1) (10 pts) ANL (Algorithm Analysis)

Consider a n-bit binary counter, which starts with the binary representation of 0 and increments by 1 until it reaches the binary value of $2^n - 1$. For n = 3, the counter would start at 000, and then change as follows:

$$000 \rightarrow 00\underline{1} \rightarrow 0\underline{10} \rightarrow 01\underline{1} \rightarrow \underline{100} \rightarrow 10\underline{1} \rightarrow 1\underline{10} \rightarrow 11\underline{1}.$$

The underlined bits represent the ones that had to be changed. In particular, for this example, 1 + 2 + 1 + 3 + 1 + 2 + 1 = 11 bits were changed as the counter progressed from 0 to $2^n - 1$. Let f(n) equal the number of bits that are changed for an n-bit binary counter counting from 0 to $2^n - 1$. Find a closed-form formula for f(n). (For example, something like $f(n) = 2^{n-1} + 2$. A formula in terms of n without any sort of recursive function definition.) Show all of your work and put a box around your final answer.

They key observation is that the lowest-order zero bit in the binary number controls the number of bits that get changed with the counter increments. Thus, when the binary counter is 101011$\underline{0}$111, for example, and we note that the lowest-order zero bit is in the fourth position, counting from the right (1-based counting), we know that exactly 4 bits will flip when the counter increments to 1010111000.

In terms of n, the counter ends in a 0 2^{n-1} times (half of the 2^n numbers displayed on the counter). In these cases, 1 bit gets flipped.

In terms of n, the counter ends in 01 2^{n-2} times and will get flipped twice in these cases.

In terms of n, the counter ends in 011 2^{n-3} times and will get flipped three times in these cases.

This pattern persists for each value of k, as k ranges from 1 to n. (The counter ends in 011..1 exactly 2^{n-n}, or 1 time and all n bits flip this one time.)

It follows that $f(n) = \sum_{k=1}^{n} k2^{n-k}$. Let's evaluate this sum. We first write it down, and then we take the whole expression and divide it by 2 and write down the corresponding sum below the original. Then we subtract the bottom equation from the top:

$$f(n) = 1 \times 2^{n-1} + 2 \times 2^{n-2} + 3 \times 2^{n-3} + \ldots + n \times 2^0$$

$$f(n)/2 = 1 \times 2^{n-2} + 2 \times 2^{n-3} + \ldots + (n-1) \times 2^0 + n \times 2^{-1}$$

$$f(n) - f(n)/2 = 2^{n-1} + 2^{n-2} + 2^{n-3} + \quad + 2^0 - n/2$$

$$f(n)/2 = 2^{n-1} + 2^{n-2} + 2^{n-3} + \quad + 2^0 - n/2$$

Use the geometric sum formula to evaluate the sum on the right except for the last term:

$$f(n)/2 = (2^n - 1) - n/2$$

Multiply by 2 to get

$f(n) = 2(2^n - 1) - n = 2^{n+1} - 2 - n =$ $\underline{\mathbf{2^{n+1} - n - 2.}}$

Alternate Solution to #1

Instead of summing each number in the original problem individually (1 + 2 + 1 + 3 + 1 + 2 + 1, which we decomposed into 4 x 1 + 2 x 2 + 1 x 3), we can view the problem differently and count how many times each bit gets flipped. The least significant bit gets flipped every time, or $2^n - 1$ times. The second least significant bit gets flipped slightly less than half of that, exactly $2^{n-1} - 1$ times. (To see this, note that we flip this bit every other time, with the flip occurring on the second of each pair. Formally, we flip this bit $\left\lfloor \frac{2^n-1}{2} \right\rfloor = 2^{n-1} - 1$ times. More generally, the kth least significant bit gets flipped exactly $\left\lfloor \frac{2^n-1}{2^{k-1}} \right\rfloor = 2^{n-k+1} - 1$ times. Thus, we can add up the total number of bit flips by adding the number of times each individual bit itself gets flipped, giving us the following summation to evaluate:

$$\sum_{k=1}^{n} 2^{n-k+1} - 1$$

$$\sum_{k=1}^{n} (2^{n-k+1} - 1) = \left(\sum_{k=1}^{n} 2^{n-k+1} \right) - (\sum_{k=1}^{n} 1)$$

$$= \left(\sum_{k=1}^{n} 2^k \right) - n$$

$$= 2(2^n - 1) - n$$

$$= 2^{n+1} - 2 - n$$

$$= 2^{n+1} - n - 2$$

Yet a third way to view this problem is to let T(n) be the answer to the question for an n-bit counter. Using the observation above where we note that the least significant bit flips every time ($2^n - 1$), notice that the remaining n – 1 bits are essentially playing the role of an n – 1 bit-counter. (Basically, the n – 1 most significant bits stay frozen every other step and then just count regularly on the even numbered steps. This means that $T(n) = T(n-1) + 2^n - 1$. The solution to this recurrence is the summation above.

Grading: 4 pts for setting up summation or recurrence relation which corresponds to the answer to the question.

6 pts for evaluating the derived summation or recurrence relation.

Give partial credit as you see fit for both parts.

If initial summation is incorrect but that incorrect sum is evaluated correctly, give a maximum of four points out of six for evaluating the summation, depending on complexity of it.