477 Homework 3:

Saturday, February 27, 2021 2:17 PM

(1)

```
for (i=0; i<n-1; i++)        T(n-2)
    for (j=i+1; j≤n-1; j++)  T(n-j-1)
        if A(i) = A(j)
            return False;
return True
```

a) This algorithim take an array of size n and checks for duplicate values, if duplicates exist, it returns false. The Outer loop only goes to n-2 because there would be nothing to check the last element against that hasn't already been checked.

b) Outer loop works from 0 to n-2, So it operates $(n-2)-\emptyset+1 = (n-1)$ times (end, start, account for $\emptyset$ index)

The inner loop goes from $J=i+1$ to $J=n-1$ (start, end, index)

1st iteration: $i=0$, so $J=1$ to $J=n-1$ $\quad (n-1)-1+1 = n-1$

2nd iteration: $i=1$, so $J=2$ to $J=n-1$ $\quad (n-1)-2+1 = n-2$

...

until

$i=n-2$, $J=n-1$ to $J=n-1$ $\quad (n-1)-(n-\frac{1}{2})+1 = 1$

The Series $\sum_{i=0}^{n-1} i$ or $\sum_{i=1}^{n-1} i$ we know that worst case is $\frac{n(n+1)}{2}$

So we plug in our n-1 from the outer loop

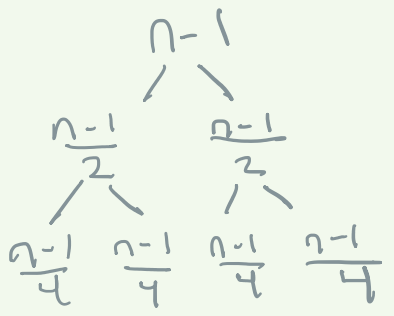
$$\frac{(n-1)(n-1+1)}{2} = \frac{(n-1)n}{2} = \frac{n^2-n}{2}$$

Leading term is $\frac{n^2}{2}$ So worst case is $\boxed{O(n^2)}$

(2) b) For several largest values in an array it will return the first instance, with the given it will be index 2

c) Line 19 & 20 are the key comparisons, 19 being the recursive function of worst case complexity $2T(\frac{n}{2})$ and worst case a line 20 is n-1

So : $2T(n/2) + (n-1)$



Problem Size	Problem Count
$n-1$	2^0
$\frac{n-1}{2}$	2^1
$\frac{n-1}{4}$	2^2
$\frac{n-1}{2^i}$	2^i

$\frac{n-1}{2^i} = 1$

$n-1 = 2^i$

$i = \lg(n-1)$

So $O(n)$

```
michaelsmac@Michaels-MacBook-Pro Homework-3 % ./problem2
The first index with the largest value is: 2
michaelsmac@Michaels-MacBook-Pro Homework-3 %
```