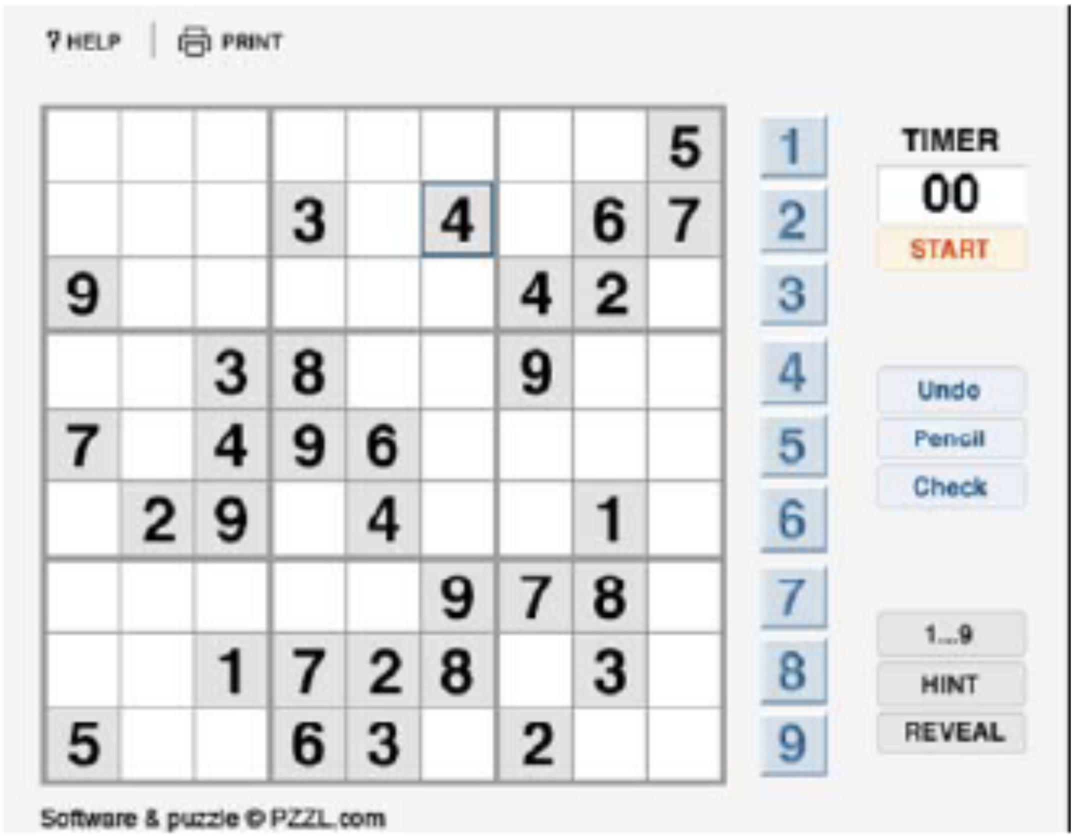
Assignment 4 Report

Saurabh Daptardar, Srikanth Kuthuru

October 18, 2017

1 Problem 1:



								5
			3		4		6	7
9						4	2	
		3	8			9		
7		4	9	6				
	2	9		4			1	
					9	7	8	
		1	7	2	8		3	
5			6	3		2		

Figure 1: Sudoku puzzle

- There are 27 general constraints for any sudoku puzzle [9 for rows, 9 for columns and 9 for sub-blocks]. We also have additional constraints because of the partially filled assignments provided initially.
  - Rows constraint: $set(AllDiff(V_{i1}, V_{i2}, ..., V_{i9}))\forall i = 1, 2, 3...9$
    This results in 9 constraints.

  - Similarly, Columns constraint: $set(AllDiff(V_{1j}, V_{2j}, ..., V_{9j}))\forall j = 1, 2, 3...9$
  - Sub-block constraints: In every subblock. all elements should be different. For example, in top-left subblock, it is $Alldiff(V_{11}, V_{12}, V_{13}, V_{21}, V_{22}, V_{23}, V_{31}, V_{32}, V_{33})$. Similarly for other subblocks.

    This results in 9 constraints.
  - Constraints due to initial assignments: $V_{ij} = v_{ij}$ for all ordered pairs (i, j) for which the initial assignment v_{ij} is given.

- Forward checking with the initial board: Considers an initially assigned variable and looks at the constraints that it is part of, and then changes the domains of all the variables in those constraints accordingly. This process is repeated for all the initially assigned variables. After that, Consider an unassigned variable V_{ij} and let D_{ij} be its domain. We assign a value to V_{ij} from its domain D_{ij} and then look at the constraints that it is part of, and then changes the domains of all the variables in those constraints accordingly [General forward checking procedure].

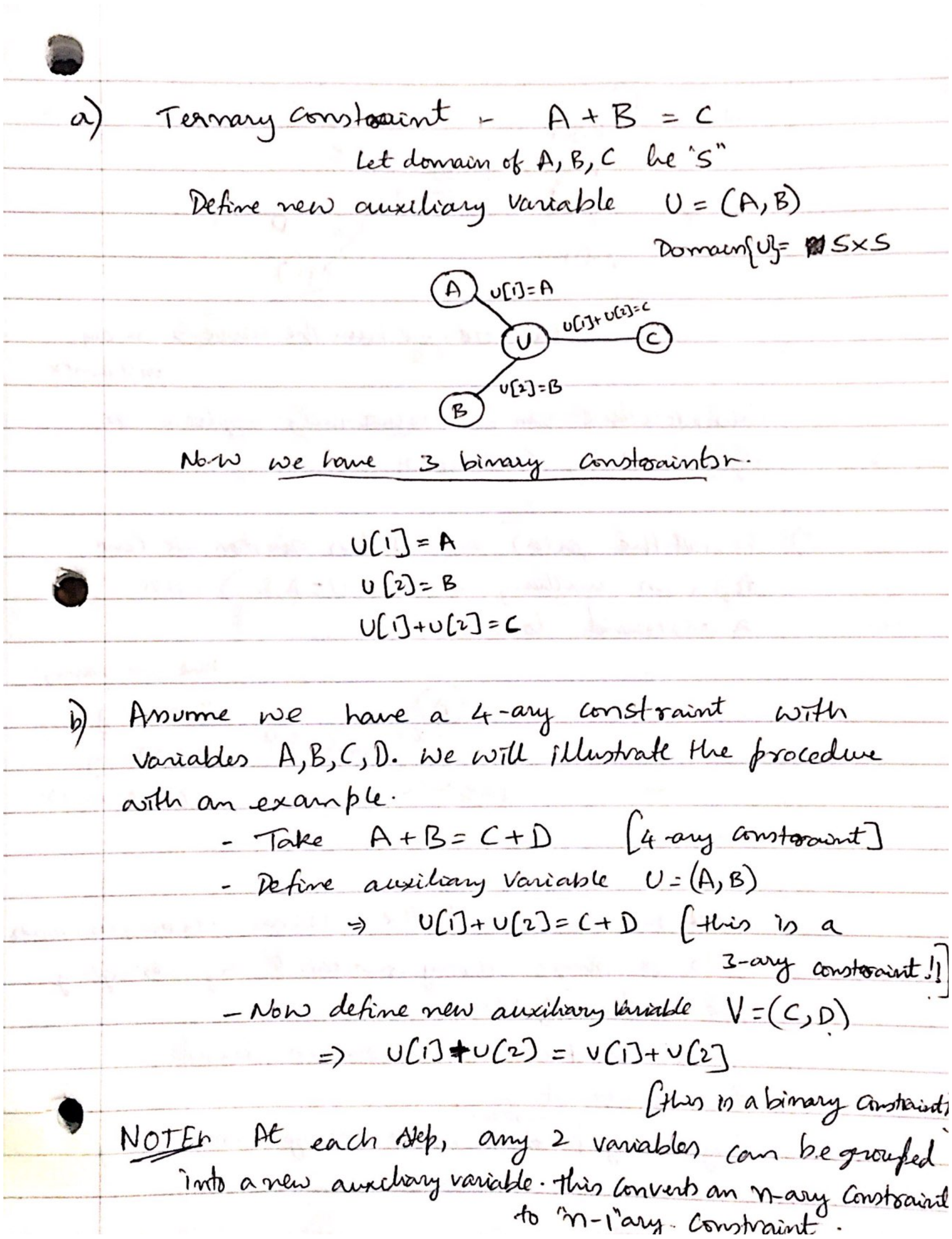
  In the initial board case, after forward checking, the domain of V_{74} becomes 1,4,5.

- Variable ordering heuristic is used to select the variable that needs to be assigned next in the backtracking algorithm. In the sudoku problem, after forward checking is done for all the initial assignments, we choose the variable that has the lowest domain size and then assign it a value from its domain. In our case, the domains D_{64} and D_{58} contain only 1 element which is 5. So, they are assigned that value first and then forward checking is done.

- After the initial forward checking the domain of V_{48} has 3 elements 4,5,7. So, it cannot assign value 7 surely, i.e. it doesn't know that the other constrained squares can't take 7. But, forward checking with arc consistency can find the value 7 because of its iterative domain refinement.

2 Problem 2:

This problem is handwritten in Figures 2 and 3.

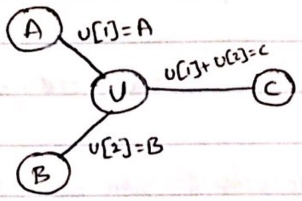


a) Ternary constraint - $A + B = C$

Let domain of A, B, C be "S"

Define new auxiliary variable $U = (A, B)$

Domain{U} = $S \times S$



Now we have 3 binary constraints.

$U[1] = A$

$U[2] = B$

$U[1] + U[2] = C$

b) Assume we have a 4-ary constraint with variables A, B, C, D. We will illustrate the procedure with an example.

- Take $A + B = C + D$ [4-ary constraint]
- Define auxiliary variable $U = (A, B)$

$\Rightarrow U[1] + U[2] = C + D$ [this is a 3-ary constraint!]

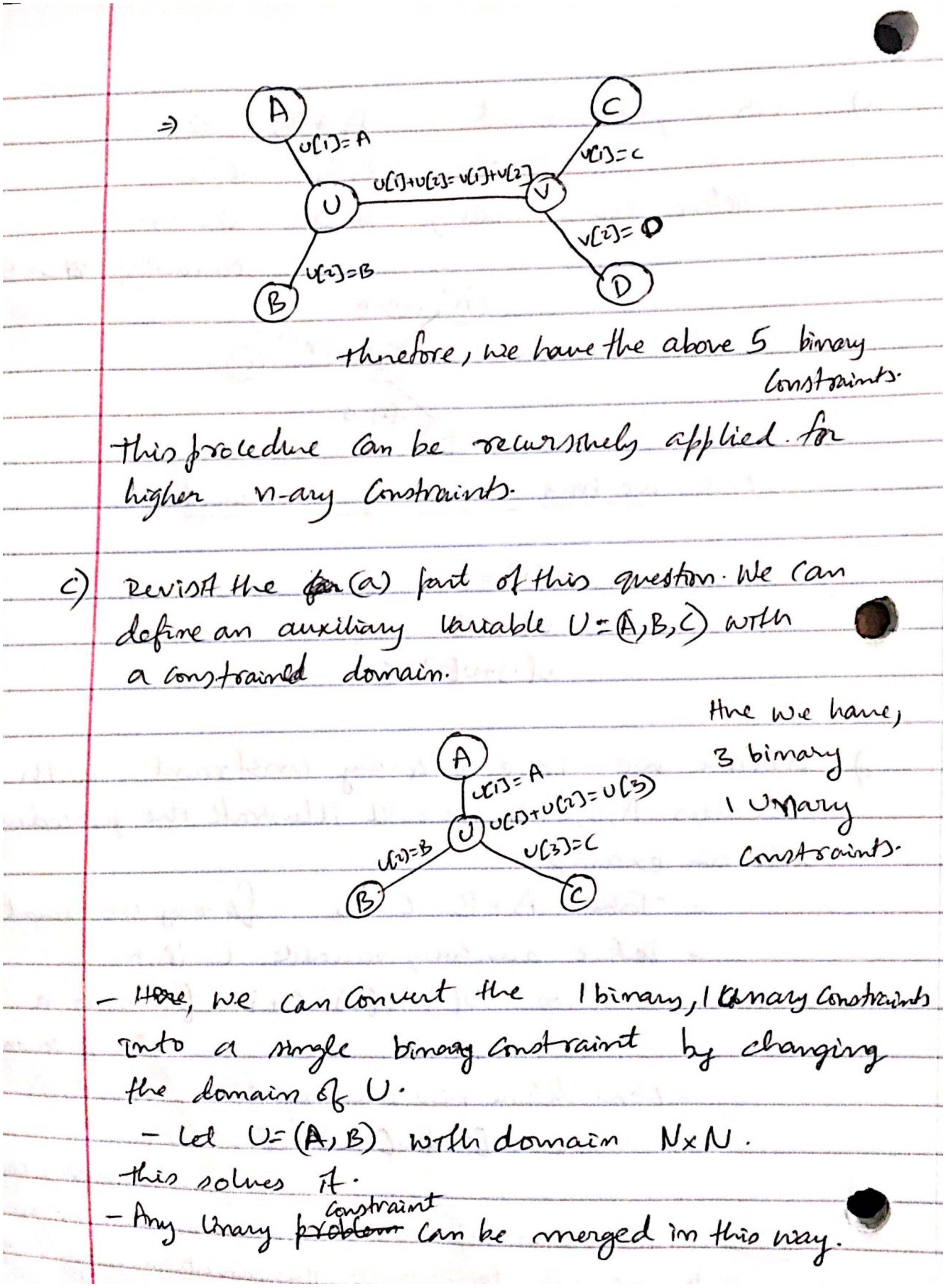
- Now define new auxiliary variable $V = (C, D)$

$\Rightarrow U[1] + U(2) = V[1] + V[2]$

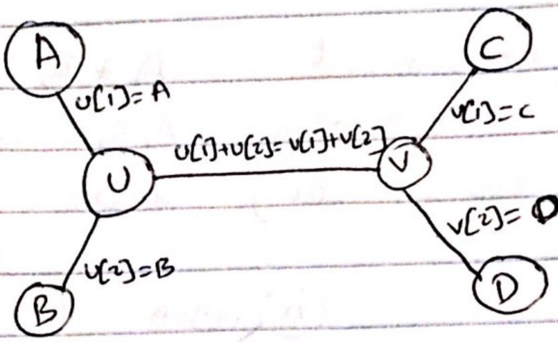
[this is a binary constraint

NOTE: At each step, any 2 variables can be grouped into a new auxiliary variable. this converts an n-ary constraint to "n-1"ary. Constraint.

Figure 2: problem 2 - page1



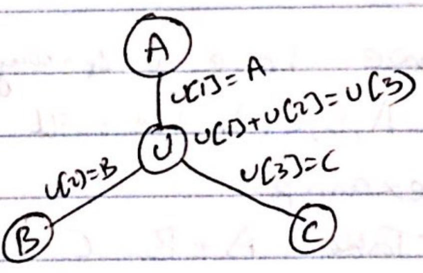
$\Rightarrow$



therefore, we have the above 5 binary Constraints.

this procedure can be recursively applied. for higher n-ary Constraints.

c) Revisit the ~~for~~ (a) part of this question. We can define an auxiliary variable U=(A,B,C) with a constrained domain.



Here we have, 3 binary 1 Unary Constraints.

- ~~Here~~, we can Convert the 1 binary, 1 ~~Un~~ unary Constraints into a single binary Constraint by changing the domain of U.
  - Let U=(A,B) with domain N×N.

this solves it.

- Any Unary ~~problem~~ constraint can be merged in this way.

Figure 3: problem 2 - page2

3 Problem 3:

Our `profile.txt` is given below:

```
# Unit limit per semester. You can ignore this for the first
# few questions in problem 3.
minUnits 6
maxUnits 9

# These are the semesters that I need to fill out.
# It is assumed that the semesters are sorted in chronological order.
register Spr2018

# Courses I've already taken
taken COMP215
taken COMP440
taken COMP502

# Courses that I'm requesting
request COMP600 in Spr2018
request COMP607 in Spr2018
request COMP322 in Spr2018
```

Running the scheduler on `profile.txt` produces a reasonable schedule and the schedule designed is given below.

Semester	Units	Course
Spr2018	1	COMP600
Spr2018	1	COMP607
Spr2018	1	COMP322

Table 1: Best schedule

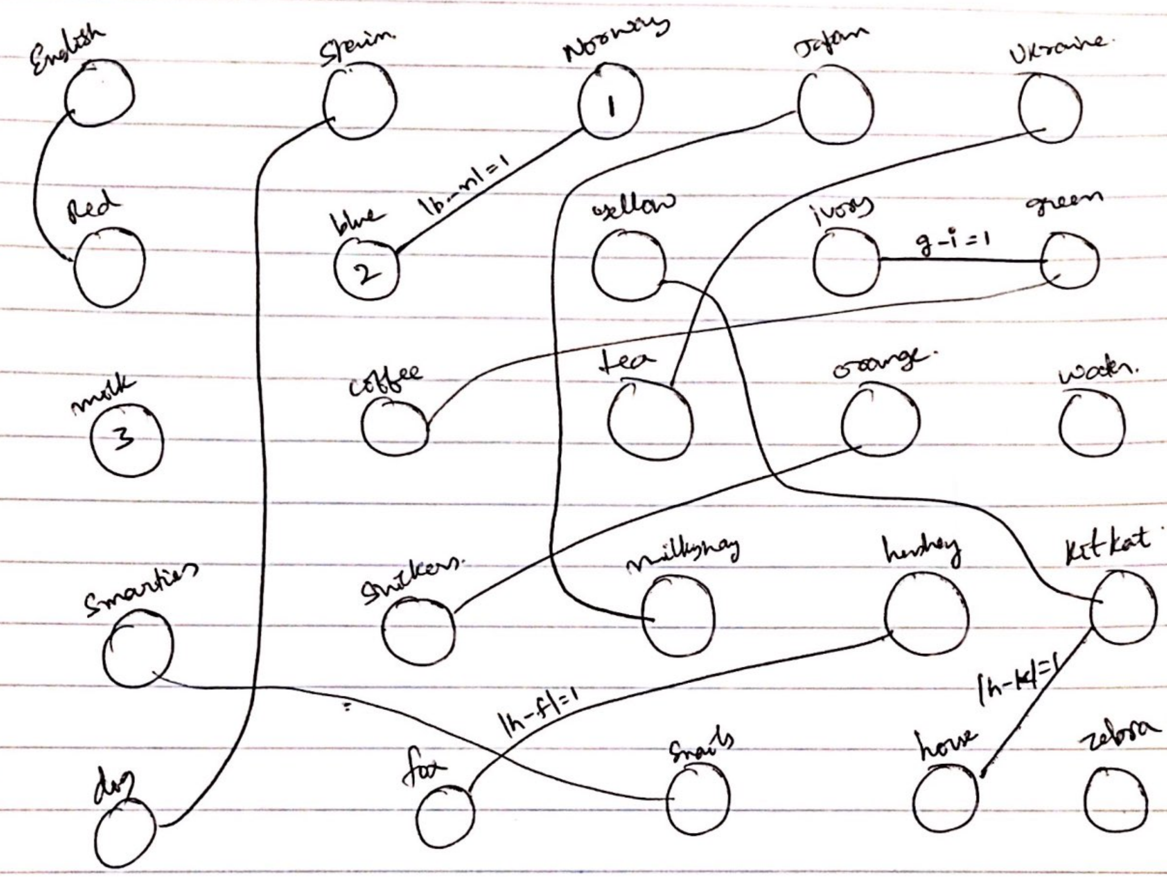
4 Problem 4:

Sudoku can be solved using repair algorithms. Initially, the unassigned variables are assigned to values randomly. Then it randomly selects a variable from the set of variables with conflicts violating one or more constraints of the sudoku problem. Then it assigns to this variable the value that minimizes the number of conflicts. This is called the min-conflicts heuristic. Repair algorithms might get stuck in local minima (wrong solutions). At local minima, changing the variable assignment does not minimize the number of conflicts. Also, the convergence rate highly depends on the initialization. It is shown that greedy initialization methods are better than random initializations.

On the other hand, Backtracking algorithm can always find the solution, if one exists. But it has high computational complexity when the number fo variables in large. Repair algorithms are shown to give solutions very fast for the N-queens problem, even for large N$\geq$10000. Repair algorithms are stochastic methods, and therefore the time complexity doesn't depend on the size of the problem (unlike backtracking algorithm). In a sudoku puzzle, the number fo variables is less, so a constructive approach like backtracking algorithm is a better choice compared to a repair algorithm.

5 Problem 5:

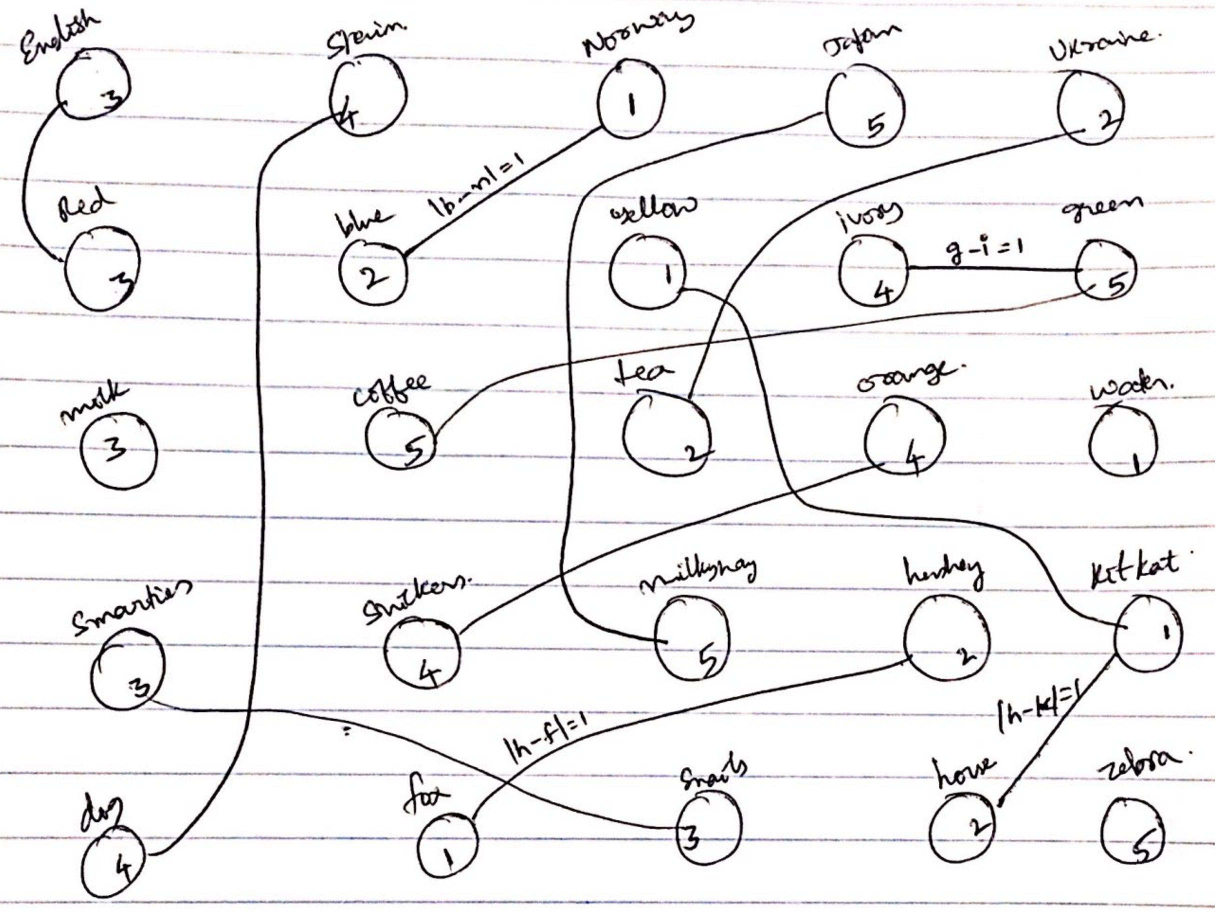
- The water is drunk in the yellow house with norweignian, who eats a kitkat and has a fox. This is house-1 which is the leftmost house. See images for other data.
- The zebra is in the green house with the japanese guy who eats milkyway and drinks coffee. This is the rightmost house. The initial state looks like the Figure 4 and the final state looks like Figure 5.



Houses : {1,2,3,4,5}
↳ leftmost house.

Domain of above variable is {1,2,3,4,5}.

Figure 4: problem 5 - initial state



Houses : {1,2,3,4,5}
↳ left most house.

Domain of above variables is {1,2,3,4,5}.

Figure 5: problem 5 - final state