Assignment 4 – Due Feb 23 at 11:30pm

Please submit your homework via BruinLearn. The Python file should be named **hw4.py**. The answers to the questions should be given in a file named **hw4.txt**.

In this problem, you will write a Python program that converts graph coloring problems into SAT problems and use a SAT solver to solve them. We have broken the task into multiple functions and provided you with some basic code for file I/O and for gluing all the functions together (see **hw4_skeleton.py**).

For each graph coloring problem, each node will be represented by a positive integer (node index). Each color is also represented by a positive integer (color index).

Similarly, for a SAT problem, each variable is represented by a positive integer (variable index). As a result, a positive integer is used for a positive literal and a negative integer is used for a negative literal. A clause is simply a list containing the literals of the clause. A CNF formula is then simply a list of clauses.

For example, if variable a has index 1, variable b has 2, and variable c has index 3, then the clause $a \vee \neg b \vee \neg c$ is represented as the list `[1, -2, -3]`. The CNF formula $(a \vee b \vee c) \wedge (\neg a \vee b \vee \neg c)$ is represented as `[[1, 2, 3], [-1, 2, -3]]`. Of course, the order of clauses and literals in each clause does not matter.

Here are your tasks:

1. Write a function `node2var(n, c, k)`. This function should return the index of the propositional variable that represents the constraint: "node n receives color c " (with k colors being considered). Use the following conversion convention:

   $$\text{variable index} = (n - 1) \cdot k + c.$$

2. Write a function `at_least_one_color(n, k)`. This function should return a clause that represents the constraint: "node n must be colored with *at least* one color whose index comes from the set $\{1, 2, \ldots, k\}$."
3. Write a function `at_most_one_color(n, k)`. This function should return a list of clauses that represents the constraint: "node n must be colored with *at most* one color whose index comes from the set $\{1, 2, \ldots, k\}$."

4. Write a function `generate_node_clauses(n, k)`. This function should return a list of clauses that constrain node n to be colored with *exactly* one color whose index is in the set $\{1, 2, \ldots, k\}$.
5. Write a function `generate_edge_clauses(e, k)`. An (undirected) edge e is simply a tuple of two node indices x, y. This function should return a list of clauses that prohibit nodes x and y from having the same color in the set $\{1, 2, \ldots, k\}$.

After finishing all the above parts, you should be able to convert a graph coloring problem into a SAT problem. To do so, call

```
graph_coloring_to_sat(graph_fl, sat_fl, k)
```

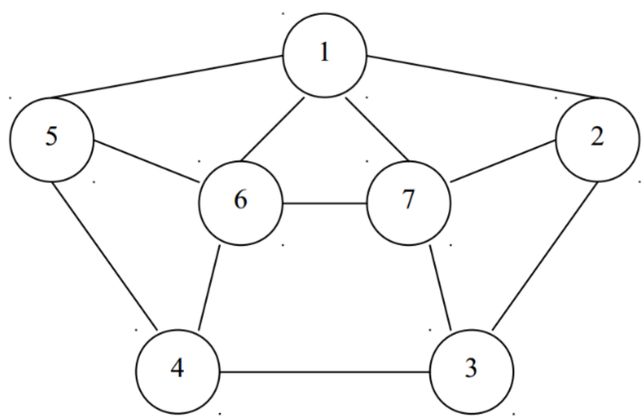
`graph_fl` is the filename of the input graph file (e.g. "graph1.txt"). `sat_fl` is the name of the output file you want the program to write to. `k` is simply the number of colors being considered in the problem.

The graph file has the following format:

- The first line contains 2 numbers. The first one is the number of vertices in the graph; the second one is the number of edges.
- Each subsequent line describes an edge. An edge is represented by two numbers —the node indices of the two nodes linked by the edge.

Now that you have a converting program working, you will use it to convert some actual graph coloring problems into SAT problems and solve them with a SAT solver.

First, consider the following graph (whose nodes are labeled with their node indices):



A graph file for this graph is also provided (graph1.txt). Convert the graph coloring problem of this graph with 3 colors into a SAT instance using the program you wrote.

Then, download the RSat SAT solver from (http://reasoning.cs.ucla.edu/rsat/). Read the manual carefully. Use RSat to solve the SAT instance obtained above. **Is the instance satisfiable?**

Do the conversion again, this time, with 4 colors. Use RSat to solve this new SAT instance. **Is the instance satisfiable?**

What do the answers of these two SAT instances tell you about the graph coloring problem of the above graph? Can you give a solution (a coloring) to the graph coloring problem of the above graph based on the results of RSat?

Now, use a similar approach to solve the graph coloring of the graph described in graph2.txt. **What is the minimum number of colors required to properly color this graph?**