INDIAN STATISTICAL INSTITUTE

MTech(CS) I year 2022-2023

Subject: Computing Laboratory

Assignment 2

Due date: 09:00AM, November 28, 2022 Total: 30 marks

Q1. *Binary Decision Diagrams.* (30 marks)

This problem is an extension of Q2 of Lab Test 2. Recall that you will be given a text file (say `input.txt`) containing the truth-table for a k-ary Boolean function f (the value of k will also be given to you). In Lab Test 2, you were asked to construct a simplified Binary Decision Diagram (BDD) for f. For this question, you need to generate even more compact representations of the BDD for a given f using the following reduction rules. Figure 1 shows how the reduction rules are applied in order to get the reduced BDD.

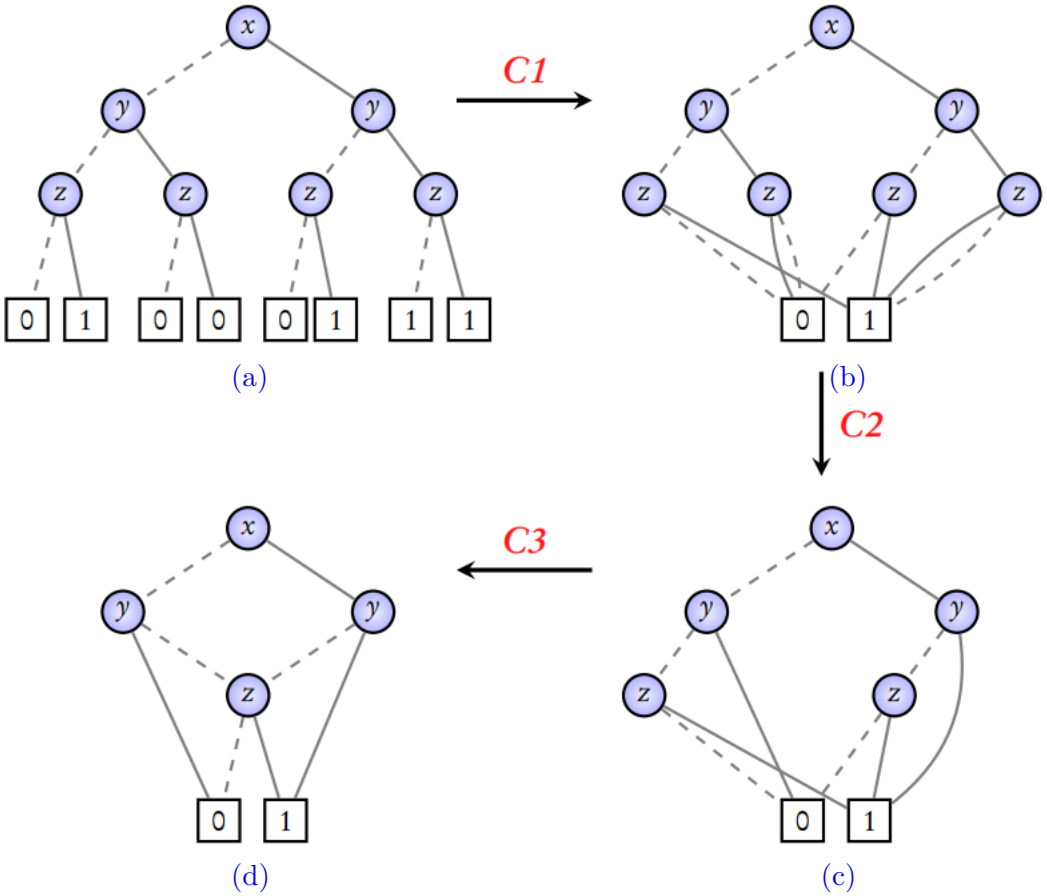

Figure 1: Applying reduction rules C1, C2 and C3 to a BDD

C1. *Removal of duplicate leaves.* Completed in Q2 of Lab Test 2.

C2. *Removal of redundant tests.* If the left and right child links of a particular internal (or "decision" node) both point to the same leaf (or "terminal" node), then that decision node may be eliminated, since its value (0 or 1) does not affect the function value. From Figure 1b, two nodes labeled z have been removed to obtain Figure 1c, since both the children of the first z node were 0-terminal nodes, and for the second z-node, both the children were 1-terminal nodes.

C3. *Removal of duplicate sub-trees.* If two different decision nodes at the same level are roots of *isomorphic* sub-trees, then those two decision nodes may be merged. For example, in Figure 1c, the sub-trees rooted at the two z nodes are isomorphic; in Figure 1d, they have been merged. **In general, applying reduction rule C3 once may create additional opportunities for applying C3, so C3 should be applied iteratively until no further simplications are possible.**

Write two additional functions, namely, `reduction_rule2`(), `reduction_rule3`() to implement reduction rules C2 and C3. Call `remove_duplicate_leaves()` (from Lab Test 2), `reduction_rule2`() and `reduction_rule3`() in that order from `main`. Print the reduced version of the BDD obtained after applying C2 and C3.

Please use the same input and output formats, as well as the `typedef`s specified earlier. For your submission, you may re-use code any code that you wrote during the examination, or code from the solution provided.

Q2. *Huffman coding.* (30 marks)

(a) Given a text file containing only printable ASCII characters, count the number of times each character occurs in the file.

(b) Based on the occurrence frequencies of the characters, construct a Huffman tree for the characters.

(c) Using the tree, encode the contents of a second text file, and write it to an output file.

(d) Decode the contents of the output file, and verify that you have been able to successfully regenerate the input file.

Input/output format. Your program should take four filenames as command-line arguments, as shown in the example below.

```
$ ./a.out training-text.txt test.txt test-encoded.hm test-decoded.txt
```

Use `training-text.txt` in part (a) to compute occurrence frequencies. For part (c), use `test.txt` and `test-encoded.hm` as the input and output, respectively. For part (d), `test-encoded.hm` should be the input, and the output of the decoder should be written to `test-decoded.txt`. If your encoder and decoder work correctly, and you issue the following command after running your program as specified above:

```
$ diff -s test.txt test-decoded.txt
```

you should get something like `Files test.txt test-decoded.txt are identical` as the output of the `diff` command.

NOTE: Some sites like `https://www.hackerrank.com/challenges/tree-huffman-decoding` and `http://www.geeksforgeeks.org/greedy-algorithms-set-3-huffman-coding/` contain fully worked out solutions. If Moss determines that your code is significantly similar to such online solutions, you will not get any marks for this question.