1 Halt or Not?

1. It always halts. If \( \text{?} < 1 \), the script halts without making any 'pop'. If \( \text{?} \geq 1 \), we make 'pop' and keep decreasing count. Any value of count will eventually fall below 1 and the script will halt.

2. No value of count will ever satisfy \( \text{count} < \text{count} - 1 \). Therefore, it will never halt.

3. We are increasing the value of count. And the script stops when count \( \leq 64 \). So, any value \( \geq 64 \) will never fall below 64. So, the script halts only when the value of \( \text{?} < 64 \).

4. Since we are decreasing count, if we initialize it with anything \( \leq 1024 \), we will never get \( \text{count} > 1024 \). So, the script halts only when \( \text{?} > 1024 \).

2 Huffman Coding

The given table has a 5 bit code for each letter. We have 8 letters, so, it should take exactly 40 bits. The bit sequence would be 11001 11100 01110 10001 01110 01100 01101 11001. Some of you have only answered 40. But note that we are asking you to write out the whole bit sequence.

The exact code for each letter depends on the tree you make. But any tree will give you a code such that the length of t's code is smaller than the length of e's code. This is the way Huffman codes ensure compression (saving of bits).

I got a 20-bit bit sequence. For answers I have accepted 19/20/21 bits. If you got anything else, you have to convince me that you made a valid Tree and a valid Huffman code. I also took off some marks for not giving any clue as to how you got 20 (or whatever) bits.

And needless to say, we save 20 bits when we use Huffman codes.

3 Growth Rates

I solved it as follows: We have a \( n \times n \) grid of streets and avenues. So, there are \( n^2 \) intersections. 'Spiraling Inwards' needs you to touch every intersection.
Therefore, it is $O(n^2)$.

4 Halt or Not?

1. If $? \leq 55$, and we are increasing count, eventually we will have $\text{count} > 55$ and halt. If $? > 55$, we halt without any 'pop'. So, the script halts for any value of $?$. 

2. $\text{count} > \text{count} + 1$ cannot be satisfied by any value of count. So, this one does not halt for any value.

3. Any positive number (except 0) grows if you multiply it with 2. If $? = 0$, the multiplication will not change the value of count. Negative numbers keep getting smaller. So, any $? > 0$ will eventually stop.

4. This will stop for any value as well. If $? < 2$ we stop without 'pop'. And if $? \geq 2$, since count is decreasing at every iteration, it will eventually fall below 2 and stop.