CSCE 625
Programming Assignment #3

WRITE UP

Choice of representation

In the jobs puzzle, the variables are the jobs i.e. chef, guard, nurse, etc. And the domain for each of these variables is the list of persons i.e. Roberta, Thelma, Steve and Pete. There are two arrays that handle the operations – alive_count and nodes_alive. alive_count array keeps track of the number of possible values that can be assigned to each of the variables. This is necessary to calculate the minimum remaining values possible to assign to a variable so that we can use the MRV heuristic. The nodes_alive array is a two dimensional array that keeps track of the possible values that can be assigned to each of the variables. We manipulate both these arrays to get to the result. In each iteration, the alive count of a particular variable is decreased and the nodes_alive array marks up whichever variable has now been assigned and what all values are remaining. We check the consistency in each iteration. This goes on until all the elements in the alive_count are reduced to having each of their index's value as 1. This means that all variables have values assigned now. If, here, consistency fails then we backtrack and look for other nodes to search. Thus, we keep on checking the consistency function for each node until we find the solution.

Everything is same for the houses or Einstein's puzzle as is for the jobs puzzle. However, the variables here are all the five variables in each of the five segments i.e. eatables, drinks, pets, color of houses and country of people staying there making it a total of twenty-five variables. The domain for each of these variables is limited to the number of values it can take in its segment i.e. five per variable. alive_count and nodes_alive behave similarly as they behaved in the jobs puzzle.

For more details, please check the code. Comments have been added to help understand the working of it much better.

Consistency function

In the jobs puzzle, consistency_jobs is the consistency function. And in the houses puzzle, consistency_houses is the consistency function. In both the puzzles the consistency function checks if the assignment of values to the currently active variables is consistent or not. In consistency_jobs, the function creates a string from the current state and checks that string's indices for characters that match the conditions or not. For example, in the first position of the string, if there is 'R' then it means Roberta has been assigned as Chef. So it checks the conditions according to the positions of the characters in that string. In consistency_houses, similar thing happens. Here the segment 'eatables' takes indices from 0 to 4, drinks from 5 to 9, pet from 10 to 14, color from 15 to 19 and country from 20 to 24. So if character 'E' is present at 20, that means the Englishman is now in the first house in the current state.

Do check the comments in the code itself. They are very helpful.

Solution (variable assignment)

The alive_count array keeps track of the number of possible values that can be assigned to a particular variable. When solution is reached, it means that all the values in alive_count array are now 1. The nodes_alive 2D array keeps track of which nodes are alive and which are dead or cannot be traversed as they may have already been traversed for a particular variable. When the solution is reached, for each of the variables, nodes_alive will have a single value's index as 1.

For example in jobs puzzle, alive_count[0] is for chef. So if alive_count[0] equals 4, then it can take all four values – Roberta, Thelma, Steve and Pete. alive_count[1] is for guard, alive_count[2] for nurse, then clerk, police, teacher, actor and alive_count[7] for boxer. nodes_alive[0][0] means if chef is Roberta, nodes_alive[0][1] means if chef is Thelma, nodes_alive[0][2] means if chef is Steve and nodes_alive[0][3] means if chef is Pete. If nodes_alive[i][j] is 1, it means for the i variable, the value j may be possible.

Similarly, for houses puzzle, alive_count[0] is for eatable in house one (the house which is the left-most). So if alive_count[0] equals 5, then it can take all five values – Smarties, KitKats, Milky Bars, Hershey Bars and Snickers. Note that for 0-4 it is eatables, but from 5-9 in the alive_count array it is drinks. And the possible values that any index in that range can take is from the available drinks. Similarly, 10-14 is for pets, 15-19 is for colors and 20-24 is for country. Note that nodes_alive[i][j] here means for the ith variable position, the jth value may be possible. nodes_alive[0][0] means if smarties is eaten in the first house, nodes_alive[0][1] means if KitKats are eaten in the first house, nodes_alive[0][2] means if Milky Way is eaten in the first house and so on. For nodes_alive[6][0] means if Water is the drink used in second house, nodes_alive[6][1] means if Tea is the drink in the second house and so on.

Eat	{Smarties, KitKats, Milky Way, Hersheys, Snickers}
Drink	{Water, Tea, Milk, Coffee, Orange Juice}
Pet	{Snail, Horse, Fox, Dog, Zebra}
House Color	{Yellow, Blue, Ivory, Green, Red}
Country Men	{Norwegian, Ukrainian, Japanese, Spaniard, Englishman}

Note that the sequence matters here. In both jobs and houses puzzle some evident initial state is taken which satisfies one or more conditions that are very clear and evident. Our initial state is same for both MRV and non MRV running of the same puzzle so that our comparison is proper.

Thus, our nodes_alive array provides us with the current state of assignment of values to variables. We convert it to string and then print the solution in a readable format for the users.

For more details, feel free to check the code comments. They are very helpful.

The number of states searched is in the next page.

Number of states searched

Program 1 (Jobs Puzzle)		
	Without MRV	With MRV
Number of states searched	19	11

Program 2 (Houses Puzzle)		
	Without MRV	With MRV
Number of states searched	43132	1889

For any queries please feel free to call or message me.

Abhinandan Aryadipta (Abhi)
UIN: 925001240
Email: aryarockstar@tamu.edu
Ph: 9799857204