15-381 Spring 2007
Assignment 2: Constraint Satisfaction Problems

For questions contact Arthur (atu@andrew.cmu.edu)
and Gil (egjones+@cs.cmu.edu)

Spring 2007
Out: Feb. 6
Due: Feb. 20, 1:30pm Tuesday

The written portion of this assignment must be turned in at the beginning of class at 1:30pm on February 20th. Type or write neatly; illegible submissions will not receive credit. Write your name and andrew id clearly at the top of your assignment. If you do not write your andrew id on your assignment, you will lose 5 points.

The code portion of this assignment must be submitted electronically by 1:30pm on February 20th. To submit your code, please copy all of the necessary files to the following directory:

/afs/andrew.cmu.edu/course/15/381/hw2_submit_directory/yourandrewid

replacing yourandrewid with your Andrew ID. The TAs grading this assignment would prefer that you use C/C++, Java, or Matlab. If you would like to use a different programming language, please give instructions on how to execute your code and comment it very thoroughly or you risk decreased partial credit. Feel free to use standard data structures and algorithms in your chosen language (like the C++ STL). If you use non-standard data structures or algorithms (like someone's internet C queue code) make sure you turn that in as well and indicate in the comments if you did not write a particular piece of code. All code will be tested on a Linux system, we will not accept Windows binaries. No matter what language you use, you must ensure that the code compiles and runs in the afs submission directory. Clearly document your program.

Late Policy. Both your written work and code are due at 1:30pm on 2/20. Submitting your work late will affect its score as follows:

- If you submit it after 1:30pm on 2/20 but before 1:30pm on 2/21, it will receive 90% of its score.
- If you submit it after 1:30pm on 2/21 but before 1:30pm on 2/22, it will receive 50% of its score.
- If you submit it after 1:30pm on 2/22, it will receive no score.

Collaboration Policy.

You are to complete this assignment individually. However, you are encouraged to discuss the general algorithms and ideas in the class in order to help each other answer homework questions. You are also welcome to give each other examples that are not on the assignment in order to demonstrate how to solve problems. But we require you to:

- not explicitly tell each other the answers

- not to copy answers
- not to allow your answers to be copied

In those cases where you work with one or more other people on the general discussion of the assignment and surrounding topics, we ask that you specifically record on the assigment the names of the people you were in discussion with (or "none" if you did not talk with anyone else). This is worth five points: for each problem, you solution should either contained the names of people you talked to about it, or "none." If you do not give references for each problem, you will lose five points. This will help resolve the situtaion where a mistake in general discussion led to a replicated weird error among multiple solutions. This policy has been established in order to be fair to everyone in the class. We have a grading policy of watching for cheating and we will follow up if it is detected.

Problem 1 (20 total points)

Alby-Bach University (ABU) wants to start a new degree program: B.S in Judgment Day Prevention (JDP).

Suppose the degree program is associated with the following courses:

15-211	Fundamental Data Structures and Algorithms
15-212	Principles of Programming
15-381	Artificial Intelligence: Representation and Problem-Solving
15-681	Machine Learning
80-310	Logic and Computation
21-484	Graph Theory
70-122	Accounting
70-311	Organizational Behavior
19-601	Information Warfare

In order to graduate from the degree program, one must complete the following four requirements:

Algorithms Requirement: (**15-211** AND **15-212**) OR (**15-211** AND **15-381**) OR (**15-681** AND **21-484**)

Machine Learning Requirement: 15-381 OR **15-681** OR **80-310**

Communications Requirement: 21-484 OR **70-311** OR **70-122**

Information Warfare Requirement: 15-381 OR **19-601**

In addition, the department imposes the following restrictions:

Information Aggressiveness Restriction: So that they can't make their programs TOO smart, students can take only one class from the set **15-381**, **15-681**, and **19-601**.

Basic Arithmetic Restriction: Students can't take both **15-211** and **70-122**.

Organization Restriction: Students can't take both **21-484** and **70-311**.

Finally, courses cannot be used to count towards multiple graduation requirements - so if you use 15-381 to fulfill part of the Algorithms requirement it can't count towards either the Machine Learning Requirement or the Information Warfare Requirement.

Question 1.1 (5 points)

John Conner just started his junior year at ABU, and needs to graduate as soon as possible. Suppose all he has left to take are JDP required classes. Model the problem of his trying to find a set of classes to satisfy all requirements as a CSP (Hint: the requirements should be your variables). What are the initial domains for each of your variables?

Question 1.2 (8 points)

Show a DFS with backtracking tree for finding a set of classes that fulfill all requirements using a variable order of the requirements in the order they are listed above, and using a value order that selects the lowest department/course number remaining in a variable's domain. Indicate which constraints were violated whenever the DFS needs to backtrack. (Note: to get full credit you must show the full DFS tree and not just the classes that are used to fulfill each requirement).

Question 1.3 (7 points)

Suppose John has already taken **19-601** towards his Information Warfare Requirement and **15-211** towards his Algorithms Requirement. Use constraint propagation to determine other classes he must take to graduate - indicate which requirements the classes fulfill. Can you create a schedule that satisfies all constraints without using search?

Problem 2 (20 total points)

Arthur is looking for a group of friends for his start-up, which develops and provides some web-based p2p downloading solutions to college students (this is before the lawsuits). Arthur has determined that he needs 2 C# Programmers, 2 Flash Designers, 1 Photoshop Guru, 1 Database Admin, and 1 Systems Engineer.

Assume that if a person knows two languages/softwares, he or she can take on two roles in the company.

So Arthurs narrowed down his selections to the following people:

Name	Abilities
Peter	C# and Flash
John	Photoshop and Flash
Jim	Flash and Systems
Jane	C# and Database
Mary	Photoshop and Flash
Bruce	Systems and C#
Chuck	Photoshop and Flash

Question 2.1 (5 points)

Suppose Arthur knows C#, and only has funds to hire three more people.

Model this scenario as a CSP - (using variables, value domains, and constraints).

Question 2.2 (6 points)

Suppose Arthur decides to make Jim a co-founder. Arthur and Jim discover that all the developers absolutely refuse to abandon their favorite platforms, and that they can only afford two single-booted workstations.

Name	Abilities	OS
Arthur	C#	Windows
Peter	C# and Flash	Windows
John	Photoshop and Flash	Windows
Jim	Flash and Systems	FreeBSD
Jane	C# and Database	FreeBSD
Mary	Photoshop and Flash	Linux
Bruce	Systems and C#	Linux
Chuck	Photoshop and Flash	Windows

What are the domains for the two remaining positions after constraint propogation?

Question 2.3 (9 points)

Assume Arthur and Jim hired Bruce and Mary and they have awarded a contract for their first project, a rush job due Friday at 5 PM. It's Monday at 9 AM and they have to put in 50 total hours of work on it by the due date. There are a number of constraints associated with their work schedules:

- They only have access to two machines of the requisite platform, and only have access to those two machines between 9 AM and 5 PM every day.
- Each person can work a maximum of 20 hours over the course of the week.
- A work session by a single person on a particular machine can last no fewer than two hours.
- Arthur cannot work from 12-4 PM Tu/Th.
- Jim can't work MWF 9-12.
- Bruce can only work between noon and 2 PM every day.
- Mary can only work Thursday and Friday.

Model this scheduling problem as a CSP. Indicate how the indicated constraints impact the domains for all variables.

Problem 3 (50 total points)

A robot is tasked with delivering packages to various locations by certain deadlines associated with package delivery. The robot begins at a given position, and receives a list of package locations and their associated deadlines. The robot must determine an order to visit the locations to deliver each package before its associated deadline.

Assume that time begins at 0, that the robot can move in straight lines between delivery locations (thus a Euclidean distance metric is appropriate), that the robot can deliver packages instantaneously upon arriving at the delivery location, and that the robot can move 1 distance unit per cycle.

For this problem you'll be using depth-first search with backtracking to determine robot schedules that satisfy the package deadline constraints.

There are five scenario files associated with this exercise (scenario1.txt, scenario2.txt, etc..).

The format of each scenario file is

```
<robot initial x location> <robot initial y location>
<number of delivery locations>
<delivery x location> <delivery y location> <package deadline>
additional entries for each package
```

All values are floats except the number of delivery locations, which is an int.

Question 3.1 (30 points)

For this question you will implement uninformed depth first search with backtracking. Your search tree should be rooted at the initial robot location, and each level of the search tree should correspond to adding an additional delivery location to the path the robot takes from the start location. You must keep track of when the robot arrives at each delivery location and compare it to the location's deadline - your DFS should backtrack if the path violates a deadline constraint.

The successor function you should use for this question should use the order that the delivery locations are listed in the scenario file. In other words, if you label the delivery locations in order starting at 1, your successor function should select the lowest numbered city not already included in the tour. For this exercise make sure your successor function is aware that it would be counterproductive to visit a location in the path more than once.

In addition to submitting your code, please submit an executable called 'dfsb' that takes a single argument specifying the scenario file, and outputs the full ordered path if one is found, the arrival times at each city in the path, and the number of evaluations performed in generating the path. Additionally, in your writeup include the following:

1. The first 10 partial paths generated in your search (i.e. "My search first considered the path $S \rightarrow 1$, then $S \rightarrow 1 \rightarrow 2$, then $S \rightarrow 1 \rightarrow 2 \rightarrow 3$ violated the deadline of location 3, so it then considered $S \rightarrow 1 \rightarrow 2 \rightarrow 4$". You don't need the text, just the sequences) for `scenario1.txt`. Label the delivery locations starting with 1.
2. For each scenario, report whether or not a satisfying solution was found. For `scenario1.txt`, give the ordered path and the arrival time at each location in the path. For the other scenarios if a solution was found just give the ordered path.
3. For each scenario report the number of evaluations performed in the search. An evaluation is performed any time the constraints associated with a partial path are checked for satisfaction.
4. If any scenario is taking more than 500,000 evaluations you can stop your program and indicate that 500,000 evaluations were exceeded in lieu of an answer.

Question 3.2 (5 points)

Propose a variable ordering strategy for the successor function for DFS with backtracking. Explain why you think your ordering strategy is reasonable for this domain.

Question 3.3 (15 points)

Implement your chosen variable ordering method in your DFS with backtracking code from question 3.1. This should only require changing your successor function. Report the same values requested in section 3.1 (first 10 partial paths evaluated and satisfying solutions with arrival times for `scenario1.tex`, ordered paths for scenarios 2-5, number of evaluations for each scenario).

In addition to submitting your code, please submit an executable `dfsbvo` that takes a single argument specifying the scenario file, and outputs the full ordered path if one is found, the arrival times at each city in the path, and the number of evaluations performed in generating the path.

Write a paragraph comparing the performance with and without variable ordering. Will your variable ordering method always perform better than uninformed search in terms of evaluations?