# TCSS 343 - Assignment 1

Jake McKenzie

July 1, 2018

(3 points) 1. Below is a self-reduction for the `MAX` problem. State a recursive algorithm using pseudocode for finding the maximum element based on this self-reduction.

**Algorithm 1** Find Max integer in an Array with simple recursion

```
1: procedure FIND MAX(A)
2:     if (a == b) then
3:         return A[a]
4:     else if (a < b) then
5:         return Max(A[a],Find Max(A[a + 1]))
6:     end if
7: end procedure
8: procedure MAX(a, b) return (a < b) ? b : a
9: end procedure
```

(6 points) 2. Using the same reduction as part 1 now state a recurrence $T(n)$ that expresses the worst case run time of the recursive algorithm. Find a similar recurrence in your notes and state the tight bound on $T(n)$.

Line 3 makes 1 amount of operations while line 5 makes $T(n-1)$, this is because there are $n-1$ amount of comparisons to check for the max in the recurrence for when this list is greater than 1. **Note**: Consistency of whether the constant amount of operations is writen as 1 or $O(1)$ are inconsistent so I went with the notation I've seen the most used often.

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ T(n-1) + 1 & \text{if } n > 1 \end{cases}$$

Claim: $\forall n > 0$, the running time of *Find Max* $\epsilon O(n)$. We consder the recurrance relation above.

1. Base Case:

$$n = 1; T(1) = 1$$

2. Inductive Hypothesis:

$$T(k) = \begin{cases} 1 & \text{if } k = 1 \\ T(k-1) + 1 & \text{if } k > 1 \end{cases}$$

Assume for an arbitrary $k, T(k) \leq k$

3. Inductive Step:

$$\text{if } k+1 > 1$$
$$T(k+1) = T(k) + 1$$
$$T(k+1) = k + 1 + 1$$
$$T(k+1) = k + 2$$
$$T(k+1) \epsilon O(k)$$

(9 points) 3. Below is a self-reduction for the `MAX` problem. State a recursive algorithm using pseudocode for finding the maximum element based on this self-reduction.

$$M(A[a \dots b]) = \begin{cases} -\infty & \text{if } a > b \\ A[a] & \text{if } a = b \\ \max(M(A[a \dots t_1]), \max(M(A[t_1 + 1 \dots t_2]), M(A[t_2 + 1 \dots b]))) & \text{if } a < b \end{cases}$$

For what it's worth, I don't think this algorithm will find the max element if it

**Algorithm 2** Find Max integer in an Array with 3-Way Split

```
1: procedure FIND MAX(A, a, t1, t2)
2:     if (a > b) then
3:         return 0x7FFFFFFF
4:     else if (a == b) then
5:         return A[a]
6:     else if (a < b) then
7:         return Max(FindMax(A,a,t1,t2), Max(FindMax(A, a, t1+1,t2), FindMax(A, a,
   t1,t2+1]))
8:     end if
9: end procedure
10: procedure MAX(a, b) return (a < b) ? b : a
11: end procedure
```

is contained within the first third of the array(we never iterate over the first third elements, only the second and last third), but it does match the self-reduction.

(7 points) 4. Using the same reduction as part 3 now state a recurrence $T(n)$ that expresses the worst case run time of the recursive algorithm. You do not need to formally prove your recurrence, but you have to show that it is a reasonable guess by using a recursion tree or by the repeated substitution method. *Hint: assume that n is a power of 3.*