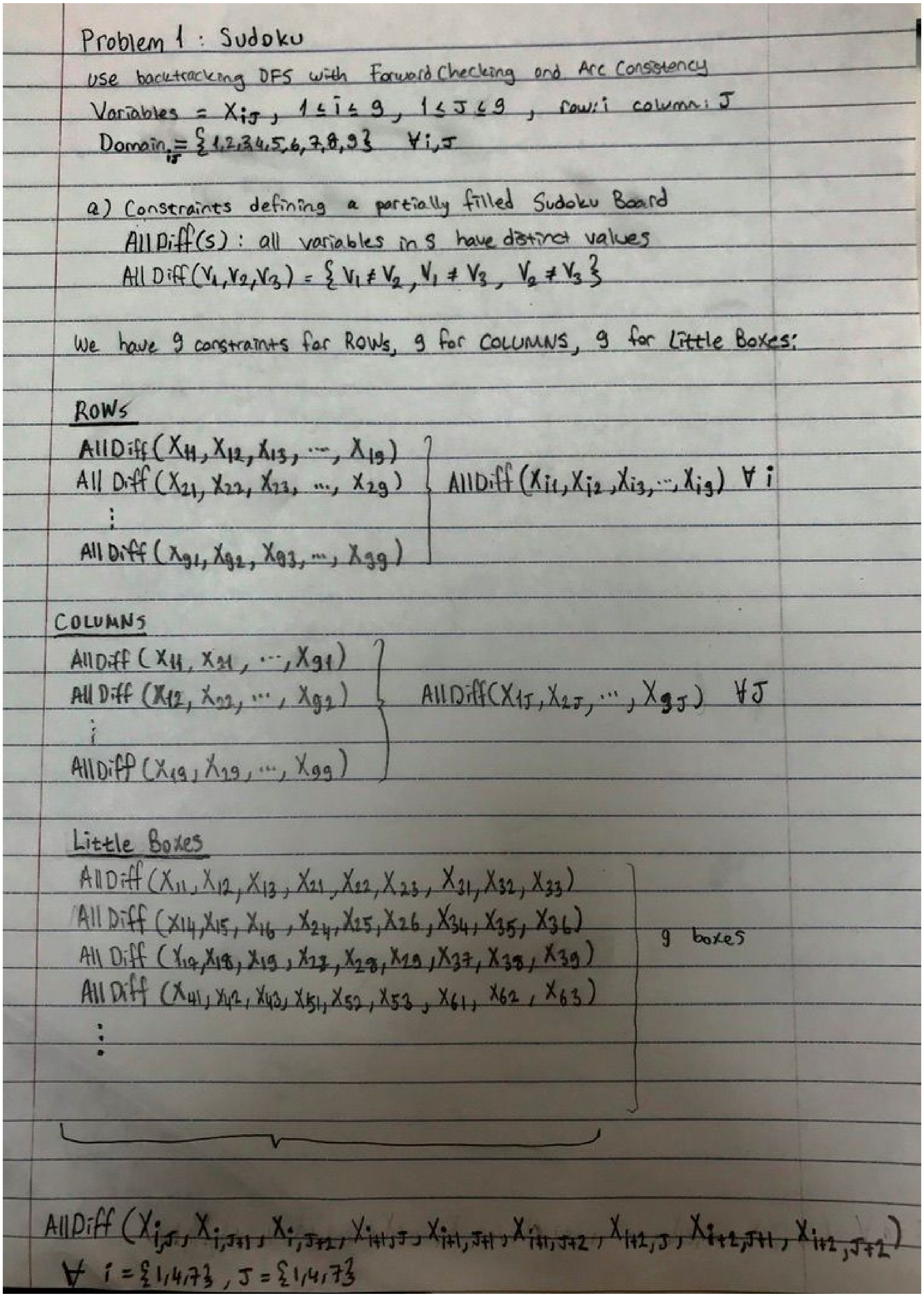
Ege Ersü
Linus Shih
Assignment 4



Problem 1 : Sudoku

use backtracking DFS with Forward Checking and Arc Consistency

Variables = X_{iJ}, $1 \le i \le 9$, $1 \le J \le 9$, row: i column: J

$\text{Domain}_{iJ} = \{1,2,3,4,5,6,7,8,9\} \quad \forall i, J$

a) Constraints defining a partially filled Sudoku Board

AllDiff(S) : all variables in S have distinct values

$\text{AllDiff}(V_1, V_2, V_3) = \{V_1 \ne V_2, V_1 \ne V_3, V_2 \ne V_3\}$

We have 9 constraints for ROWS, 9 for COLUMNS, 9 for Little Boxes:

ROWS

$$\left.\begin{array}{l} \text{AllDiff}(X_{11}, X_{12}, X_{13}, \cdots, X_{19}) \\ \text{AllDiff}(X_{21}, X_{22}, X_{23}, \cdots, X_{29}) \\ \vdots \\ \text{AllDiff}(X_{91}, X_{92}, X_{93}, \cdots, X_{99}) \end{array}\right\} \text{AllDiff}(X_{i1}, X_{i2}, X_{i3}, \cdots, X_{i9}) \ \forall i$$

COLUMNS

$$\left.\begin{array}{l} \text{AllDiff}(X_{11}, X_{21}, \cdots, X_{91}) \\ \text{AllDiff}(X_{12}, X_{22}, \cdots, X_{92}) \\ \vdots \\ \text{AllDiff}(X_{19}, X_{29}, \cdots, X_{99}) \end{array}\right\} \text{AllDiff}(X_{1J}, X_{2J}, \cdots, X_{9J}) \ \forall J$$

Little Boxes

$$\underbrace{\left.\begin{array}{l} \text{AllDiff}(X_{11}, X_{12}, X_{13}, X_{21}, X_{22}, X_{23}, X_{31}, X_{32}, X_{33}) \\ \text{AllDiff}(X_{14}, X_{15}, X_{16}, X_{24}, X_{25}, X_{26}, X_{34}, X_{35}, X_{36}) \\ \text{AllDiff}(X_{17}, X_{18}, X_{19}, X_{27}, X_{28}, X_{29}, X_{37}, X_{38}, X_{39}) \\ \text{AllDiff}(X_{41}, X_{42}, X_{43}, X_{51}, X_{52}, X_{53}, X_{61}, X_{62}, X_{63}) \\ \vdots \end{array}\right\} \text{9 boxes}}$$

$\text{AllDiff}(X_{i,J}, X_{i,J+1}, X_{i,J+2}, X_{i+1,J}, X_{i+1,J+1}, X_{i+1,J+2}, X_{i+2,J}, X_{i+2,J+1}, X_{i+2,J+2})$

$\forall \ i = \{1,4,7\}, J = \{1,4,7\}$

b) Forward Checking occurs after a variable is assigned a value. For any variable Xij, after it's assigned the value v, forward checking will:

1) Go over all the unassigned variables in row = i and remove v from their domain if it exists.
2) Go over all the unassigned variables in column = j and remove v from their domain if it exists.
3) Go over all the unassigned variables in the box of Xij and remove 5 from their domain if it exists. The box can be figured out by taking i mod 3 and j mod 3 to find its place wrt to the box. If i mod 3 = 0: we will take values with i+1 and i+2, if i mod 3 = 1: we will take i-1 and i+1, if i mod 3 = 2: we will take values with i-1 and i-2. The same can be done with j mod 3 to figure out which j values fall into the box.

Variable X64's domain = {5} currently, so it must take the value 5.
After the value is assigned, forward checking will

1) Go over all the unassigned variables in row = 6 and remove 5 from their domain if it exists. These variables would be: X61, X66, X67, X69
2) Go over all the unassigned variables in column = 4 and remove 5 from their domain if it exists. These variables would be: X14, X34, X64, X74
3) Go over all the unassigned variables in the box of X64 and remove 5 from their domain if it exists. These variables would be: X45, X46, X56, X66

I have not specified their domains, but for any variable Xij, if there is any variable with value 5 in row i or there is any variable with value 5 in column j or there is any variable within Xij's little box, 5 will not be in the domain, so we won't have to remove it.

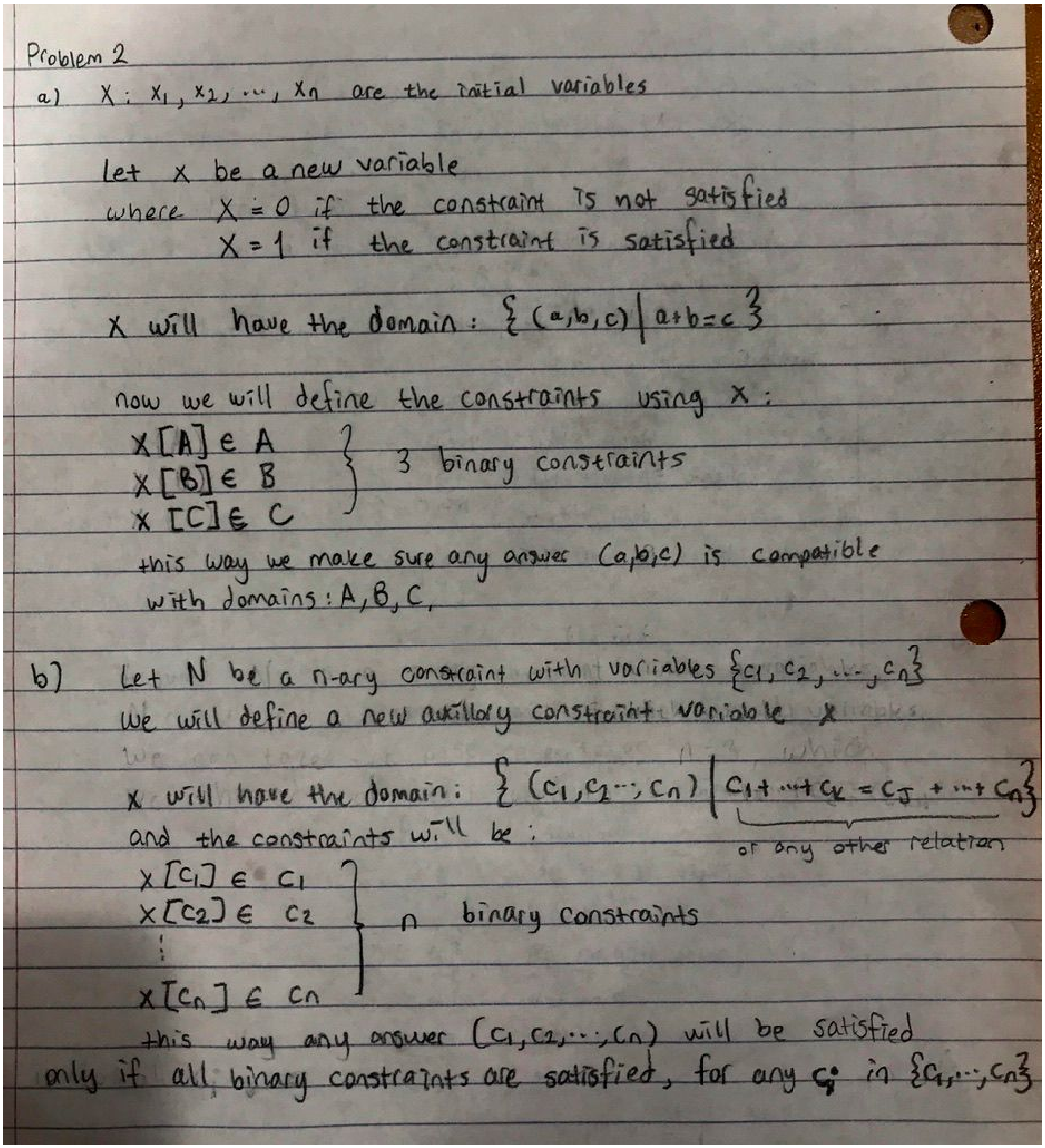
c) Most constrained Variable heuristic selects the variable with the fewest legal values consistent with the current partial assignment. I will assume that Backtracking search starts from X11 (the top left square) and a variable's neighbors are the ones adjacent to it. So after it's assigned a value, we will either pick X12, X21 or X22. But we can't pick X21 since it's already assigned. Let's examine their domains:
D(X12) = {4,5,7,8}
D(X22) = {2,4,6,7,8}
We will pick X12 since it's the variable with the tightest domain.
We will then compute the size of the domains of the new neighbors and again pick the one with the smallest domain.

d) Domain(X48) = {4,5,7}
We can compute the domain by manually checking the row, column, and the little box X48 belongs to. Since it has 7 in it, it can take the value 7.
Forward Checking and Arc consistency will be able to derive it. If we assign X48 the value 7, if it was not a consistent assignment the weight would have turned out to be 0. The weight can be calculated using the constraints specified in part (a). If it turns out to be 0 we would try the next value, and when we run out of values to try we would backtrack to the previous state.

But if 7 is a consistent assignment we would update the domain of all other variables with Arc Consistency. First the neighbors of X48 will be added to the queue and their domains will get updated. If a variable’s domain gets updated, it will be added to queue so we will have to update it’s neighbor’s domains too. This will make sure all inconsistent assignment’s are eliminated from our model. If we start using Arc Consistency from the initial state of the game, at any turn every variable will have a correct domain without any inconsistent values. So we will know if 7 is a consistent assignment or not.

Ege Ersü
Linus Shih
Assignment 4

Problem 2:



Problem 2

a) $X: x_1, x_2, \dots, x_n$ are the initial variables

let x be a new variable
where $X = 0$ if the constraint is not satisfied
$X = 1$ if the constraint is satisfied

X will have the domain: $\{(a,b,c) \mid a+b=c\}$

now we will define the constraints using x:

$$\left.\begin{array}{l} x[A] \in A \\ x[B] \in B \\ x[C] \in C \end{array}\right\} \text{3 binary constraints}$$

this way we make sure any answer (a,b,c) is compatible with domains: A, B, C,

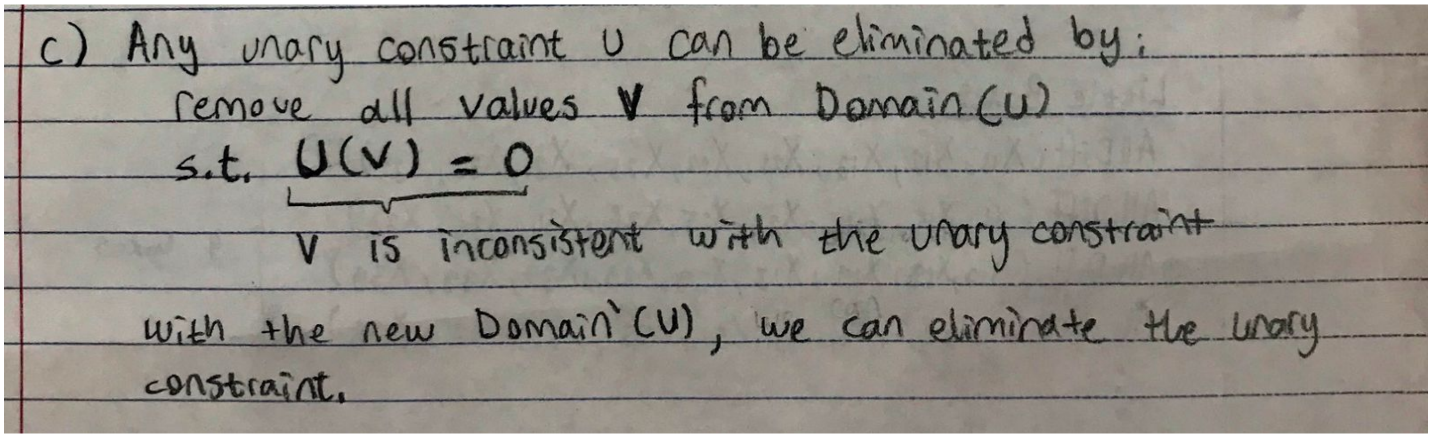
b) Let N be a n-ary constraint with variables $\{c_1, c_2, \dots, c_n\}$
we will define a new auxillary constraint variable x

x will have the domain: $\{(c_1, c_2 \dots, c_n) \mid \underbrace{c_1 + \dots + c_k = c_J + \dots + c_n}_{\text{or any other relation}}\}$
and the constraints will be:

$$\left.\begin{array}{l} x[c_1] \in c_1 \\ x[c_2] \in c_2 \\ \vdots \\ x[c_n] \in c_n \end{array}\right\} n \text{ binary constraints}$$

this way any answer $(c_1, c_2, \dots, c_n)$ will be satisfied only if all binary constraints are satisfied, for any c_i in $\{c_1, \dots, c_n\}$

Ege Ersü
Linus Shih
Assignment 4



c) Any unary constraint U can be eliminated by:
remove all values V from Domain(U)
s.t. $\underbrace{U(V) = 0}_{\text{V is inconsistent with the unary constraint}}$

with the new Domain'(U), we can eliminate the unary constraint.

Problem 4:

We will argue that a constructive approach will be more reliable. The main problem with Local Search is that it only keeps the current node in memory. The only advantage it has is that it will have a smaller space complexity. With a constructive approach we can always go back with backtrack and try another value. But the biggest advantage it has will be arc consistency. After each assignment on variable x, we will actually update the domain of all variable in x's row, x's column and x's 3x3 box. This way we will get rid of multiple solutions thanks to the domain updates after each iteration, whereas in Local Search we have no way of updating domains and eliminating possible solutions. Because of that we will search over unnecessary solutions that could have been prevented through arc consistency. Also we can implement arc consistency in a way such that whenever a variable has 1 value left in its domain, we will assign that value to the variable immediately. These strategies will immensely reduce the solution space whereas local search has no way of eliminating solutions in a systematic way.

These are of course only true under the assumption that the repair algorithm even finds a solution. Since local search is not complete for CSPs, most of the time we will have to restart the algorithm with a different initialization, which will make it an unreliable method. Although it has the potential to find a solution faster than a constructive method, there is no guarantee that it will find it in a reasonable amount of tries. This is due to the fact that whenever a row/column/3x3 box is consistent, it will no longer change any variables in that region. It's rarely the case in Sudoku that your first assignment to a row/column will be a part of the correct solution. It's like trying to solve Rubik's cube by first making one side fully green, then trying to fix the other sides without ever touching the completed side again. Backtracking is a strong tool for problems of this kind.

Ege Ersü
Linus Shih
Assignment 4

Problem 5: The Zebra Problem

Colors = {1: red, 2: green, 3: ivory, 4: yellow, 5: blue}
Nationality: = {1: english, 2: spanish, 3: norwegian, 4: ukranian, 5: japanese}
Candy = {1: hershey bar, 2: kitkat, 3: smarties, 4: snickers, 5: milky way}
Drink = {1: orange juice, 2: tea, 3: coffee, 4: milk, 5: water}
Pet = {1: dog, 2: fox, 3:snails, 4: horse, 5: zebra}

Entry at (Value, Attribute) in this table is the house that has that Value for its Attribute.
Ex: If the entry at (Nation, 3) is 4 it means that house 4's owner is of Nationality: Norwegian.

	Color	Nation	Candy	Drink	Pet
1	A	A	D	G	B
2	C	B	E	H	D-1 or D+1
3	C-1	1	F	C	F
4	E	H	G	3	E+1 or E-1
5	2	J	J	W	Z

Constraint: every column must have an element of (1,2,3,4,5) at least once as a value.
I will update domains by lightly checking every column

	A	B	C	D	E	F	G	H	J	W	Z
initial	1,2,3,4,5	{1,2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}	{1,2,3,4,5}
After color	1,3,4,5	1,2,3,4,5	3,4,5	1,2,3,4,5	1,3,4,5	1,2,3,4,5	1,2,3,4,5	1,2,3,4,5	1,2,3,4,5	1,2,3,4,5	1,2,3,4,5
After Nation	3,4,5	2,3,4,5	3,4,5	1,2,3,4,5	1,3,4,5	1,2,3,4,5	1,2,3,4,5	2,3,4,5	2,3,4,5	1,2,3,4,5	1,2,3,4,5
After Candy	3,4,5	2,3,4,5	3,4,5	1,2,3,4,5	1,3,4,5	1,2,3,4,5	1,2,3,4,5	2,3,4,5	2,3,4,5	1,2,3,4,5	1,2,3,4,5

Ege Ersü
Linus Shih
Assignment 4

After Drink	3,4,5	2,3,4, 5	4,5	1,2,3, 4,5	1,3,4, 5	1,2,3, 4,5	1,2,4, 5	2,4,5	2,3,4, 5	1,2,4, 5	1,2,3, 4,5
After Pet	**3**,4,5	2,3,4, 5	4,**5**	1,2,3, 4,5	1,3,4, 5	1,2,3, 4,5	2,**4**,5	**2**,4,5	2,3,4, 5	1,2,4, 5	1,2,3, 4,5
result	**3**	4	5	2	1	3	4	**2**	5	1	5

Now I will make sure every column only has
Looking at column: color
A != C != C-1 != E
For C=4: (A=5,C-1=3,E=1)
For C=5: (A=3, C-1=4, E=1)

So E = 1 for any assignment A,B
C=4 => A=5
C=5 => A=3
If C=4 and A=5 is chosen, It will be impossible to pick F,B,J due to constraints
Therefore, C=5 and A=3

For column Drink
G != H != C != W
Since C=5, new domains are:
G = {2,4}
H = {2,4}
W = {1,2,4}
Therefore, W = 1
If H=4, G and J must both be 2, due to column Candy.
Therefore H=2 and G=4

For column Nation
A != B != H != J
A = 3
B = {4,5}
H = 2
J = {4,5}
=> j turns out to be 5 => b = 4

For column Candy
D != E != F != G != J
G = 4
D = {2,3}
E = 1

Ege Ersü
Linus Shih
Assignment 4

F = {2,3}
J = 5

For column Pet:
B != D-1 or D+1 != F != E+1 or E-1 != Z
B = {4,5}
D = {1,3}
F = {1,3}
Z = {1,3,4,5}
E = 1 so E+1 = 2
Since b=4, d=2, f=3,e=1 => z=5

F = 3 Because of Candy & Pet constraint
=> D = 2

Question 1: Where does the zebra live?
(Pet,5) = House Z = 5 (rightmost house)
Question 2: In which house do they drink water?
(Drink,5) = House W = 1 (leftmost house)