EDA Lab 1

1. Overview:

Binary Decision Trees is a data structure that is mainly used to represent ‘Boolean Functions’. This data structure for n number of variables takes around 2^n nodes. So the **Reduced Ordered Binary Tree** is created using Shannon’s Expansion to optimize memory usage.
In this lab assignment, functions such as Build, Make, Apply, Restrict, SatCount and AnySat are implemented.

Make and Build Function:

A. Build ROBDD

1. We create two tables T(u -> idx, low, high) and H(reverse of T idx, low, high -> u)
2. Using **Make** and **Build** functions to create ROBDD.
    - Make function is necessary to reduce the tree simultaneously, as we create it.
    - Make function takes i,l,h as input values are checks whether these values are present in the reversed table H or not or else create an entry.
    - Build function creates a tree from a boolean expression and at the same time reducing it.

3. Procedure Followed:
    - A package ROBDD is created and all the supporting classes such as ROBDD, Apply, BooleanParser, etc, are stored in this package.
    - The build operation is performed at the beginning to create a ROBDD.
    - This function takes an expression as an input parameter.
    - Boolean Parser class has a function that takes expression in the form op(bool1, bool2) and returns a true or false value.
    - Table T and H are two tables created using HashMap.
    - T has u index as the key and a T_table class object as the value, whereas H has a hash value of (i,l,h) as the key and u index as the value.

Test Cases:

1. A simple test case with three variables

```
********************************************
          ROBDD has been created
********************************************
Given expression is : or(equiv(x1,x2),x3)

Table T from the ROBDD
u  |  var | l | h
-------------------
0  :  4   -1  -1
1  :  4   -2  -2
2  :  3   0  1
3  :  2   1  2
4  :  2   2  1
5  :  1   3  4
Execution Time: 0.016452436 sec
```

Execution Time for this program is: 16.45 ms

2. Test Case to check all the boolean operators with three parameters:
   Parameters: x1, x2, x3

```
/Library/Java/JavaVirtualMachines/jdk-13.0.2.jdk/Contents/Home/bin/java "-javaagent:/Applications/
********************************************
          ROBDD has been created
********************************************
Given expression is : and(imp(not(x1) , equiv(1,x2)) , not(x3))

Table T from the ROBDD
u  |  var | l | h
-------------------
0  :  4   -1  -1
1  :  4   -2  -2
2  :  3   1  0
3  :  2   0  2
4  :  1   3  2
Execution Time: 0.018282928 sec
```

Execution Time for this program is: 16.28 ms

3. Test Case to check a longer boolean expression with 7 variables:

```
*******************************************
          ROBDD has been created
*******************************************
Given expression is : and(or(x1,x2), and(and(x3,x4), equiv(x5,imp(x6,x7))))

Table T from the ROBDD
u  |  var | l | h
------------------
0  :  8   -1  -1
1  :  8   -2  -2
2  :  7   1  0
3  :  6   0  2
4  :  7   0  1
5  :  6   1  4
6  :  5   3  5
7  :  4   0  6
8  :  3   0  7
9  :  2   0  8
10  :  1   9  8
Execution Time: 0.035267858 sec
```

The execution time for this program is 35.26 ms.

SatCount and AnySat Functions:

B. SatCount and AnySat:

1. SatCount function finds the total number of satisfying conditions for given expression and a node. This function implemented using recursion and traverse through the node to reach the terminal value 1.
2. AnySat function returns one of the satisfying conditions for the given expression.

1. **Test Case 1:**
   Expression: **or(equiv(x1,x2),x3)**
   Simple boolean expression with three variables

```
*******************************************
     Testing for SatCount and AnySat
*******************************************
Count of Satisfying Conditions is (SatCount):6
Any of the Satisfying Conditions (AnySat) :
3 -> 0    5 -> 0
Process finished with exit code 0
```

2. Test Case 2:

Boolean expression with seven parameters

Execution time for the given test case is around 18 ms.

```
******************************************
    Testing for SatCount and AnySat
******************************************
Given expression is : and(or(x1,x2), and(and(x3,x4), equiv(x5,imp(x6,x7))))
Count of Satisfying Conditions is (SatCount):32
Any of the Satisfying Conditions (AnySat) :
2 -> 0   3 -> 1   6 -> 0   7 -> 1   Execution time: 0.184333 ms
```

Restrict Function:

C. Restrict Function:

1. Restrict function reduces the ROBDD by keeping the value of one of the variables to either 0 or 1.
2. This function takes two arguments j and b.
3. Algorithm restricts [T,H] (u,j,b) which computes ROBDD for t^u[j/b]

1. **Test case 1:**

   Expression: **or(equiv(x1,x2),x3)**

   The expression has 3 variables and execution time is 0.018 ms.

```
*******************************************
          Testing for Restrict Function
*******************************************
Given expression is : or(equiv(x1,x2),x3)
Restricting for the node x[1] =0
Execution time: 0.017982637000000003 ms
Table T from the ROBDD after restricting one of the nodes
u  |  var | l | h
-------------------
  0 : 4 -1 -1
  1 : 4 -2 -2
  2 : 3 0 1
  3 : 2 1 2
```

2. **Test case 2:**

Testing with a long boolean expression and 7 variables.
Given Expression: **and(or(x1,x2), and(and(x3,x4), equiv(x5,imp(x6,x7))))**
Execution Time for the given expression is 0.4 ms.

```
********************************************
           ROBDD has been created
********************************************
Given expression is : and(or(x1,x2), and(and(x3,x4), equiv(x5,imp(x6,x7))))

Table T from the ROBDD
u  |  var | l | h
-------------------
0  :  8   -1  -1
1  :  8   -2  -2
2  :  7   1  0
3  :  6   0  2
4  :  7   0  1
5  :  6   1  4
6  :  5   3  5
7  :  4   0  6
8  :  3   0  7
9  :  2   0  8
10  :  1   9  8
Execution Time: 0.041864932 sec

********************************************
          Testing for Restrict Function
********************************************
Given expression is : and(or(x1,x2), and(and(x3,x4), equiv(x5,imp(x6,x7))))
Restricting for the node x[1] =0
Execution time: 0.041864932 ms
Table T from the ROBDD after restricting one of the nodes
u  |  var | l | h
-------------------
  0 : 8 -1 -1
  1 : 8 -2 -2
  2 : 2 0 8
********************************************
```

3. **Test case3 :**
Given Expression: **and(equiv(x1,x2), equiv(x3,x4))**
No. of Variables: 4
Execution Time: 0.029 ms.

```
*******************************************
          ROBDD has been created
*******************************************
Given expression is : and(equiv(x1,x2), equiv(x3,x4))

Table T from the ROBDD
u  |  var | l | h
-------------------
0  :  5   -1  -1
1  :  5   -2  -2
2  :  4   1  0
3  :  4   0  1
4  :  3   2  3
5  :  2   4  0
6  :  2   0  4
7  :  1   5  6
Execution Time: 0.029742755000000003 sec

*******************************************
         Testing for Restrict Function
*******************************************
Given expression is : and(equiv(x1,x2), equiv(x3,x4))
Restricting for the node x[1] =0
Execution time: 0.029742755000000003 ms
Table T from the ROBDD after restricting one of the nodes
u  |  var | l | h
-------------------
  0 : 5 -1 -1
  1 : 5 -2 -2
  2 : 3 2 3
  3 : 1 2 -1
```

Apply Function:

D. Apply Test Case:

Binary boolean Operators on ROBDDs implemented using Shannons expression. So according to this expansion theorom,

X ->(t1, t2)   op   x -> (t1_ , t2_)   =   x -> (t1 op t1_ , t2 op t2_)

So using this methodology, apply accepts two ROBDDs as inputs.
If at least one of them is not a terminal node, then we proceed according to the index, if both of them are terminal nodes, we solve the equation using the given operator.

In the given example the first expression is: **or(equiv(x1, x2), x3)**
The second Expression is : **Equiv(and(x1,x2) , x3)**
The operator used in AND
The T table after completing the AND operation is given below.

```
********************************************
          Testing for Apply Function
********************************************
Expression for first ROBDD: or(equiv(x1, x2), x3 )
Expression for second ROBDD: equiv( and(x1,x2) , x3 )
The operator used in AND
Output of Apply: 5
Execution Time is 1.70386 ms
Table T from the ROBDD
u  |  var | l | h
-------------------
0  :  4   -1  -1
1  :  4   -2  -2
2  :  3   0  1
3  :  2   1  2
4  :  2   2  1
5  :  1   3  4

Process finished with exit code 0
```

References:

1. https://unnikked.ga/how-to-build-a-boolean-expression-evaluator-518e9e068a65