CS 5001/6001: Intro to Quantum Computing **Spring 2021**
Group Members: Phill Janowski,...

1 The Problem and the Algorithm

We sought to find the minimum value of a list of numbers while using the fewest amount of queries checking that the value was the minimum value. Grover's Search Algorithm allows one query of a function, or oracle, to be used to check the returned value from the oracle by providing a quantum circuit that implements Grover's Search the superposition of the values of all possible values. We designed an oracle generator that creates a boolean function that should return true if the binary value of an index is passed in in binary form as a statevector and satisfies the conditions we choose. In our case our condition is if the index of some list L corresponding to the input value is greater than the current lowest value we have found in index y of L, or L[n] <L[y]. The boolean value will be returned from the function and used to determine whether or not the sign on the statevector is flipped. If the return value is true, the sign will be flipped, marking the value as a valid state, else it remains the same. The Grover algorithm then flips all of the states across the axis of the superposition, making the states we want to show as valid that are now negative, positive states with a much greater amplitude than the invalid states that are still sitting at the same uniform superposition amplitude.

To test the implementation, we generated a random list of numbers L and pick a starting index, y, at random. If the after an iteration of the circuit, the top measured value of the circuit represents an index of L that holds a smaller value than y, then we use the top measured value as our y on the next iteration of the circuit. In simpler terms, if the measured value of the circuit is y' and L[y'] < L[y] then y = y'. The Oracle is regenerated with the new value of y in mind and the circuit is recreated with the new oracle. This continues for a set amount of iterations, m, that should be optimized for our set of parameters to solve this problem in the least amount of iterations.

2 Circuit Design

The Grover's algorithm part of our circuit was mostly handle by qiskit, using their canned Grover operator to wrap an oracle that we provided. The operator begins by placing Hadamard gates on every qbit, creating a superposition of all states. The states then pass into one of two oracles that are implemented and in various states of working, however both use the same algorithm to generate the function for the oracle.

The most working implementation is using a qiskit Logical Expression Oracle that takes in a boolean expression such as (not a b not c d) or (not a b not c not d) to determine the validity of the state that is currently on the circuit in relation to the L[n] < L[y] condition we set. The boolean expression is generated before every run of the circuit because if the algorithm has not found the correct value and the circuit is working as expected, the value of y should have changed and the boolean expression will have fewer possible inputs that will satisfy it. The expression is eventually whittled down one expression that is only satisfied by the index value of L that contains the smallest value in the list.

The other implementation, using a qiskit Custom Circuit Oracle, works almost identically, except when it is used in conjunction with the Grover algorithm implementation in qiskit, it fails to flip the sign on any state. From the logical expression generator that is used with the logical expression oracle, we create a boolean function that is provided as a callback function to the oracle. The callback function takes in the bits that are provided by the circuit and are then placed in the boolean expression and evaluated. The result of the evaluation is returned along with the every input after the first xor'd with the return value to make the circuit reversible. If we call result of the evaluated expression 'result' and the input to the oracle is w, x, y, and z, then the output of the oracle would be [result, result $\oplus$ x, result $\oplus$ y, result $\oplus$ z]

The results of the are taken out superposition by applying Hadamard gates again to the qbits and the marked states are returned with higher amplitudes than the non marked states.

Fig. 1: A generated circuit using the Logical Expression Oracle. This is generated using the logical expression that is processed and used to construct the circuit and then wrapped by the standard Grover circuit.

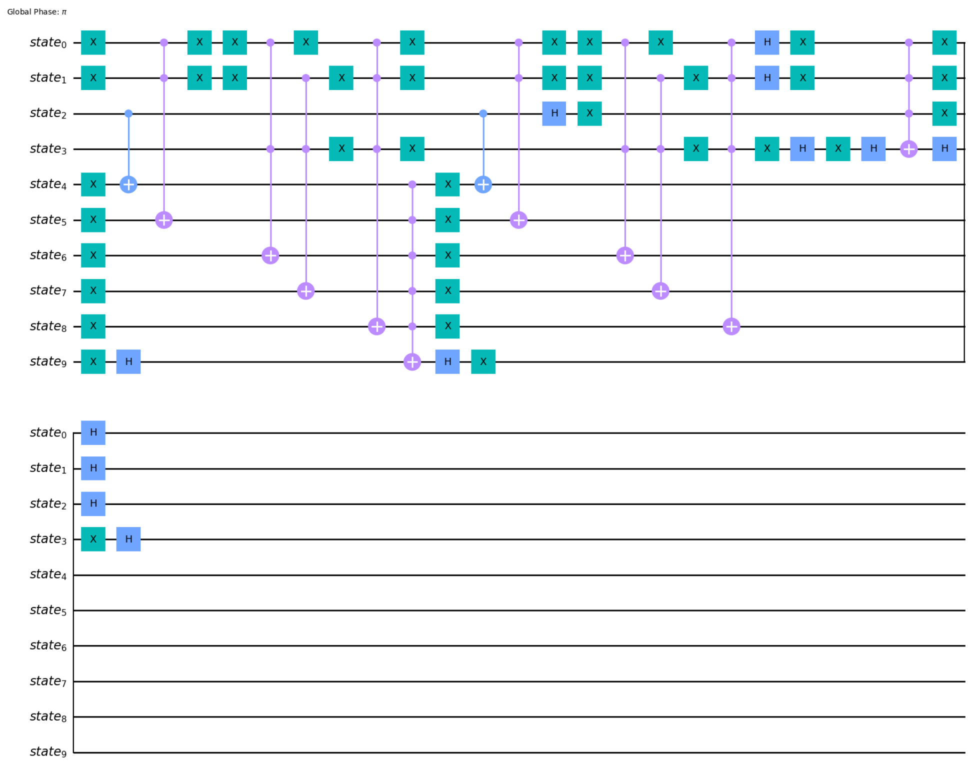
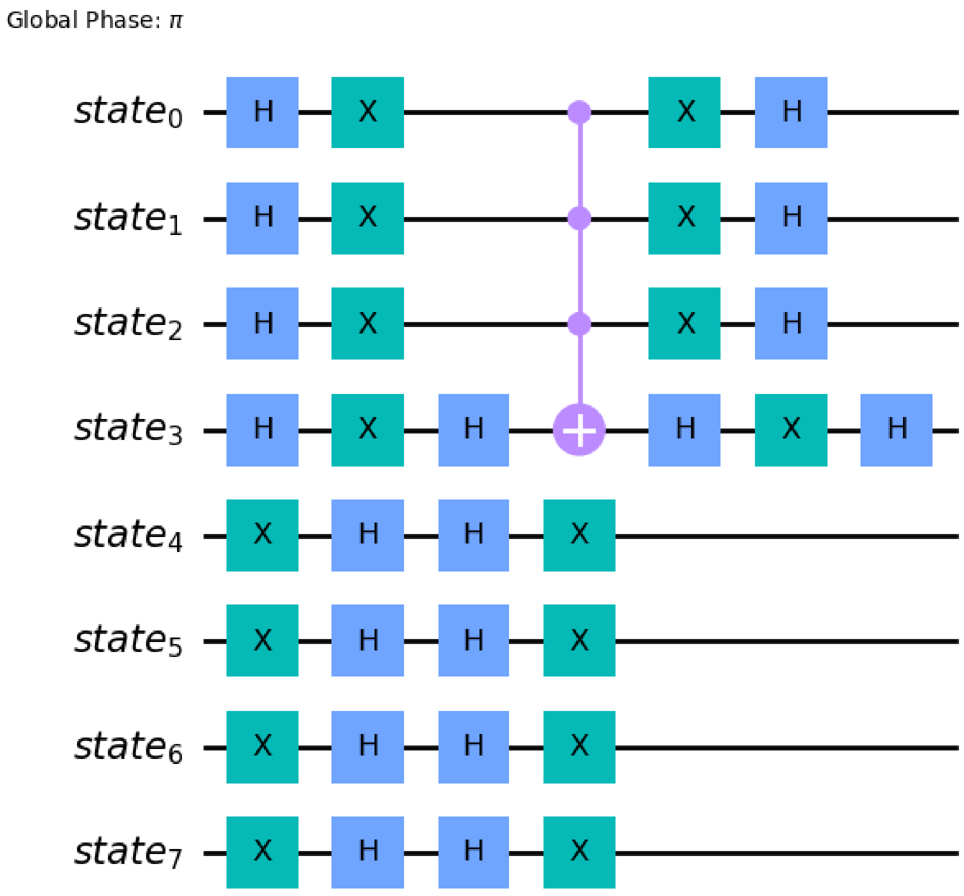


Fig. 2: A generated circuit using the Custom Circuit Oracle. This is generated and displays only as the standard Grover circuit without the oracle. The logic from the generated callback function is not shown here.



3 Experiments

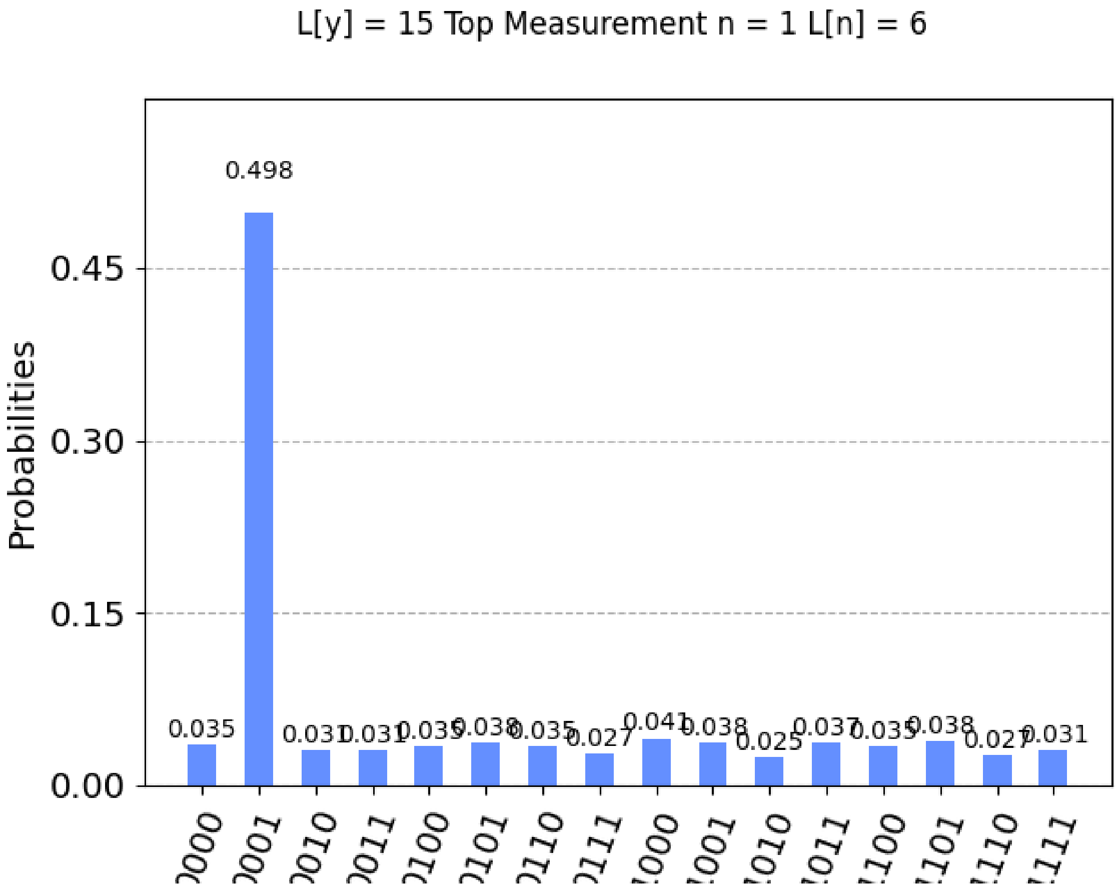
Experimenting with these circuits proved difficult given that one circuit, the Logical Expression Oracle circuit worked and marked the correct states some of the time and the Custom Circuit Oracle never marked anything.

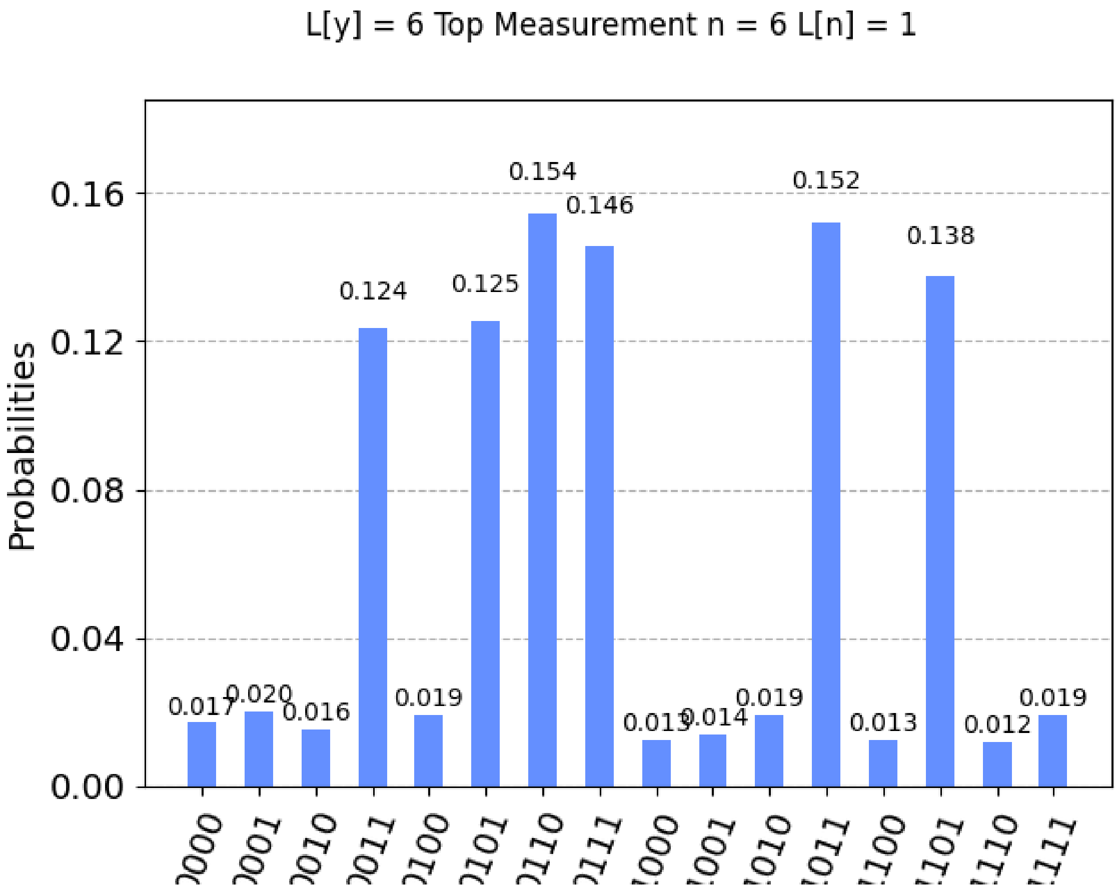
A pattern arose from the circuit running with the logical expression oracle. Proper states were getting marked sometimes and if they weren't, the reversed bit string of the proper states were. For example, if the oracle should have marked the value 1 (state 0001) as valid, the value 8 (state 1000) would be marked. Logs would show that there wasn't a valid boolean expression for 1000 to make it return true. Creating logs that output the values of the versed values of the marked states showed this to be a very common occurrence. Unit tests were written for the generation and evaluation functions that were being used that proved that the functions were behaving as they were expected to be.

A similar issue around what seemed to be the black box of the Grover circuit creator provided by qiskit was evident with runs using the Custom Circuit Oracle. As previously stated, circuits created with this oracle did not mark any states. However, the callback function provided to this oracle was being hit and was returning the correct values.

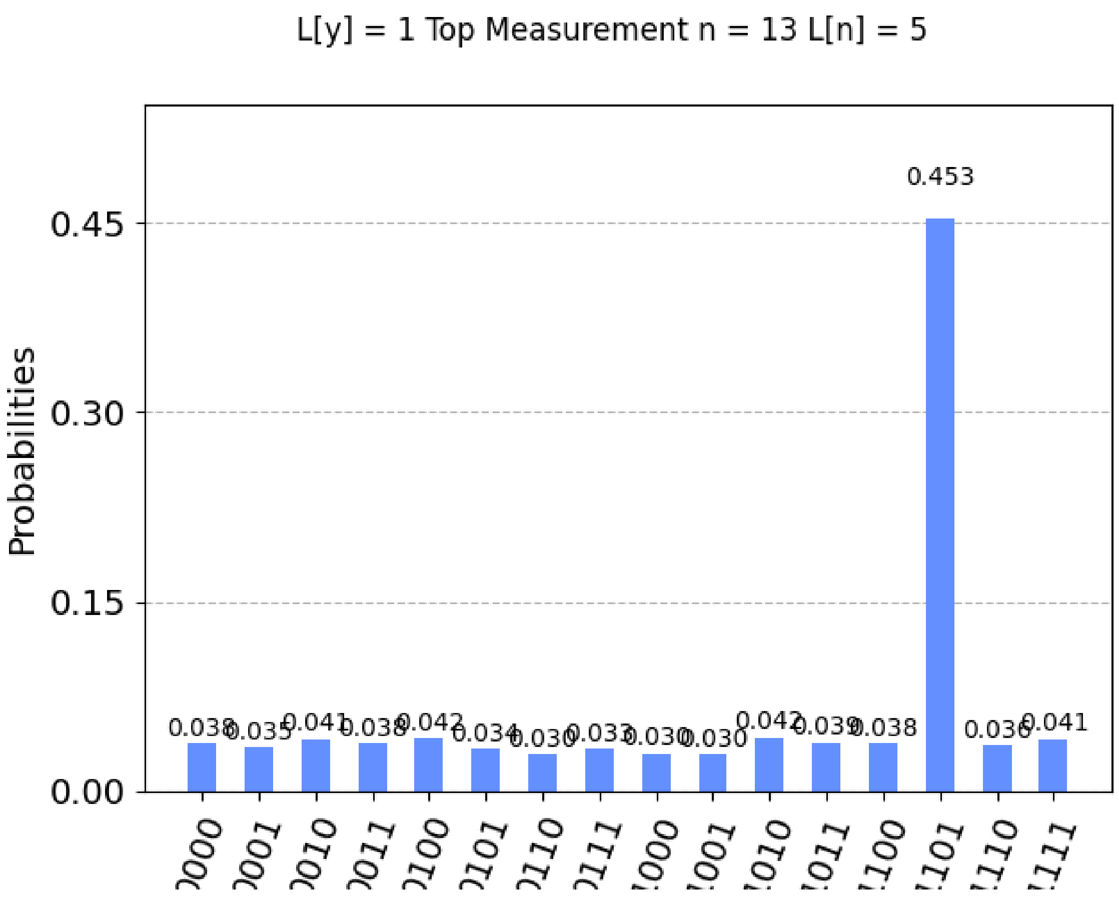
Different m values were used, m being the number of attempts given to find the index of the smallest value in the list L. This did not seem to affect the number of times that the logical expression gate was successful but it seem to help the custom circuit oracle come up with the correct value. This is definitely because the value picked for the custom circuit oracle circuit was much more random since no state was ever marked, so the more runs it had the more likely it was to end up with an index of L as the top measurement that was less than the current L[y]. m = 5 seemed to be as consistent for the logical expression oracle circuit as any other value.

Here are the graphs showing the marked states of the logical expression oracle circuit.

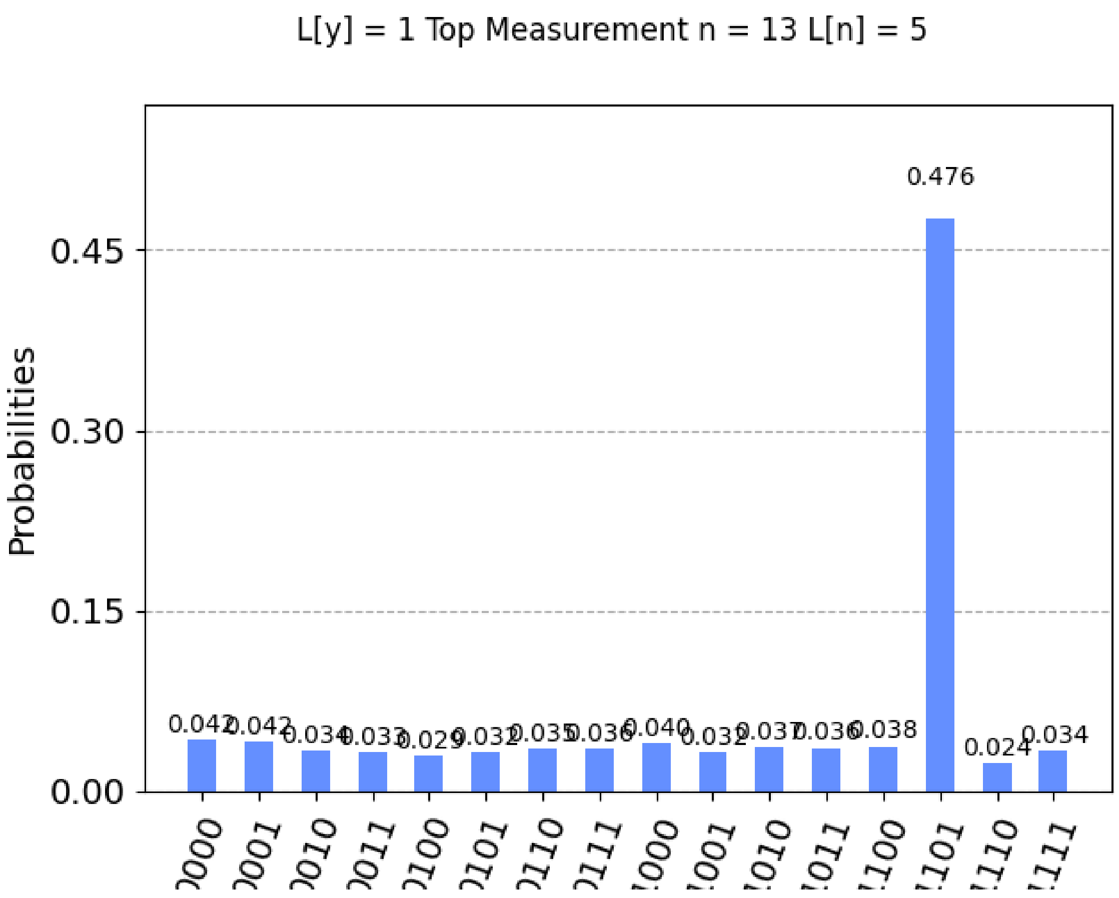


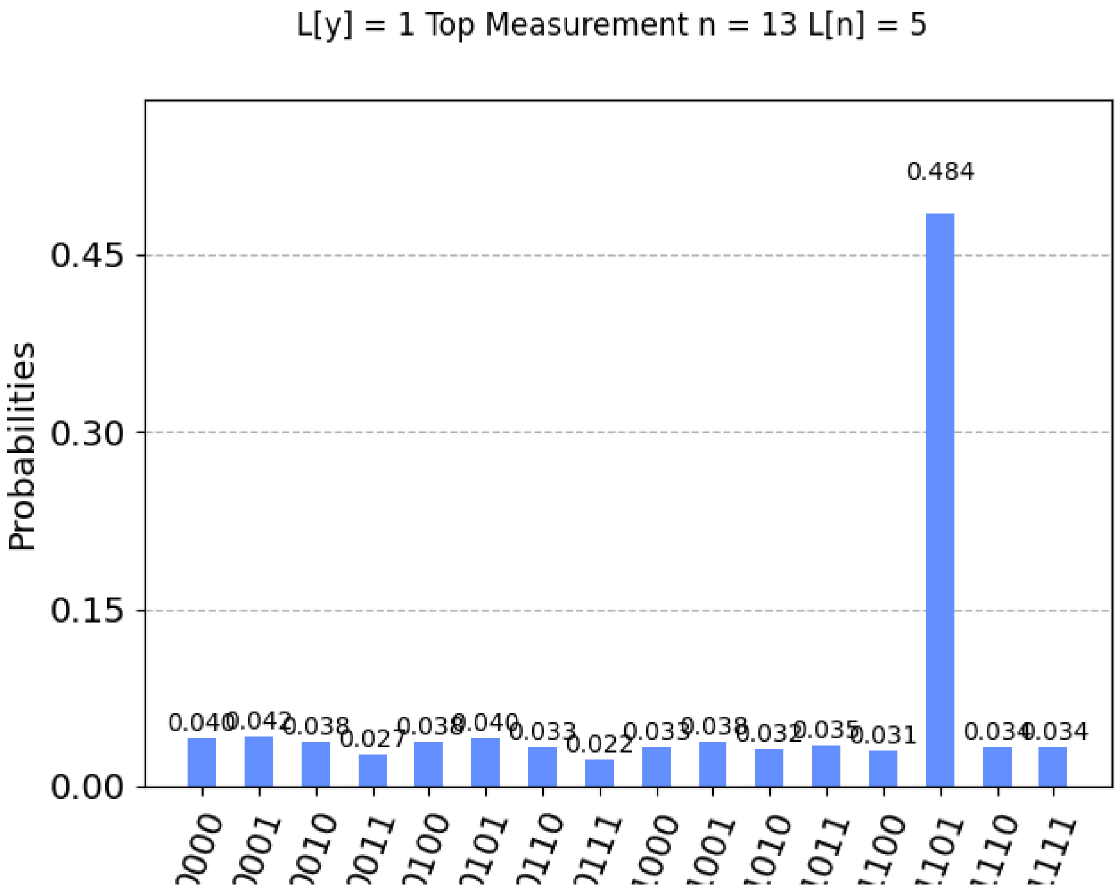


L[y] = 6 Top Measurement n = 6 L[n] = 1
Probabilities
0.00
0.04
0.08
0.12
0.16
0.017
0.020
0.016
0.124
0.019
0.125
0.154
0.146
0.013
0.014
0.019
0.152
0.013
0.138
0.012
0.019
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

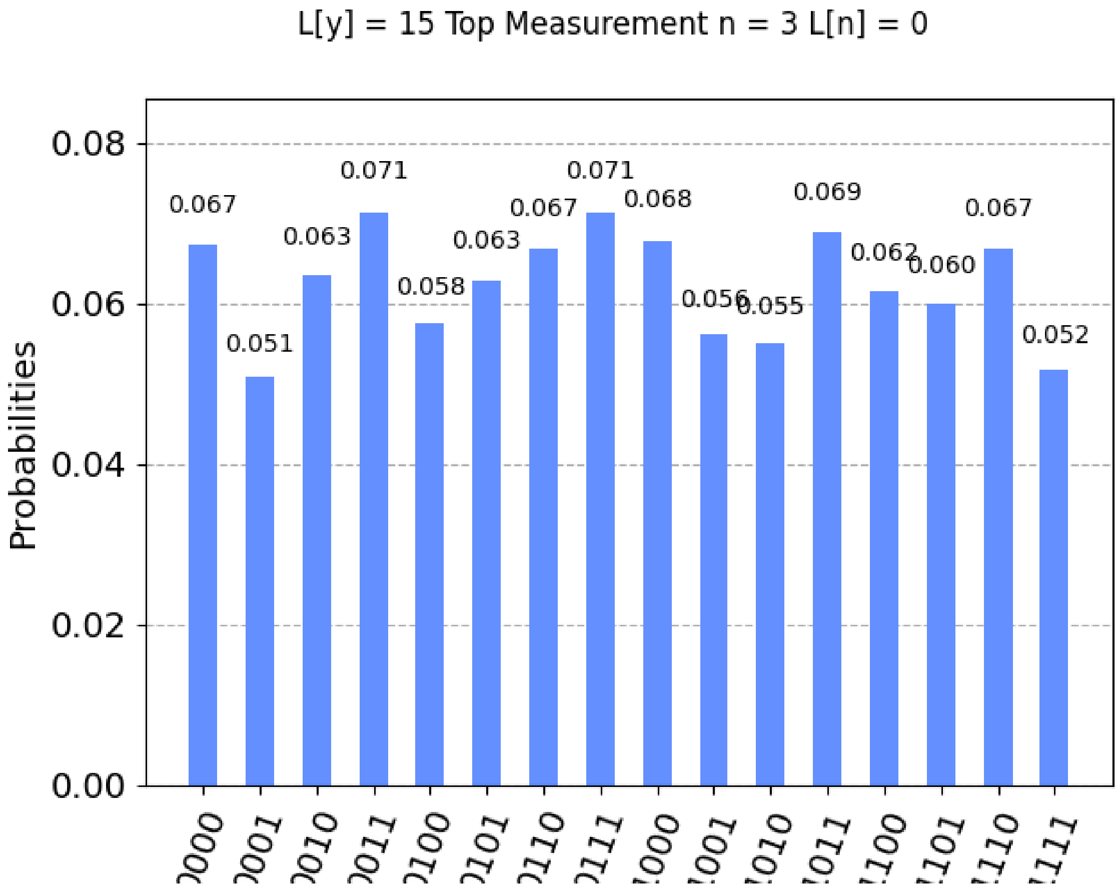


L[y] = 1 Top Measurement n = 13 L[n] = 5
Probabilities
0.00
0.15
0.30
0.45
0.038
0.035
0.041
0.038
0.042
0.034
0.030
0.033
0.030
0.030
0.042
0.039
0.038
0.453
0.036
0.041
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

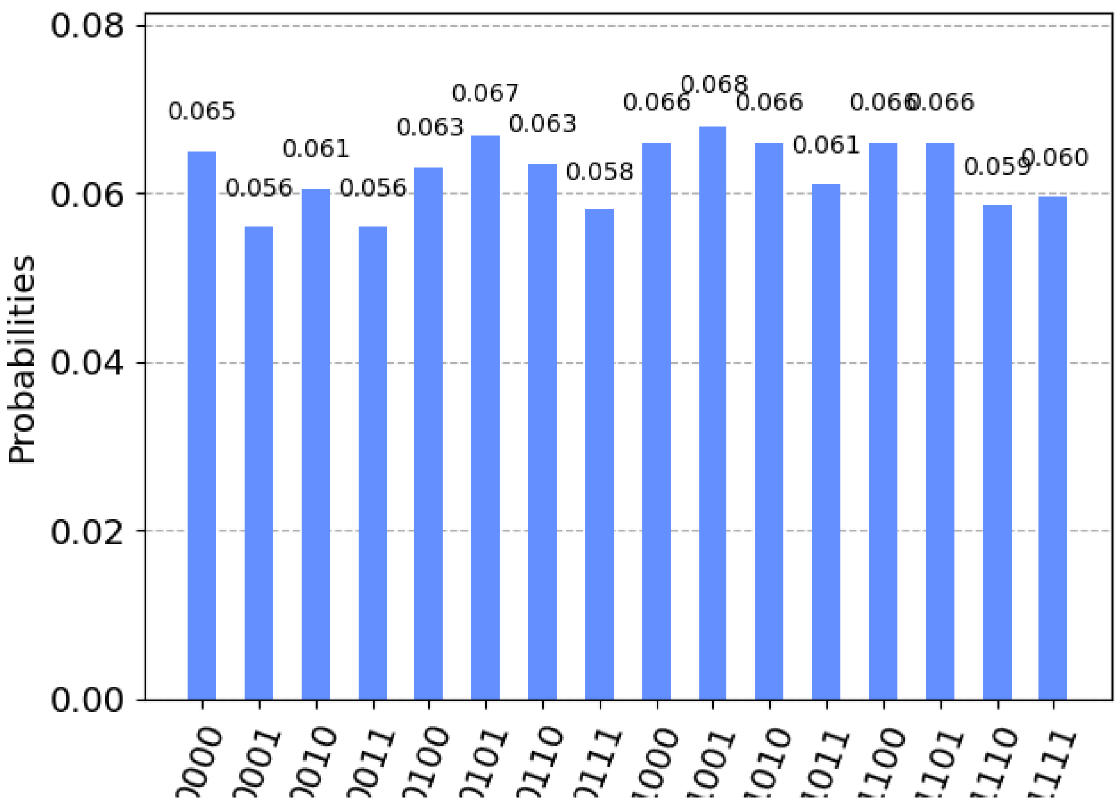

L[y] = 1 Top Measurement n = 13 L[n] = 5
Probabilities
0.00
0.15
0.30
0.45
0.042
0.042
0.034
0.033
0.029
0.032
0.035
0.036
0.040
0.032
0.037
0.036
0.038
0.476
0.024
0.034
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111


L[y] = 1 Top Measurement n = 13 L[n] = 5
Probabilities
0.45
0.30
0.15
0.00
0.484
0.040
0.042
0.038
0.027
0.038
0.040
0.033
0.022
0.033
0.038
0.032
0.035
0.031
0.034
0.034
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

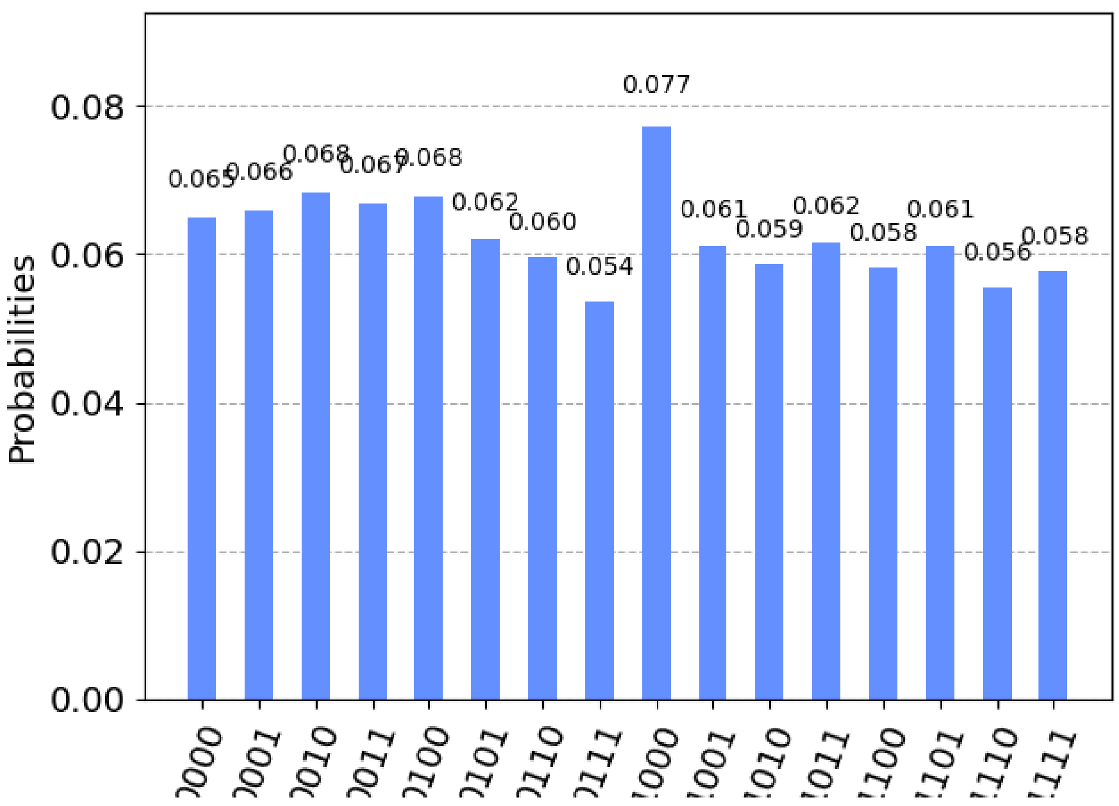
Here are the graphs showing that all states are unmarked when using the custom circuit oracle.

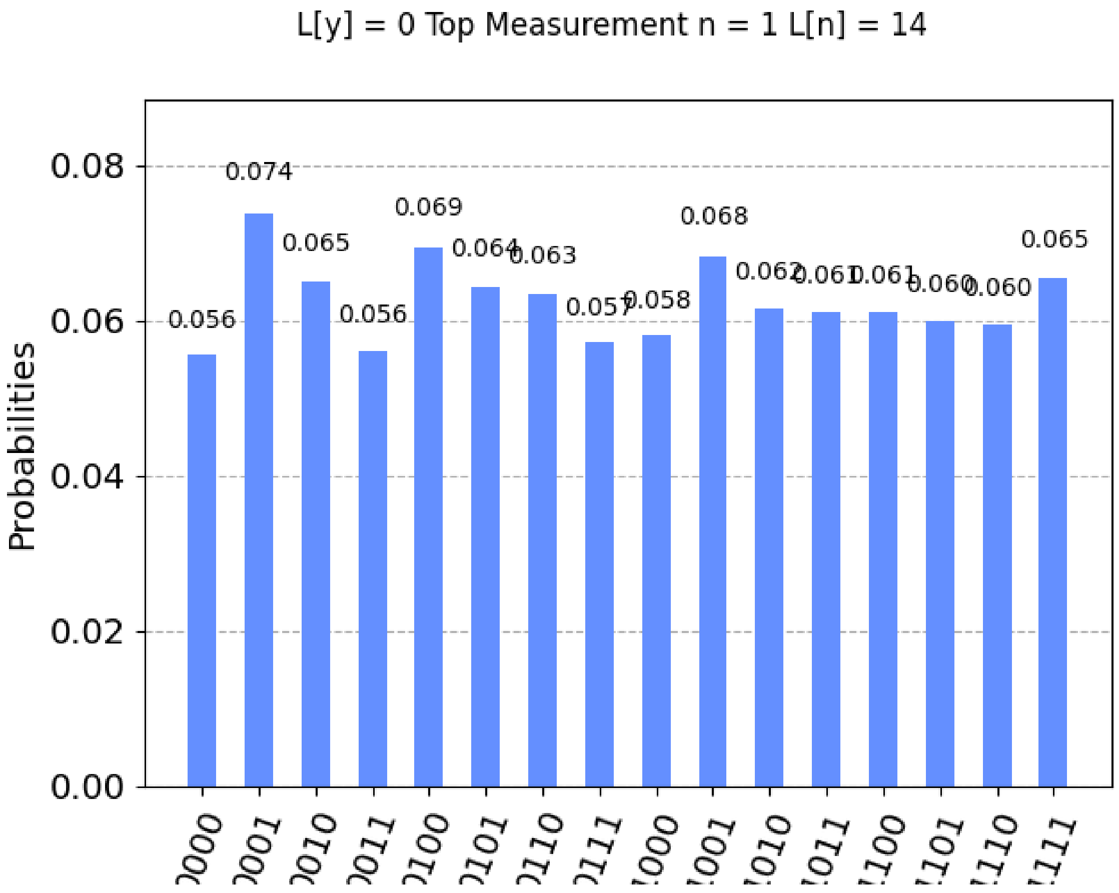


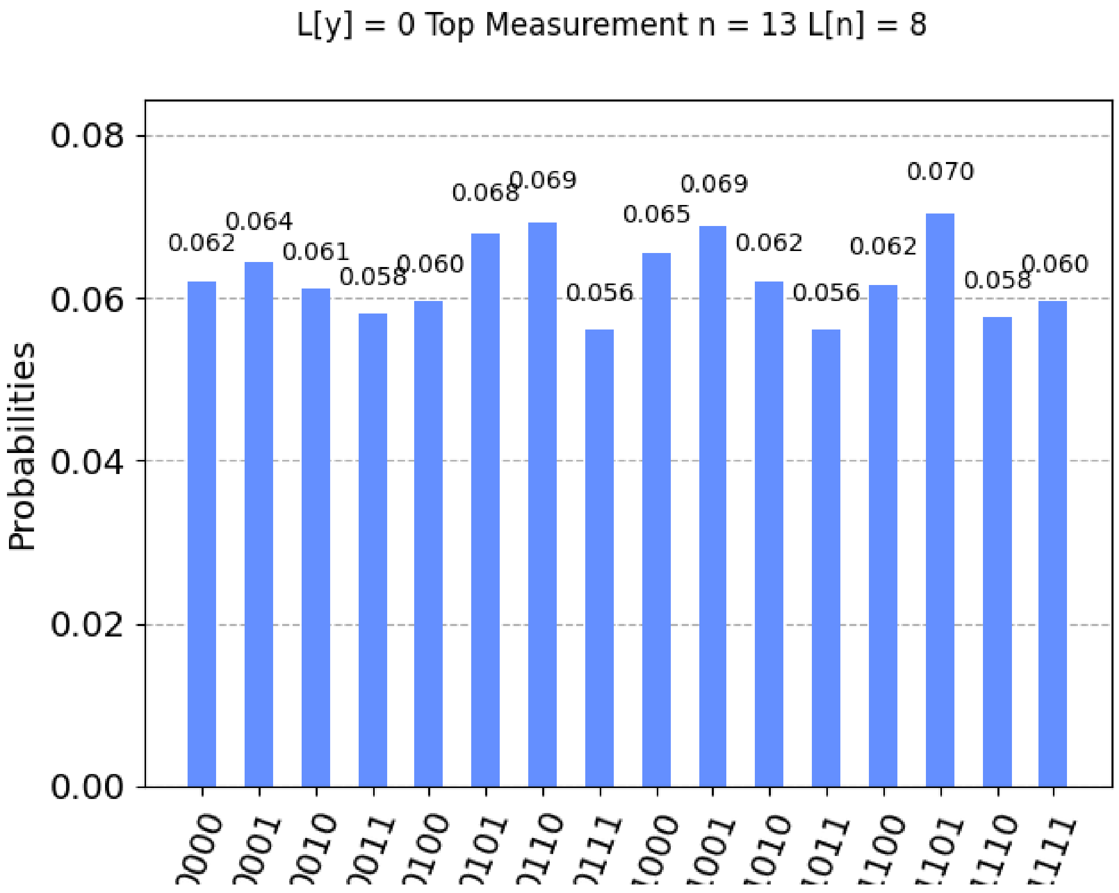
L[y] = 0 Top Measurement n = 9 L[n] = 11



L[y] = 0 Top Measurement n = 8 L[n] = 13




L[y] = 0 Top Measurement n = 1 L[n] = 14
Probabilities
0.00
0.02
0.04
0.06
0.08
0.056
0.074
0.065
0.056
0.069
0.064
0.063
0.057
0.058
0.068
0.062
0.061
0.061
0.060
0.060
0.065
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111


L[y] = 0 Top Measurement n = 13 L[n] = 8
Probabilities
0.00
0.02
0.04
0.06
0.08
0.062
0.064
0.061
0.058
0.060
0.068
0.069
0.056
0.065
0.069
0.062
0.056
0.062
0.070
0.058
0.060

Here is the log output from a run of the logical expression circuit showing that the proper boolean functions and being run and evaluating correctly yet the proper states are not being marked. The format for the 'all measurements' is as follows, '6': (144, '0110', 7), '6' being the integer value of the state, '144' being the number of hits, '0110' being the bit value, and '7' being the L[6] value.

```
L = [8, 7, 5, 4, 3, 2, 15, 1, 0, 13, 12, 14, 10, 9, 11, 6]
index containing 0 = 8
y = 6 | 0110
L[y] = 15
truth table = ['1', '1', '1', '1', '1', '1', '0', '1', '1', '1', '1', '1', '1', '1', '1', '1']

Boolean Functions for Oracle
 (  ~a &  ~b &  ~c &  ~d )  clause for marking y=0 L[y]=8
 (  ~a &  ~b &  ~c & d )  clause for marking y=1 L[y]=7
 (  ~a &  ~b & c &  ~d )  clause for marking y=2 L[y]=5
 (  ~a &  ~b & c & d )  clause for marking y=3 L[y]=4
 (  ~a & b &  ~c &  ~d )  clause for marking y=4 L[y]=3
 (  ~a & b &  ~c & d )  clause for marking y=5 L[y]=2
 (  ~a & b & c & d )  clause for marking y=7 L[y]=1
 ( a &  ~b &  ~c &  ~d )  clause for marking y=8 L[y]=0
 ( a &  ~b &  ~c & d )  clause for marking y=9 L[y]=13
 ( a &  ~b & c &  ~d )  clause for marking y=10 L[y]=12
 ( a &  ~b & c & d )  clause for marking y=11 L[y]=14
 ( a & b &  ~c &  ~d )  clause for marking y=12 L[y]=10
 ( a & b &  ~c & d )  clause for marking y=13 L[y]=9
 ( a & b & c &  ~d )  clause for marking y=14 L[y]=11
 ( a & b & c & d )  clause for marking y=15 L[y]=6

all measurements {'0': (67, '0000', 8),
'6': (964, '0110', 15),
'8': (72, '1000', 0),
'9': (65, '1001', 13),
'12': (74, '1100', 10),
'4': (98, '0100', 3),
'7': (68, '0111', 1),
'10': (68, '1010', 12),
'5': (71, '0101', 2),
'3': (59, '0011', 4),
'1': (72, '0001', 7),
'13': (65, '1101', 9),
'15': (73, '1111', 6),
'11': (76, '1011', 14),
'14': (82, '1110', 11),
'2': (74, '0010', 5)}
marked measurements {'0': (67, '0000', 8),
'8': (72, '1000', 0),
'9': (65, '1001', 13),
'12': (74, '1100', 10),
'4': (98, '0100', 3),
'7': (68, '0111', 1),
'10': (68, '1010', 12),
'5': (71, '0101', 2),
'3': (59, '0011', 4),
'1': (72, '0001', 7),
'13': (65, '1101', 9),
'15': (73, '1111', 6),
'11': (76, '1011', 14),
'14': (82, '1110', 11),
'2': (74, '0010', 5)}
marked reversed measurements {'0': (67, '0000', 8),
'1': (72, '0001', 7),
'9': (65, '1001', 13),
'3': (74, '0011', 4),
'2': (98, '0010', 5),
'14': (68, '1110', 11),
'5': (68, '0101', 2),
'10': (71, '1010', 12),
'12': (59, '1100', 10),
'8': (72, '1000', 0),
'11': (65, '1011', 14),
'15': (73, '1111', 6),
'13': (76, '1101', 9),
'7': (82, '0111', 1),
'4': (74, '0100', 3)}
top_measurement = 0110
rev_top_measurement = 0110
L[y_top] = 15, L[y_top_rev] = 15
old y value 6 L[y] = 15
y_primed value 6 L[y] = 15
new y value 6 L[y] = 15
finished run 1 / 5
y = 6 | 0110
L[y] = 15
```

```
truth table = ['1', '1', '1', '1', '1', '1', '0', '1', '1', '1', '1', '1', '1', '1', '1', '1']

Boolean Functions for Oracle
 (  ~a &  ~b &  ~c &  ~d )  clause for marking y=0 L[y]=8
 (  ~a &  ~b &  ~c & d )  clause for marking y=1 L[y]=7
 (  ~a &  ~b & c &  ~d )  clause for marking y=2 L[y]=5
 (  ~a &  ~b & c & d )  clause for marking y=3 L[y]=4
 (  ~a & b &  ~c &  ~d )  clause for marking y=4 L[y]=3
 (  ~a & b &  ~c & d )  clause for marking y=5 L[y]=2
 (  ~a & b & c & d )  clause for marking y=7 L[y]=1
 ( a &  ~b &  ~c &  ~d )  clause for marking y=8 L[y]=0
 ( a &  ~b &  ~c & d )  clause for marking y=9 L[y]=13
 ( a &  ~b & c &  ~d )  clause for marking y=10 L[y]=12
 ( a &  ~b & c & d )  clause for marking y=11 L[y]=14
 ( a & b &  ~c &  ~d )  clause for marking y=12 L[y]=10
 ( a & b &  ~c & d )  clause for marking y=13 L[y]=9
 ( a & b & c &  ~d )  clause for marking y=14 L[y]=11
 ( a & b & c & d )  clause for marking y=15 L[y]=6
all measurements {'6': (979, '0110', 15),
'2': (68, '0010', 5),
'4': (79, '0100', 3),
'5': (69, '0101', 2),
'1': (69, '0001', 7),
'12': (68, '1100', 10),
'0': (58, '0000', 8),
'15': (77, '1111', 6),
'8': (81, '1000', 0),
'3': (62, '0011', 4),
'10': (74, '1010', 12),
'14': (80, '1110', 11),
'9': (58, '1001', 13),
'13': (71, '1101', 9),
'11': (76, '1011', 14),
'7': (79, '0111', 1)}
marked measurements {'2': (68, '0010', 5),
'4': (79, '0100', 3),
'5': (69, '0101', 2),
'1': (69, '0001', 7),
'12': (68, '1100', 10),
'0': (58, '0000', 8),
'15': (77, '1111', 6),
'8': (81, '1000', 0),
'3': (62, '0011', 4),
'10': (74, '1010', 12),
'14': (80, '1110', 11),
'9': (58, '1001', 13),
'13': (71, '1101', 9),
'11': (76, '1011', 14),
'7': (79, '0111', 1)}
marked reversed measurements {'4': (68, '0100', 3),
'2': (79, '0010', 5),
'10': (69, '1010', 12),
'8': (69, '1000', 0),
'3': (68, '0011', 4),
'0': (58, '0000', 8),
'15': (77, '1111', 6),
'1': (81, '0001', 7),
'12': (62, '1100', 10),
'5': (74, '0101', 2),
'7': (80, '0111', 1),
'9': (58, '1001', 13),
'11': (71, '1011', 14),
'13': (76, '1101', 9),
'14': (79, '1110', 11)}
top_measurement = 0110
rev_top_measurement = 0110
L[y_top] = 15, L[y_top_rev] = 15
old y value 6 L[y] = 15
y_primed value 6 L[y] = 15
new y value 6 L[y] = 15
finished run 2 / 5
y = 6 | 0110
L[y] = 15
truth table = ['1', '1', '1', '1', '1', '1', '0', '1', '1', '1', '1', '1', '1', '1', '1', '1']

Boolean Functions for Oracle
 (  ~a &  ~b &  ~c &  ~d )  clause for marking y=0 L[y]=8
 (  ~a &  ~b &  ~c & d )  clause for marking y=1 L[y]=7
 (  ~a &  ~b & c &  ~d )  clause for marking y=2 L[y]=5
 (  ~a &  ~b & c & d )  clause for marking y=3 L[y]=4
 (  ~a & b &  ~c &  ~d )  clause for marking y=4 L[y]=3
 (  ~a & b &  ~c & d )  clause for marking y=5 L[y]=2
 (  ~a & b & c & d )  clause for marking y=7 L[y]=1
 ( a &  ~b &  ~c &  ~d )  clause for marking y=8 L[y]=0
 ( a &  ~b &  ~c & d )  clause for marking y=9 L[y]=13
 ( a &  ~b & c &  ~d )  clause for marking y=10 L[y]=12
 ( a &  ~b & c & d )  clause for marking y=11 L[y]=14
 ( a & b &  ~c &  ~d )  clause for marking y=12 L[y]=10
 ( a & b &  ~c & d )  clause for marking y=13 L[y]=9
 ( a & b & c &  ~d )  clause for marking y=14 L[y]=11
 ( a & b & c & d )  clause for marking y=15 L[y]=6
all measurements {'6': (993, '0110', 15),
'2': (77, '0010', 5),
'15': (77, '1111', 6),
'13': (60, '1101', 9),
'4': (66, '0100', 3),
'11': (68, '1011', 14),
'5': (76, '0101', 2),
'14': (82, '1110', 11),
'0': (71, '0000', 8),
'8': (67, '1000', 0),
'9': (57, '1001', 13),
'7': (79, '0111', 1),
'12': (69, '1100', 10),
'10': (60, '1010', 12),
'1': (74, '0001', 7),
'3': (72, '0011', 4)}
marked measurements {'2': (77, '0010', 5),
'15': (77, '1111', 6),
'13': (60, '1101', 9),
'4': (66, '0100', 3),
'11': (68, '1011', 14),
'5': (76, '0101', 2),
'14': (82, '1110', 11),
'0': (71, '0000', 8),
'8': (67, '1000', 0),
'9': (57, '1001', 13),
'7': (79, '0111', 1),
'12': (69, '1100', 10),
'10': (60, '1010', 12),
'1': (74, '0001', 7),
'3': (72, '0011', 4)}
marked reversed measurements {'4': (77, '0100', 3),
'15': (77, '1111', 6),
'11': (60, '1011', 14),
'2': (66, '0010', 5),
'13': (68, '1101', 9),
'10': (76, '1010', 12),
'7': (82, '0111', 1),
'0': (71, '0000', 8),
'1': (67, '0001', 7),
'9': (57, '1001', 13),
'14': (79, '1110', 11),
'3': (69, '0011', 4),
'5': (60, '0101', 2),
'8': (74, '1000', 0),
'12': (72, '1100', 10)}
top_measurement = 0110
rev_top_measurement = 0110
L[y_top] = 15, L[y_top_rev] = 15
old y value 6 L[y] = 15
y_primed value 6 L[y] = 15
new y value 6 L[y] = 15
finished run 3 / 5
y = 6 | 0110
L[y] = 15
truth table = ['1', '1', '1', '1', '1', '1', '0', '1', '1', '1', '1', '1', '1', '1', '1', '1']

Boolean Functions for Oracle
 (  ~a &  ~b &  ~c &  ~d )  clause for marking y=0 L[y]=8
 (  ~a &  ~b &  ~c & d )  clause for marking y=1 L[y]=7
 (  ~a &  ~b & c &  ~d )  clause for marking y=2 L[y]=5
 (  ~a &  ~b & c & d )  clause for marking y=3 L[y]=4
 (  ~a & b &  ~c &  ~d )  clause for marking y=4 L[y]=3
 (  ~a & b &  ~c & d )  clause for marking y=5 L[y]=2
 (  ~a & b & c & d )  clause for marking y=7 L[y]=1
 ( a &  ~b &  ~c &  ~d )  clause for marking y=8 L[y]=0
 ( a &  ~b &  ~c & d )  clause for marking y=9 L[y]=13
 ( a &  ~b & c &  ~d )  clause for marking y=10 L[y]=12
 ( a &  ~b & c & d )  clause for marking y=11 L[y]=14
 ( a & b &  ~c &  ~d )  clause for marking y=12 L[y]=10
 ( a & b &  ~c & d )  clause for marking y=13 L[y]=9
 ( a & b & c &  ~d )  clause for marking y=14 L[y]=11
 ( a & b & c & d )  clause for marking y=15 L[y]=6
```

```
all measurements {'0': (73, '0000', 8),
'8': (54, '1000', 0),
'6': (970, '0110', 15),
'12': (70, '1100', 10),
'10': (67, '1010', 12),
'3': (67, '0011', 4),
'9': (64, '1001', 13),
'5': (62, '0101', 2),
'11': (81, '1011', 14),
'4': (94, '0100', 3),
'7': (68, '0111', 1),
'14': (74, '1110', 11),
'1': (73, '0001', 7),
'2': (84, '0010', 5),
'13': (77, '1101', 9),
'15': (70, '1111', 6)}
marked measurements {'0': (73, '0000', 8),
'8': (54, '1000', 0),
'12': (70, '1100', 10),
'10': (67, '1010', 12),
'3': (67, '0011', 4),
'9': (64, '1001', 13),
'5': (62, '0101', 2),
'11': (81, '1011', 14),
'4': (94, '0100', 3),
'7': (68, '0111', 1),
'14': (74, '1110', 11),
'1': (73, '0001', 7),
'2': (84, '0010', 5),
'13': (77, '1101', 9),
'15': (70, '1111', 6)}
marked reversed measurements {'0': (73, '0000', 8),
'1': (54, '0001', 7),
'3': (70, '0011', 4),
'5': (67, '0101', 2),
'12': (67, '1100', 10),
'9': (64, '1001', 13),
'10': (62, '1010', 12),
'13': (81, '1101', 9),
'2': (94, '0010', 5),
'14': (68, '1110', 11),
'7': (74, '0111', 1),
'8': (73, '1000', 0),
'4': (84, '0100', 3),
'11': (77, '1011', 14),
'15': (70, '1111', 6)}
top_measurement = 0110
rev_top_measurement = 0110
L[y_top] = 15, L[y_top_rev] = 15
old y value 6 L[y] = 15
y_primed value 6 L[y] = 15
new y value 6 L[y] = 15
finished run 4 / 5
y = 6 | 0110
L[y] = 15
truth table = ['1', '1', '1', '1', '1', '1', '0', '1', '1', '1', '1', '1', '1', '1', '1', '1']

Boolean Functions for Oracle
 (  ~a &  ~b &  ~c &  ~d )  clause for marking y=0 L[y]=8
 (  ~a &  ~b &  ~c & d )  clause for marking y=1 L[y]=7
 (  ~a &  ~b & c &  ~d )  clause for marking y=2 L[y]=5
 (  ~a &  ~b & c & d )  clause for marking y=3 L[y]=4
 (  ~a & b &  ~c &  ~d )  clause for marking y=4 L[y]=3
 (  ~a & b &  ~c & d )  clause for marking y=5 L[y]=2
 (  ~a & b & c & d )  clause for marking y=7 L[y]=1
 ( a &  ~b &  ~c &  ~d )  clause for marking y=8 L[y]=0
 ( a &  ~b &  ~c & d )  clause for marking y=9 L[y]=13
 ( a &  ~b & c &  ~d )  clause for marking y=10 L[y]=12
 ( a &  ~b & c & d )  clause for marking y=11 L[y]=14
 ( a & b &  ~c &  ~d )  clause for marking y=12 L[y]=10
 ( a & b &  ~c & d )  clause for marking y=13 L[y]=9
 ( a & b & c &  ~d )  clause for marking y=14 L[y]=11
 ( a & b & c & d )  clause for marking y=15 L[y]=6
all measurements {'14': (80, '1110', 11),
'7': (77, '0111', 1),
'8': (77, '1000', 0),
'6': (926, '0110', 15),
'2': (73, '0010', 5),
'11': (63, '1011', 14),
'0': (83, '0000', 8),
'4': (91, '0100', 3),
'3': (81, '0011', 4),
'5': (79, '0101', 2),
'1': (85, '0001', 7),
'10': (57, '1010', 12),
'13': (59, '1101', 9),
'15': (77, '1111', 6),
'9': (68, '1001', 13),
'12': (72, '1100', 10)}
marked measurements {'14': (80, '1110', 11),
'7': (77, '0111', 1),
'8': (77, '1000', 0),
'2': (73, '0010', 5),
'11': (63, '1011', 14),
'0': (83, '0000', 8),
'4': (91, '0100', 3),
'3': (81, '0011', 4),
'5': (79, '0101', 2),
'1': (85, '0001', 7),
'10': (57, '1010', 12),
'13': (59, '1101', 9),
'15': (77, '1111', 6),
'9': (68, '1001', 13),
'12': (72, '1100', 10)}
marked reversed measurements {'7': (80, '0111', 1),
'14': (77, '1110', 11),
'1': (77, '0001', 7),
'4': (73, '0100', 3),
'13': (63, '1101', 9),
'0': (83, '0000', 8),
'2': (91, '0010', 5),
'12': (81, '1100', 10),
'10': (79, '1010', 12),
'8': (85, '1000', 0),
'5': (57, '0101', 2),
'11': (59, '1011', 14),
'15': (77, '1111', 6),
'9': (68, '1001', 13),
'3': (72, '0011', 4)}
top_measurement = 0110
rev_top_measurement = 0110
L[y_top] = 15, L[y_top_rev] = 15
old y value 6 L[y] = 15
```

```
y_primed value 6 L[y] = 15
new y value 6 L[y] = 15
finished run 5 / 5
final result of grover search, smallest y = 6 L[y] = 15
final result of y should be 8
```

Below is the log out from a run of the custom circuit oracle circuit. It does find the smallest value's index in L but its out of luck with the top measured states that are coming out of the grover circuit, not being affected by the oracle.

```
L = [5, 11, 8, 15, 14, 0, 7, 6, 2, 10, 9, 1, 13, 12, 4, 3]
index containing 0 = 5
y = 3 | 0011
L[y] = 15
truth table = ['1', '1', '1', '0', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1']

Boolean Functions for Oracle
 (  not  {0}  and  not  {1}  and  not  {2}  and  not  {3}  )  clause for marking y=0 L[y]=5
 (  not  {0}  and  not  {1}  and  not  {2}  and  {3}  )  clause for marking y=1 L[y]=11
 (  not  {0}  and  not  {1}  and  {2}  and  not  {3}  )  clause for marking y=2 L[y]=8
 (  not  {0}  and  {1}  and  not  {2}  and  not  {3}  )  clause for marking y=4 L[y]=14
 (  not  {0}  and  {1}  and  not  {2}  and  {3}  )  clause for marking y=5 L[y]=0
 (  not  {0}  and  {1}  and  {2}  and  not  {3}  )  clause for marking y=6 L[y]=7
 (  not  {0}  and  {1}  and  {2}  and  {3}  )  clause for marking y=7 L[y]=6
 (  {0}  and  not  {1}  and  not  {2}  and  not  {3}  )  clause for marking y=8 L[y]=2
 (  {0}  and  not  {1}  and  not  {2}  and  {3}  )  clause for marking y=9 L[y]=10
 (  {0}  and  not  {1}  and  {2}  and  not  {3}  )  clause for marking y=10 L[y]=9
 (  {0}  and  not  {1}  and  {2}  and  {3}  )  clause for marking y=11 L[y]=1
 (  {0}  and  {1}  and  not  {2}  and  not  {3}  )  clause for marking y=12 L[y]=13
 (  {0}  and  {1}  and  not  {2}  and  {3}  )  clause for marking y=13 L[y]=12
 (  {0}  and  {1}  and  {2}  and  not  {3}  )  clause for marking y=14 L[y]=4
 (  {0}  and  {1}  and  {2}  and  {3}  )  clause for marking y=15 L[y]=3
inputs bits for f_L = 1000
 ( not True and not False and not False and not False ) or
 ( not True and not False and not False and False ) or
 ( not True and not False and False and not False ) or
 ( not True and False and not False and not False ) or
 ( not True and False and not False and False ) or
 ( not True and False and False and not False ) or
 ( not True and False and False and False ) or
 ( True and not False and not False and not False ) or
 ( True and not False and not False and False ) or
 ( True and not False and False and not False ) or
 ( True and not False and False and False ) or
 ( True and False and not False and not False ) or
 ( True and False and not False and False ) or
 ( True and False and False and not False ) or
 ( True and False and False and False )  evaluated to True
```

```
truth table = ['1', '1', '1', '0', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1']

Boolean Functions for Oracle
 (  not  {0}  and  not  {1}  and  not  {2}  and  not  {3}  )  clause for marking y=0 L[y]=5
 (  not  {0}  and  not  {1}  and  not  {2}  and  {3}  )  clause for marking y=1 L[y]=11
 (  not  {0}  and  not  {1}  and  {2}  and  not  {3}  )  clause for marking y=2 L[y]=8
 (  not  {0}  and  {1}  and  not  {2}  and  not  {3}  )  clause for marking y=4 L[y]=14
 (  not  {0}  and  {1}  and  not  {2}  and  {3}  )  clause for marking y=5 L[y]=0
 (  not  {0}  and  {1}  and  {2}  and  not  {3}  )  clause for marking y=6 L[y]=7
 (  not  {0}  and  {1}  and  {2}  and  {3}  )  clause for marking y=7 L[y]=6
 (  {0}  and  not  {1}  and  not  {2}  and  not  {3}  )  clause for marking y=8 L[y]=2
 (  {0}  and  not  {1}  and  not  {2}  and  {3}  )  clause for marking y=9 L[y]=10
 (  {0}  and  not  {1}  and  {2}  and  not  {3}  )  clause for marking y=10 L[y]=9
 (  {0}  and  not  {1}  and  {2}  and  {3}  )  clause for marking y=11 L[y]=1
 (  {0}  and  {1}  and  not  {2}  and  not  {3}  )  clause for marking y=12 L[y]=13
 (  {0}  and  {1}  and  not  {2}  and  {3}  )  clause for marking y=13 L[y]=12
 (  {0}  and  {1}  and  {2}  and  not  {3}  )  clause for marking y=14 L[y]=4
 (  {0}  and  {1}  and  {2}  and  {3}  )  clause for marking y=15 L[y]=3
inputs bits for f_L = 1000
( not True and not False and not False and not False ) or
( not True and not False and not False and False ) or
( not True and not False and False and not False ) or
( not True and False and not False and not False ) or
( not True and False and not False and False ) or
( not True and False and False and not False ) or
( not True and False and False and False ) or
( True and not False and not False and not False ) or
( True and not False and not False and False ) or
( True and not False and False and not False ) or
( True and not False and False and False ) or
( True and False and not False and not False ) or
( True and False and not False and False ) or
( True and False and False and not False ) or
( True and False and False and False )  evaluated to True
all measurements {'13': (129, '1101', 12),
'6': (144, '0110', 7),
'0': (119, '0000', 5),
'9': (128, '1001', 10),
'12': (115, '1100', 13),
'15': (115, '1111', 3),
'5': (140, '0101', 0),
'4': (103, '0100', 14),
'1': (135, '0001', 11),
'14': (140, '1110', 4),
'11': (123, '1011', 1),
'3': (131, '0011', 15),
'7': (133, '0111', 6),
'8': (146, '1000', 2),
'10': (126, '1010', 9),
'2': (121, '0010', 8)}
marked measurements {'13': (129, '1101', 12),
'6': (144, '0110', 7),
'0': (119, '0000', 5),
'9': (128, '1001', 10),
'12': (115, '1100', 13),
'15': (115, '1111', 3),
'5': (140, '0101', 0),
'4': (103, '0100', 14),
'1': (135, '0001', 11),
'14': (140, '1110', 4),
'11': (123, '1011', 1),
'7': (133, '0111', 6),
'8': (146, '1000', 2),
'10': (126, '1010', 9),
'2': (121, '0010', 8)}
marked reversed measurements {'11': (129, '1011', 1),
'6': (144, '0110', 7),
'0': (119, '0000', 5),
'9': (128, '1001', 10),
'15': (115, '1111', 3),
'10': (140, '1010', 9),
'2': (103, '0010', 8),
'8': (135, '1000', 2),
'7': (140, '0111', 6),
'13': (123, '1101', 12),
'12': (131, '1100', 13),
'14': (133, '1110', 4),
'1': (146, '0001', 11),
'5': (126, '0101', 0),
'4': (121, '0100', 14)}
top_measurement = 1000
rev_top_measurement = 0001
```

```
L[y_top] = 2, L[y_top_rev] = 11
old y value 3 L[y] = 15
y_primed value 8 L[y] = 2
new y value 8 L[y] = 2
finished run 1 / 5
y = 8 | 1000
L[y] = 2
truth table = ['0', '0', '0', '0', '0', '1', '0', '0', '0', '0', '0', '1', '0', '0', '0', '0']

Boolean Functions for Oracle
 (  not  {0}  and  {1}  and  not  {2}  and  {3}  )  clause for marking y=5 L[y]=0
 (  {0}  and  not  {1}  and  {2}  and  {3}  )  clause for marking y=11 L[y]=1
truth table = ['0', '0', '0', '0', '0', '1', '0', '0', '0', '0', '0', '1', '0', '0', '0', '0']

Boolean Functions for Oracle
 (  not  {0}  and  {1}  and  not  {2}  and  {3}  )  clause for marking y=5 L[y]=0
 (  {0}  and  not  {1}  and  {2}  and  {3}  )  clause for marking y=11 L[y]=1
all measurements {
'0': (120, '0000', 5), '13': (123, '1101', 12),
'9': (147, '1001', 10), '4': (119, '0100', 14),
'3': (131, '0011', 15), '5': (139, '0101', 0),
'2': (122, '0010', 8), '6': (135, '0110', 7),
'1': (137, '0001', 11), '8': (107, '1000', 2),
'12': (140, '1100', 13), '14': (132, '1110', 4),
'7': (131, '0111', 6), '10': (128, '1010', 9),
'11': (115, '1011', 1), '15': (122, '1111', 3)
}
marked measurements {'5': (139, '0101', 0), '11': (115, '1011', 1)}
marked reversed measurements {'11': (123, '1011', 1), '5': (128, '0101', 0)}
top_measurement = 1001
rev_top_measurement = 1001
L[y_top] = 10, L[y_top_rev] = 10
old y value 8 L[y] = 2
y_primed value 9 L[y] = 10
new y value 8 L[y] = 2
finished run 2 / 5
y = 8 | 1000
L[y] = 2
truth table = ['0', '0', '0', '0', '0', '1', '0', '0', '0', '0', '0', '1', '0', '0', '0', '0']

Boolean Functions for Oracle
```

```
 (  not  {0}  and  {1}  and  not  {2}  and  {3}  )  clause for marking y=5 L[y]=0
 (  {0}  and  not  {1}  and  {2}  and  {3}  )  clause for marking y=11 L[y]=1
truth table = ['0', '0', '0', '0', '0', '1', '0', '0', '0', '0', '0', '1', '0', '0', '0', '0']

Boolean Functions for Oracle
 (  not  {0}  and  {1}  and  not  {2}  and  {3}  )  clause for marking y=5 L[y]=0
 (  {0}  and  not  {1}  and  {2}  and  {3}  )  clause for marking y=11 L[y]=1
all measurements {
'1': (132, '0001', 11), '15': (135, '1111', 3),
'13': (113, '1101', 12), '5': (124, '0101', 0),
'7': (105, '0111', 6), '12': (142, '1100', 13),
'9': (121, '1001', 10), '2': (136, '0010', 8),
'3': (120, '0011', 15), '4': (138, '0100', 14),
'14': (129, '1110', 4), '10': (130, '1010', 9),
'8': (142, '1000', 2), '6': (121, '0110', 7),
'11': (124, '1011', 1), '0': (136, '0000', 5)
}
marked measurements {'5': (124, '0101', 0), '11': (124, '1011', 1)}
marked reversed measurements {'11': (113, '1011', 1), '5': (130, '0101', 0)}
top_measurement = 1100
rev_top_measurement = 0011
L[y_top] = 13, L[y_top_rev] = 15
old y value 8 L[y] = 2
y_primed value 12 L[y] = 13
new y value 8 L[y] = 2
finished run 3 / 5
y = 8 | 1000
L[y] = 2
truth table = ['0', '0', '0', '0', '0', '1', '0', '0', '0', '0', '0', '1', '0', '0', '0', '0']

Boolean Functions for Oracle
 (  not  {0}  and  {1}  and  not  {2}  and  {3}  )  clause for marking y=5 L[y]=0
 (  {0}  and  not  {1}  and  {2}  and  {3}  )  clause for marking y=11 L[y]=1
truth table = ['0', '0', '0', '0', '0', '1', '0', '0', '0', '0', '0', '1', '0', '0', '0', '0']

Boolean Functions for Oracle
 (  not  {0}  and  {1}  and  not  {2}  and  {3}  )  clause for marking y=5 L[y]=0
 (  {0}  and  not  {1}  and  {2}  and  {3}  )  clause for marking y=11 L[y]=1
all measurements {
'6': (126, '0110', 7), '14': (125, '1110', 4),
'5': (140, '0101', 0), '1': (121, '0001', 11),
'11': (130, '1011', 1), '10': (129, '1010', 9),
'15': (154, '1111', 3), '9': (115, '1001', 10),
'3': (134, '0011', 15), '7': (137, '0111', 6),
'13': (124, '1101', 12), '2': (126, '0010', 8),
'4': (144, '0100', 14), '0': (98, '0000', 5),
'12': (115, '1100', 13), '8': (130, '1000', 2)
}
marked measurements {'5': (140, '0101', 0), '11': (130, '1011', 1)}
marked reversed measurements {'5': (129, '0101', 0), '11': (124, '1011', 1)}
top_measurement = 1111
rev_top_measurement = 1111
L[y_top] = 3, L[y_top_rev] = 3
old y value 8 L[y] = 2
y_primed value 15 L[y] = 3
new y value 8 L[y] = 2
finished run 4 / 5
y = 8 | 1000
L[y] = 2
truth table = ['0', '0', '0', '0', '0', '1', '0', '0', '0', '0', '0', '1', '0', '0', '0', '0']

Boolean Functions for Oracle
 (  not  {0}  and  {1}  and  not  {2}  and  {3}  )  clause for marking y=5 L[y]=0
 (  {0}  and  not  {1}  and  {2}  and  {3}  )  clause for marking y=11 L[y]=1
truth table = ['0', '0', '0', '0', '0', '1', '0', '0', '0', '0', '0', '1', '0', '0', '0', '0']

Boolean Functions for Oracle
 (  not  {0}  and  {1}  and  not  {2}  and  {3}  )  clause for marking y=5 L[y]=0
 (  {0}  and  not  {1}  and  {2}  and  {3}  )  clause for marking y=11 L[y]=1
all measurements {
'6': (131, '0110', 7), '0': (127, '0000', 5),
'9': (136, '1001', 10), '12': (122, '1100', 13),
'5': (136, '0101', 0), '1': (125, '0001', 11),
'4': (136, '0100', 14), '2': (102, '0010', 8),
'3': (99, '0011', 15), '11': (132, '1011', 1),
'14': (157, '1110', 4), '13': (128, '1101', 12),
'15': (128, '1111', 3), '8': (116, '1000', 2),
'10': (129, '1010', 9), '7': (144, '0111', 6)
}
marked measurements {'5': (136, '0101', 0), '11': (132, '1011', 1)}
marked reversed measurements {'11': (128, '1011', 1), '5': (129, '0101', 0)}
top_measurement = 1110
```

```
rev_top_measurement = 0111
L[y_top] = 4, L[y_top_rev] = 6
old y value 8 L[y] = 2
y_primed value 14 L[y] = 4
new y value 8 L[y] = 2
finished run 5 / 5
final result of grover search, smallest y = 8 L[y] = 2
final result of y should be 5
```