

Logbook – Part 4

Table of Contents

Week 17	1
Logbook Circuit	2
Self Assessment	3
Week 20	4
Self Assessment	4
Week 21	5
Self Assessment	6
Week 22	7
Self Assessment	10

Week 17

Task 1

1. $3 \times a = 12 + 3i$
2. $-4i \times b = 8i - 24i^2$
3. $a + b = 2 + 7i$
4. $a - b = 6 - 5i$
5. $a \times b = -8 = 24i - 2i + 6i^2$
 $6i^2 + 22i - 8$
6. $a/b = \frac{4 + i}{-2 + 6i} \times \frac{-2 - 6i}{-2 - 6i}$

$$\frac{(4 + i)(-2 - 6i)}{(-2 + 6i)(-2 - 6i)} = \frac{-8 - 26i - 6i^2}{4 - 36i^2}$$

Task 2

9. $3.A = \begin{pmatrix} 0 & -12 & 9 & 0 \\ 3 & 6 & 0 & -6 \\ 0 & -9 & 3 & -3 \end{pmatrix}$

10. $-5.B = \begin{pmatrix} -5 & 20 & 0 & 0 & -10 \\ 0 & 5 & -15 & 10 & 0 \\ -5 & -5 & -5 & 0 & 5 \\ 0 & -10 & 15 & -15 & -20 \end{pmatrix}$

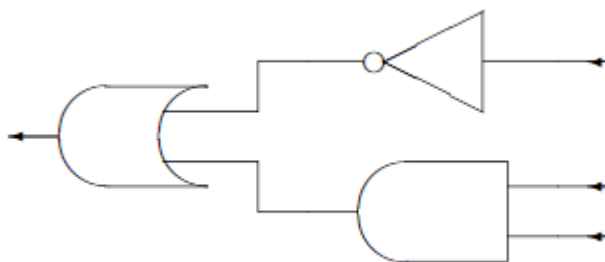
11. $A \cdot B =$

3	7	-9	8	-3
1	-10	12	-10	-6
1	2	-5	3	-5

12. A tensor B =

0	0	0	0	0	-4	16	0	0	-8	3	-12	0	0	6	0	0	0	0	0
0	0	0	0	0	0	4	-12	8	0	0	-3	9	-6	0	0	0	0	0	0
0	0	0	0	0	-4	-4	-4	0	4	3	3	3	0	-3	0	0	0	0	0
0	0	0	0	0	0	-8	12	-12	-16	0	6	-9	9	12	0	0	0	0	0
1	-4	0	0	2	2	-8	0	0	4	0	0	0	0	0	-2	8	0	0	-4
0	-1	3	-2	0	0	-2	6	-4	0	0	0	0	0	0	0	2	-6	4	0
1	1	1	0	-1	2	2	2	0	-2	0	0	0	0	0	-2	-2	-2	0	2
0	2	-3	3	4	0	4	-6	6	8	0	0	0	0	0	0	-4	6	-6	-8
0	0	0	0	0	-3	12	0	0	-6	1	-4	0	0	2	-1	4	0	0	-2
0	0	0	0	0	0	3	-9	6	0	0	-1	3	-2	0	0	1	-3	2	0
0	0	0	0	0	-3	-3	-3	0	3	1	1	1	0	-1	-1	-1	-1	0	1
0	0	0	0	0	0	-6	9	-9	-12	0	2	-3	3	4	0	-2	3	-3	-4

Logbook Circuit



0	1
1	0

Not matrix

1	1	1	0
0	0	0	1

And matrix

1	0	0	0
0	1	1	1

Or matrix

A	0	0	0	0	1	1	1	1
B	0	0	1	1	0	0	1	1
C	0	1	0	1	0	1	0	1
'A	1	1	1	1	0	0	0	0
B and C	0	0	0	1	0	0	0	1
'A or (B and C)	1	1	1	1	0	0	0	1

1 1 1 1 0 0 0 1 as a quantum matrix is

0 0 0 0 1 1 1 0

1 1 1 1 0 0 0 1

This was achieved by first assigning each of the inputs on the right an arbitrary name from top to bottom (a, b, and c respectively). I then filled out the rows for a, b, and c from left to right incrementing from 0 to 7, with the most significant bit (a) on the top, and the least significant (c) at the bottom. I then used the not matrix on A to invert the bits as they would through the gate.

I then used the And matrix on B and C to get the bit output with the given input states, and finally did an Or operation on these last two rows to get the final output for the possible bit sequences.

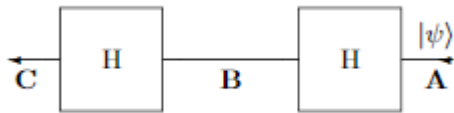
This bit sequence was then turned into a matrix representation of those bits.

Self Assessment

3/5

I would give myself 3 marks for this work as I have demonstrated the way in which a circuit can be represented as a truth table and how that table can create the circuit as a matrix, but I have not verified this with another method.

Week 20



Using this diagram, the input of any pure state will, at C, be the same pure state. This is due to the fact that the gate will turn a pure state into a 50 50 superposition, and turn a 50 50 superposition into a pure state by collapsing the superposition into a definitive state.

This table is read from right to left to copy the order of the diagram.

C	B	A	Input
[-0, 1]	[0,7071, -0.7071]	[0, 1]	1
[1, -0]	[0.7071, 0.7071]	[1, 0]	0

C	B	A	Input
100% 1	50% 1 50% 0	100% 1	1
100% 0	50% 1 50% 0	100% 0	0

As the output is an irrational number stored as an approximation, there are errors caused by performing maths on these approximate values – causing the negative zero in the output.

As the probabilities of these are 100% 50% 100% for both possible inputs at A B and C, this isn't very useful as you don't know what the 100% was for (100% 1 or 100% 0) unless you know the input. As you cannot see what has happened to the Qubit, the data is destroyed about its history which is not allowed within quantum computing as data-loss is not possible.

Self Assessment

3/5

I would give myself 3 marks for this weeks work as I have shown what happens when you chain two Hadamard gates together, but feel my explanation may be lacking in detail and I haven't proved this using another method to validate my result.

Week 21

This is the cubed code provided within the practical notes:

Logbook question. The algorithm shown below will calculate n^3 .

```
public static int cube(int n) {
    int i=0, cube = 0, threeNsqr = 0, threeN = 0;

    while (i < n) {
        cube = cube + threeNsqr + threeN + 1;
        threeNsqr = threeNsqr + 2*threeN + 3;
        threeN = threeN + 3;
        i++;
    }
    return cube;
}
```

Here it is again with assertions as proof of it's function:

```
Public static int cube(int n) {

    int i = 0; cube = 0; threeNsqr = 0, threeN = 0;

    [1]{cube = i3}

    While (i < n) {

        [2]{cube = i3, threeNsqr = 3i2, threeN = 3i}

        Cube = cube + threeNsqr + threeN + 1;

        [3]{cube = i3+3i2+3i+1 = (i + 1)3, threeNsqr = 3i2, threeN = 3i }

        threeNsqr = threeNsqr + 2*threeN + 3;

        [4]{cube = i3+3i2+3i+1 = (i + 1)3, threeNsqr = 3(i2 + 2i + 1)=3(i+i)2}

        threeN = threeN + 3;

        [5]{cube = i3+3i2+3i+1 = (i + 1)3, threeNsqr = 3(i2 + 2i + 1)=3(i+i)2, threeN = 3(i+1) }

        i ++;

        [6]{cube = i3, threeNsqr = 3i2, threeN = 3i}

    }

    [7]{Cube = i3 = n3, i = n}

    Return cube;

}
```

Eludications:

- [1] If the while loop is not triggered, i must be the same or larger than n . For the algorithm to be correct, n^3 must be the same as i^3
- [2] The variables must be the same as they were in [1] and [5], so $threeNsq$ must be $3i^2$ and $threeN$ must be $3i$.
- [3] This is where cubed is iterated, but no other variable is altered so they must stay the same and cubed must now be $(i + 1)^3$.
- [4] This is where $threeNsq$ is altered, and must be $3(i+1)^3$
- [5] This is where $threeN$ is altered, and must be $3(i+1)$
- [6] n^3 must be the same as i^3 to exit the loop, for which $threeNsq$ must be $3i^2$ and $threeN$ must be $3i$ as these are the final states to create the cube.
- [7] n^3 must be the same as i^3 for the return. To do this, i and n must be the same.

Based on how the program is expected to run, it is possible to make these eludications based on how the data should be manipulated given a specific area of a run time. If an if statement is expected to be used then the condition for it must be true. If a while loop is in use, the condition must be true during it, and false just after it.

These can be powerful assumptions, allowing for the proof above. This doesn't prove the code is perfect or will run without issue, just that the logic behind it should in theory work.

Self Assessment

4/5

I would award myself 4 marks for this work as I feel I have made a decent attempt at creating assertions within this code, but I don't feel that my eludications are entirely correct. I understand the concept that things must meet conditions at specific points within the execution based on the logic used within the method being tested.

I could have used an example, eg 5, and walked through it by hand. This would show the value like running the program in Debug mode. This would prove that the method and the assertions were correct and been enough for 4 or 5 marks, in my opinion.

Week 22

As my new methods are based off of the Quadratic model answer provided, Here are the Method() and Complexity() methods for the Cubic and Logarithmic classes.

Cubic:

```
/**
 * This method will execute in quadratic time.
 */
public void method(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            for (int k = 0; k < n; k++) {
                instruction();
            }
        }
    }
}


/**
 * This method will return a value that grows cubically.
 */
public int complexity(int n) {
    return (int) Math.pow(3,n);
}
```

Logarithmic:

```
/**
 * This method will execute in logarithmic time.
 */
public void method(int n) {
    for (int i = 0; i < n; i = i*2) {
        instruction();
    }
}


/**
 * This method will return a value that grows logarithmically.
 */
public int complexity(int n) {
    return (int) Math.log(n);
}
```

To get the time required for each method to execute, I used the following settings:


 Running Cubic: base size 5, run size 12, delay 2.0ms, time limit 1.000 minutes, size limit 10

OK


Here are the results:

 Cubic(1) took 2.843 milliseconds, expected time was 4.372 milliseconds


OK

 Cubic(2) took 22.512 milliseconds, expected time was 13.117 milliseconds


OK

 Cubic(3) took 77.094 milliseconds, expected time was 39.351 milliseconds

OK

 Cubic(4) took 175.566 milliseconds, expected time was 118.053 milliseconds

OK

 Cubic(5) took 348.481 milliseconds, expected time was 354.158 milliseconds

OK



Cubic(6) took 601.187 milliseconds, expected time was 1.062 seconds

OK



Cubic(7) took 961.819 milliseconds, expected time was 3.187 seconds

OK



Cubic(8) took 1.420 seconds, expected time was 9.562 seconds

OK



Cubic(9) took 2.039 seconds, expected time was 28.687 seconds

OK



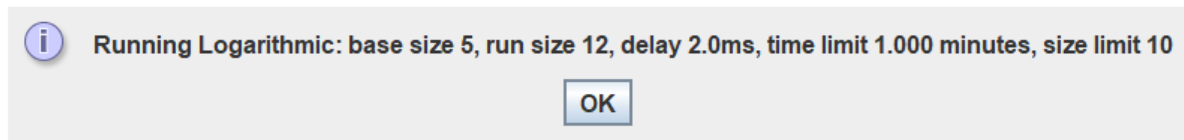
Cubic(10) took 2.787 seconds, expected time was 1.434 minutes

OK

Here it is again as a console output:

```
Cubic: 1-10 Expected    Actual
1    4.372    2.843
2   13.117   22.512
3   39.351   77.094
4  118.053  175.566
5  354.158  348.481
6 1,062.474  601.187
7 3,187.422  961.819
8 9,562.267 1,420.032
9 28,686.802 2,039.2
10 86,060.406 2,787.129
```

Here are the settings for the Logarithmic method:



Unfortunately, I got no results (or errors) while running the solution – meaning there is likely an issue with the implementation of my Method().

The cubic Method() was created with three nested for loops. As each requires N number of implementations within itself, each nested N adds another power to the value of N. As there are 3 loops, the complexity is $O(n^3)$ for this method.

The logarithmic Method() is more complicated. I used the loop:

```
for (int i = 0; i < n; i = i*2)
```

This increments N by doubling it each time – creating a logarithm around N.

However I don't know if this is correct as I get no output (I assume through an infinite loop) so cannot back this up.

Self Assessment

2/5

I would give myself 2 marks for this work as I have made an attempt, and feel that I understand the simpler exponential growth much better than the logarithmic change in the execution time of the method.