**CS 370 – Project 3: CSPs**

**Due Date:** Sunday, November 17, 2019

**Introduction**

In this project, you will implement constraint satisfaction problem solving algorithms and apply them to map-solving and a problem of your choice. I supply useful classes for Java and Python. If you intend to use a different language, you will have to do without this assistance.

**Included Classes**

* CSP – Represents a general Constraint Satisfaction Problem. Several useful functions are provided, though you will want to extend this class for specific problems in order to specify variables, domains, and constraints.
* Variable – Simple class to represent a variable. Only used in Java. Strings will suffice in Python.
* Domain – Simple class to represent a domain. Uses an array of Objects to represent the acceptable options.
* Constraint – An interface for defining constraints.
* NotEqualConstraint – A class implementing Constraint to represent a constraint requiring that two variables are not equal. Use this class as a guide for implementing your own constraints, if necessary.
* Assignment – A class that assigns values to some or all variables of a CSP.
* ArrayIterator – Class implementing Iterator interface for iterating through an array. Only used in Java.
* SolutionStrategy – An abstract class with one method, solve(), which takes a CSP as input and returns an Assignment. Your code with CSP solver algorithms should extend this class.

**Files to Edit:** None of the given classes should require modification.

**Evaluation:** Your code will be tested on the example CSP problems assigned. Please *do not* change the names of any of the provided functions or classes. Modifying code that you were instructed not to modify is grounds for losing credit on this assignment.

**Getting Help:** You are not alone! If you find yourself stuck on something, contact me for help sooner rather than later. I want these projects to be rewarding and instructional, not frustrating or demoralizing. But I do not know how or when to help unless you ask.

Also, remember, discussion of the assignment is perfectly acceptable. However, this discussion should not extend to sharing code. If in doubt of what would constitute academic dishonesty, contact me.

**Assignment Questions**

**Question 1 (3 points)**

Define the class MapColoringCSP by extending CSP, to represent the task of coloring a particular type of map. The constructor should take two arguments: an adjacency list (that defines the constraints) and a number of colors. (You may assume 2 ≤ colors ≤ 4.)

In the constructor, you should define the variables, domains, and constraints. Use the addVariable(), setDomain(), and addConstraint() methods to attach them to the problem. You may hard-code values (variables, domains, constraints) or read them in from input, whichever you consider easier.

I have provided adjacency lists for Australia and USA.

**Question 2 (4 points)**

Define a class BacktrackingSearch, which extends SolutionStrategy. This class should implement basic backtracking search in the solve() method. For now, your variable selection methods and value ordering methods can be arbitrary. The inference step should do simple forward checking that eliminates values from each neighboring variable’s domain.

### Java instructions

Define a CSPSolver class that contains your main() method. Your code should use command line arguments. For now, running BacktrackingSearch with Australia and America should be defined by the following two commands, where the number represents the number of colors:

>java CSPSolver backtrack aus 3

>java CSPSolver backtrack usa 4

### Python instructions

Define a module cspsolver.py that contains your main() method. Your code should use command line arguments. For now, running BacktrackingSearch with Australia and America should be defined by the following two commands, where the number represents the number of colors:

>python cspsolver.py backtrack aus 3

>python cspsolver.py backtrack usa 4

**Question 3 (2 points)**

Implement the Minimum-Remaining-Values heuristic in BacktrackingSearch. This heuristic should be implemented as a flag that can be turned on/off. If it should be turned on, an additional argument mrv will be supplied as the last command-line argument.

**Question 4 (4 points)**

Implement the Minimum-Conflicts algorithm in a class MinConflictsSearch that extends SolutionStrategy.

The command minconflicts would replace backtrack in the examples above.

>java CSPSolver minconflicts usa 4

>python cspsolver.py minconflicts usa 4

**Question 5 (4 points)**

Now that we have our algorithms in place, we should be able to define and solve any kind of CSP. For example, our algorithms should work on a cryptarithmetic problem (e.g. TWO+TWO=FOUR or SEND+MORE=MONEY). However, we need to define two new types of constraints first. We can do that by implementing the Constraint interface in Java, or extending the Constraint class in Python. Use the given NotEqualConstraint class as a guide.

Define AllDiffConstraint, which should take any number of variables and be satisfied if and only if each of them has a unique value.

Define AdditiveConstraint, which should be able to represent constraints such as:

**Hint**: Make your constructor’s signature something like:

AdditiveConstraint(List<Variable> leftSideVars, List<Integer> leftSideCoefficients, List<Variable> rightSideVars, List<Integer> rightSideCoefficients)

For the example above, the arguments would be ([C10, W, W], [1, 1, 1], [U, C100], [1, 10])

**Question 6 (3 points)**

Now we are ready to define CryptarithmeticCSP as a child class of CSP. As in Question 1, the constructor should define the variables, domains, and constraints. Use the addVariable(), setDomain(), and addConstraint() methods to attach them to the problem.

My command-line input for the problem SEND+MORE=MONEY using backtracking search is given below:

>java CSPSolver backtrack crypt SEND MORE MONEY

>python cspsolver.py backtrack crypt SEND MORE MONEY

Note that I may run this problem using Backtracking Search (with or without MRV) or Min-Conflicts Search.

**What to Submit**

Zip all files, including the files given as part of the assignment, and submit it on Kodiak before the due date. Any submission after the due date is subject to the late penalties described in the syllabus.