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

1) (5 pts) An algorithm processes n values in $O(n^3)$ time. When run on an input of size n = 400, the algorithm takes 500 ms. How long, **in seconds**, is the algorithm expected to take when run on an input of size n = 960? **(Note: $12^3 = 1728$.)**

Let $T(n) = cn^3$ be the amount of time the algorithm takes on an input of size n for some constant c. Using the given information, we have:

$$T(400) = c(400)^3 = 500ms$$
$$c = \frac{500}{400^3}ms$$

Now, let's solve for T(960)?

$$T(960) = c(960)^3 = \frac{500ms}{400^3} \times 960^3 = (500ms) \times \left(\frac{960}{400}\right)^3 = (500ms) \times (\frac{12}{5})^3$$
$$= (500ms) \times \frac{1728}{125} = 4 \times 1728ms = 6912ms = \mathbf{6.912seconds}$$

Grading: 2 pts solve for c, 2 pts solve for T(960), 1 pt convert to seconds

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

$$\sum_{i=n}^{2n}(\sum_{j=0}^{i-1}(2^j))$$

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

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

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

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

Grading: 2 pts inner sum, 2 pts split for bounds, 3 pts evalulating both sums, 1 pt simplification

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

$$T(n) = 3T\left(\frac{n}{3}\right) + n^2, for\ all\ integers\ n > 0$$
$$T(1) = 1$$

Please provide your answer as **Big-Oh** bound on T(n).

$$T(n) = 3T\left(\frac{n}{3}\right) + n^2$$

Now, let's substitute for T(n/3):

$$T(n) = 3\left(3T\left(\frac{n}{9}\right) + \left(\frac{n}{3}\right)^2\right) + n^2$$
$$T(n) = 9T\left(\frac{n}{9}\right) + (\frac{n^2}{3} + n^2)$$

Now, let's substitute for T(n/9):

$$T(n) = 9\left(3T\left(\frac{n}{27}\right) + \left(\frac{n}{9}\right)^2\right) + +(\frac{n^2}{3} + n^2)$$

$$T(n) = 27T\left(\frac{n}{27}\right) + (\frac{n^2}{9} + \frac{n^2}{3} + n^2)$$

After k iterations, we have:

$$T(n) = 3^k T\left(\frac{n}{3^k}\right) + n^2 \sum_{i=0}^{k-1} \frac{1}{3^i}$$

Plug in $\frac{n}{3^k} = 1$, bounding the summation by taking it to infinity:

$$T(n) \leq nT(1) + n^2 \sum_{i=0}^{\infty} \frac{1}{3^i}$$

$$T(n) \leq n + n^2\left(\frac{1}{1-\frac{1}{3}}\right) = n + \frac{3}{2}n^2 = \boldsymbol{O(n^2)}$$

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	44	33	22	25	18	16	31
1st iteration	33	22	25	18	16	31	44
2nd iteration	22	25	18	16	31	33	44
3rd iteration	22	18	16	25	31	33	44
4th iteration	18	16	22	25	31	33	44
5th iteration	16	18	22	25	31	33	44
Last iteration	16	18	22	25	31	33	44

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	15	6	19	27	13	22	3	12	44	18	2
After Partition	3	6	2	12	13	15	22	27	44	18	19

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 whatsort(int* array, int n) {
    if (n == 1) return;
    int x = 0;
    for (int i=1; i<n; i++)
        if (array[i] > array[x])
            x = i;
    int tmp = array[x];
    array[x] = array[n-1];
    array[n-1] = tmp;
    whatsort(array, n-1);
}
```

Selection Sort (3 pts all or nothing)