cpts350 Symbolic graph project

0. Make yourself be familiar with Python and `pyEDA` package (see the email that I sent earlier this week and read the example code in the documentation of the package). You may find installation instructions at

`https://pyeda.readthedocs.io/en/latest/install.html`

1. Look at your class notes on how a graph is represented in a Boolean formula and then a Boolean formula is represented in BDD, and on how the transitive closure is computed, in particular, looking at the example of computing the transitive closure of $R \circ R$.

2. Let G be a graph over 32 nodes (namely, node 0, $\cdots$, node 31). For all $0 \le i, j \le 31$, there is an edge from node i to node j iff $(i + 3)\%32 = j\%32$ or $(i + 8)\%32 = j\%32$. (% is the modular operator in C; e.g., 35% 32=3.) A node i is `even` if i is an even number. A node i is `prime` if i is a prime number. In particular, we define [`even`] as the set $\{0, 2, 4, 6, \cdots, 30\}$ and [`prime`] as the set $\{3, 5, 7, 11, 13, 17, 19, 23, 29, 31\}$. We use R to denote the set of all edges in G.

3. (graded on correctness and clarity. If you use explicit graph search such as DFS, you receive 0.) (coding in Python) Every finite set can be coded as a BDD. Please write a Python program to decide whether the following is true:

(StatementA) for each node u in [`prime`], there is a node v in [`even`] such that u can reach v in even number of steps.

You code shall implement the following steps.

step3.1. Obtain BDDs RR, $EVEN$, $PRIME$ for the finite sets R, [`even`], [`prime`] , respectively. Pay attention to the use of BDD variables in your BDDs.

step3.2. Compute BDD $RR2$ for the set $R \circ R$, from BDD RR. Herein, $RR2$ encodes the set of node pairs such that one can reach the other in two steps.

step3.3. Compute the transitive closure $RR2star$ of $RR2$. Herein, $RR2star$ encodes the set of all node pairs such that one can reach the other in even number of steps.

step3.4. Compute the BDD PE, from BDDs $PRIME$, $EVEN$, and $RR2star$, that is to encode the set of all node pairs (u, v) such that u is prime and v is even and u can reach v in even number of steps.

step3.5. Here comes the most difficult part. You need formulate StatementA in terms of BDD operations on the BDD PE. There are two quantifiers in StatementA: one is "for each", and the other is "there is". First, from what you have learned from math216 (discrete math), "for each" can be expressed through "there is". Second, "there is" can be implemented using existential quantifier elimination method $BDD.smoothing()$. As a result, the entire StatementA is a BDD without free variables and hence it is either true or false. Return the truth vlue.

Many students find methods $BDD.compose()$ and $BDD.smoothing()$ are quite useful in the package.

4. (This part takes 10 pts that will be added to your existing midterm score) Code with BDD is extremely hard to test. Here is a way to test the key part (step3.4) of your code.

(a). I pick a prime number, say, $u = 5$. By manually going through the definitions in 2 above, can you tell me an even v in StatementA? Please show me the steps. Using the v that you identified, you test your step3.4 using the u and the v; i.e., you verify (by writing test code) that the (u, v) does satisfy your PE in step3.4.

(b). I pick a prime number, say, $u = 5$. I also pick an arbitrary even number, say, $v = 8$. Can you write a test code to check whether the (u, v) does satisfy your PE in step3.4. ? You can try many such pairs of (u, v) with u being a prime and v being an even and tell me your conclusion.

5. You need turn-in working code, screen-shot of code execution results. Make sure that you put comments along with your code so it is readable. Your TAs will probably run your code!