CMPU 241 - Algorithmics

Spring 2019

Assignment 1 – Due February 5th, 2019

NAME: Jonah Tuckman

---

1. (10 points) Given the following algorithm:

Algorithm UniqueElements(A)

Input: n-element array of numbers, A$[1 \ldots n]$
Output: Returns "true" if all items are unique and "false" otherwise

```
1. for ( i = 1 to n−1 )
2.     for ( j = i+1 to n )
3.         if ( A[ i ] == A[ j ] ) return false
4. return true
```

(a) Is there a difference in $T(n)$ (measured by number of operations executed) for best- and worst-case input?

If your answer is "yes", give examples of best- and worst-case inputs. If your answer is "no", explain why there is no difference between best- and worst-case inputs.

Yes there is a difference between the best and worst case input. For this algorithm to work the length of the input array A must be > 1. Otherwise there would be an out of range error as j would be $i = 1 + 1 = 2$ which is not in range of the array of length 1.

Best Case: The best case would be if the array A had the same value at index 0 and 1. This would then mean that the array does one single check and because the two values are indeed equal the loops breaks and return false rather than continuing to check.

Worst Case: The worst case in this algorithm would be if the array A did indeed have all unique items. This would mean that each loop would have to run in completion before exiting with the return value of true.

(b) Give the line number(s) of the basic operation.

Express the running time using θ notation if there is no difference between best-case and worst-case running times, or give best-case and worst-case asymptotic running time using O and Ω if there is a difference.

The line number for the basic operation is line 3.

Best Case: $T(n) = O(1)$

This is constant run time because there is one single loop for each.
The loop breaks after the first comparison thus the data size will not influence the run time.

Worst Case: $T(n) = \Omega(n^2)$

Run time is omega of n^2 in the worst case because there are two complete loops that are being run.
The loop in line 1 runs $n - 1$ times (which we can assume to be n times.
The loop in line 2 runs $n - 1$ times which we will again call n times.
Because both loops run n times, the total run time in the worst case when the program will run in both loops completely is $\Omega(n^2)$.