COP 3502 Quiz #3 Version D (Algorithm Analysis, Sorting) Solutions

1) (5 pts) A sorting algorithm sorts n values in O(nlgn) time. When run on an input of size $n = 2^{16}$, the algorithm takes 8 ms. How long, **in seconds**, is the algorithm expected to take when run on an input of size $n = 2^{24}$?

Let the algorithm take T(n) = cnlgn time for an input size of n, for some constant c. Using the given information, solve for c. We can choose the base for our logarithm (and keep it the same), so we can choose base 2:

$$T(2^{16}) = c(2^{16})log_2 2^{16} = 8ms$$
$$16c(2^{16}) = 8\,ms$$
$$c = \frac{8ms}{2^{20}} = \frac{1ms}{2^{17}}$$

Now, solve for $T(2^{24})$:

$$T(2^{24}) = c(2^{24})log_2 2^{24} = \frac{1ms}{2^{17}}(2^{24})24 = (2^7)(24)ms = 128 \times 24ms = 3072\,ms$$
$$= \mathbf{3.072\ sec}$$

Grading: 2 pts solving for c, 2 pts solving for $T(2^{24})$, 1 pt converting to seconds

2) (8 pts) Give a closed form solution for the following summation in terms of n:

$$\sum_{i=1}^{2n}(2^i - i)$$

$$\sum_{i=1}^{2n}(2^i - i) = \sum_{i=1}^{2n}2^i - \sum_{i=1}^{2n} i$$

$$= \left(\sum_{i=0}^{2n}2^i\right) - 2^0 - \frac{2n(2n+1)}{2}$$

$$= \frac{2^{2n+1} - 1}{2 - 1} - 1 - n(2n+1)$$

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

Grading: 1 pt split, 3 pts left sum, 2 pts right sum, 2 pts simplify

3) (10 pts) Use the iteration technique to solve the following recurrence relation:

$$T(n) = 4T(n-1) + 2^n, for\ all\ integers\ n > 0$$
$$T(0) = 1$$

Please provide your answer as an exact function that is a closed form representation of T(n).

$$T(n) = 4T(n-1) + 2^n$$

Now, let's substitute for T(n-1):

$$T(n) = 4(4T(n-2) + 2^{n-1}) + 2^n$$

$$T(n) = 16T(n-2) + (2^{n+1} + 2^n)$$

Now, let's substitute for T(n-2):

$$T(n) = 16(4T(n-3) + 2^{n-2}) + +(2^{n+1} + 2^n)$$

$$T(n) = 64T(n-3) + (2^{n+2} + 2^{n+1} + 2^n)$$

After k iterations, we have:

$$T(n) = 4^k T(n-k) + \sum_{i=0}^{k-1} 2^{n+i}$$

Let k = n and substitute:

$$T(n) = 4^n T(n-n) + \sum_{i=0}^{n-1} 2^{n+i}$$

$$T(n) = 4^n T(0) + \sum_{i=0}^{n-1} 2^n 2^i$$

$$T(n) = 4^n + 2^n \sum_{i=0}^{n-1} 2^i$$

$$T(n) = 4^n + 2^n(2^n - 1) = 4^n + 4^n - 2^n = \mathbf{2^{2n+1} - 2^n}$$

Grading: 1 pt copying down first iteration, 2 pts to get to second iteration, 2 pts to get to third iteration, 2 pts for guess to k, 1 pt to plug in for k, 2 pts to finish it up.

4) (5 pts) Show the contents of the following array after each iteration of Bubble Sort:

Initial Values	12	77	16	8	4	13	9
1st iteration	12	16	8	4	13	9	77
2nd iteration	12	8	4	13	9	16	77
3rd iteration	8	4	12	9	13	16	77
4th iteration	4	8	9	12	13	16	77
5th iteration	4	8	9	12	13	16	77
Last iteration	4	8	9	12	13	16	77

Grading: 1 pt per row, row must be perfect to get the point.

5) (4 pts) Show the result of partitioning the array below, using the leftmost element as the partition element. Please use the in-place partitioning algorithm shown in class.

Initial Values	43	13	79	82	8	16	88	31	27	97	4
After Partition	31	13	4	27	8	16	43	88	82	97	79

Grading: 4 pts if all 11 values correct, 3 pts if 9 or 10 are correct, 2 pts if 6, 7 or 8 are correct, 1 pt if 3, 4 or 5 are correct, 0 otherwise.

6) (3 pts) Which sort is shown below, in its recursive implementation?

```
void whatsort2(int* array, int n) {
    if (n == 1) return;
    for (int i=1; i<n; i++) {
        if (array[i-1] > array[i]) {
            int tmp = array[i-1];
            array[i-1] = array[i];
            array[i] = tmp;
        }
    }
    whatsort2(array, n-1);
}
```

Bubble Sort (3 pts all or nothing)