Cpt S 350 BDD project

Please print your name!

No late homework!

Before you start, pls read. Download PerlDD at

http://vlsi.colorado.edu/~fabio/

and install it. I have tried a number of BDD implementations, and this is probably the easiest, however, as always, it is also poorly documented. Please read test.pl file under the directory PerlDD-0.09 after you install the package, in particular, read parts surrounding the key word "preimage" (by the way, I invented the word many years ago in my dissertation) in the code test.pl. If you believe that you can understand what's going on, you may start right away to write the code. The entire code should take no more than 100 lines excluding I/O. If after reading these, you find it is way above your head to write working code, you may still be able to turn in psue-do perl code.

You TA told me that one can also do it in Python, by using Python EDA. I will report to you when she finishes. If you can do it in Python, that is very okay.

Happy coding!

This project will take 30% of your entire HW grades towards the course letter grade.

==

Let G be a directed graph on nodes $0, \cdots, n$ and take $k = \lceil \log n \rceil$. G is given in a ascii file `graph.txt` where each line encodes an edge that is a pair of two distinct numbers separated by a space:

```
3 5
7 4
```
$\vdots$

Let $\mathbf{x} = (x_1, \cdots, x_k)$ be a tuple of k Boolean variables. Each node i uniquely corresponds to a k-bits vector $\mathbf{v}^i$ that is the k-bits unsigned number representation of number i. For instance, if $k = 5$ and $i = 3$, then $\mathbf{v}^i$ is $(0, 0, 0, 1, 1)$. In this way, an edge (i, j) corresponds to a $2k$-bits vector $(\mathbf{v}^i, \mathbf{v}^j)$. Let $\mathcal{G}$ be a Boolean formula on $2k$ Boolean variables $\mathbf{x}, \mathbf{y}$ such that, for all i, j,

(i, j) is an edge of G iff $\mathcal{G}(\mathbf{v}^i, \mathbf{v}^j)$ is true.

Step1. From the given file `graph.txt`, build a BDD $\hat{\mathcal{G}}$ for the Boolean formula $\mathcal{G}$.

Step2. Write a function

    Boolean FiveStepReach(i,j)

to decide, given two nodes i and j, whether i can reach j in 5 steps. Your program is based on BDD operations on $\hat{\mathcal{G}}$.

Step3. Test your code in step2 on

(a). The simple graph of $(0, 1), (1, 2), (2, 3), (3, 0)$ with $i = 0$ and $j = 0$,

(b) A random graph (you generate) with 1024 nodes and $i = 0$ and $j = 1$.