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 $0 \leq i, j \leq 31$, 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:

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

(Important: your code shall first encode R, [`even`], [`prime`] in BDDs using `pyEDA` and then using methods provided with the package, implement the iteration algorithms for transitive closure computation symbolically in BDD using methods in the package. Many students find methods $BDD.compose()$ and $BDD.smoothing()$ are quite useful in the package.)