# Data Structures and Algorithms
COMP 251, Winter 2013

## Assignment 2

**Due date: Monday, February 11, 2013**
**6pm**

All coding for the assignment must be in Java. All code should be well-commented, in a professional style, with appropriate variables names, indenting, etc. Your code must be clear and readable. **Marks will be very generously deducted for bad style or lack of clarity.**

1. The boolean satisfiability or *SAT* problem is notoriously difficult to solve. The *2SAT* problem, however, is a special case of SAT that is solveable with techniques you know (or will know shortly).

   The 2SAT problem consists of boolean expression in a special kind of *Conjunctive Normal Form* (CNF). This form structures a boolean expression as a big conjunction (*and*) of individual disjunctions (*or*s). In 2SAT we have the additional constraint on our CNF representation that each individual disjunction is an *or*-expression of exactly 2 boolean variables (known as 2-CNF form). For example, the following is a 2-CNF representation of a problem in 2SAT:

   $$(a \vee b) \wedge (c \vee \neg a) \wedge (\neg d \vee e) \ldots$$

   Note that variables in a disjunction can appear in either positive or negative form.

   The 2SAT problem is to find an assignment of boolean values to each of the variables in such an expression that makes the entire expression true (or declare that no such assignment exists).

   A trivial way to solve 2SAT problems is to just use brute force: try each possible assignment of boolean values to variables, and evaluate the entire expression.

   A more clever approach exists, however, but requires first converting the 2-CNF form into an *implicative* form, using the following identities:

   $$(a \vee b) \equiv (\neg b \Rightarrow a), \text{ and as well } (a \vee b) \equiv (\neg a \Rightarrow b)$$

   For example, converting the 2-CNF example expression above results in:

   $$(\neg b \Rightarrow a) \wedge (\neg a \Rightarrow b) \wedge (a \Rightarrow c) \wedge (\neg c \Rightarrow \neg a) \wedge (\neg e \Rightarrow \neg d) \wedge (d \Rightarrow e) \ldots$$

   Once that is complete, the following steps will compute a solution or determine there is none:

   (i) Build an *implication graph* $G$. For each variable $a$, create two nodes in $G$; one representing $a$, and one representing $\neg a$. Add directed edges in $G$ between the nodes representing $\alpha$ and $\beta$ if there exists an *implication* $\alpha \Rightarrow \beta$.

   (ii) Find all *strongly connected components* $S$ of $G$. Note that each strongly connected component implies an implication chain between any two nodes it contains. Thus, trivially, if both $a$ and $\neg a$ are in the same SCC then there is no solution to the original 2SAT problem.

   (iii) (Optional) To find an actual solution, form a condensed graph $C$, consisting of nodes representing each $s \in S$, with a directed edge between components if one component is reachable from the other. This should be a *DAG*.

   (iv) (Optional) Extract nodes from $C$ in topological order. Each node/variable in a component is going to be given the same truth value. So, as you extract $c \in C$, if $c$ does not have a truth value then assign it false. This makes any component containing the complementary of a node/variable in $c$ have the value true.

(v) (Optional) Once all nodes in $C$ have been processed you should have a solution.

Your tasks is to process an input 2SAT problem and output "yes" or "no" depending on whether it is solveable or not. Do the following:

(a) Implement a brute force solution to 2SAT. **5**

(b) Implement the first 2 steps (i and ii) of the above, more efficient solution based on computing the SCC of an implication graph. Note that you only need to determine whether a solution exists or not, you do not need to actually output the assignment (ie the last 3 steps of the algorithm description are optional). **10**

(c) Under what conditions is your brute force solution better in practice than your more efficient one? Give appropriate evidence. Note that for this you will need to do some experimentation, varying both the number of variables under consideration as well as the number of disjunctions. **5**

A program for generating arbitrary sample 2SAT instances is available in *MyCourses*, as is a basic template code for reading that output that you can use as a template for your implementations. The template allows for branching based on a `-nobrute` flag, so you can use the same codebase for both solutions.

2. A basic queue has 3 operations: *makeQueue*, *enqueue* and *dequeue*. Queues maintain a FIFO property—first-inserted, first-out.

   Suppose you want to implement a queue, but have only available a stack. Recall, a *stack* supports 3 operations, *makeStack*, *push* and *pop,* and acts in a LIFO (last-in, first-out) manner. Each stack operation is $O(1)$.

   (a) Describe in pseudo-code an implementation of a queue based on using 2 stacks. You may use an additional $O(1)$ space if you need. Note that you do not need to actually implement this! **10**

   (b) The time bounds in your analysis be improved by amortization. Come up with an appropriate potential function and prove that your implementation enables each queue operation to be performed in $O(1)$ amortized time. **10**

3. Suppose you have an undirected graph $G = (V, E)$, where $n = |V|$, and $n$ is even. Prove that for all $n \geq 2$, if every $v \in V$ has *degree*$(v) \geq n/2$ then $G$ is necessarily connected. **10**

## What to hand in

Submit your assignment to *MyCourses*. Note that clock accuracy varies, and late assignments will not be accepted without a medical note: **do not wait until the last minute**. Assignments must be submitted on the due date **before 6pm**.

Where possible hand in only **source code** files containing code you write. Do not submit compiled binaries or .class files. For the written answer questions submit either an ASCII text document or a .pdf file *with all fonts embedded*. Do not submit .doc or .docx files. Images (plots or scans) are acceptable in all common graphic file formats.

Note that for written answers you must show all intermediate work to receive full marks.

This assignment is worth 6% of your final grade. $\overline{\textbf{50}}$