1. Determine the Big-O Running Time (tight-bound) of Bubble Sort

|  |  |
| --- | --- |
| Bubble Sort Pseudo Code:  \*NOTE: ASSUMES ARRAY STARTS AT INDEX 1  \*NOTE: COUNT COMPARISONS AS CRITICAL OPERATIONS  BubbleSort(A, n):  // A is array, n is items to sort  For i = n – 1 to 1:  For j = 1 to i:  if A[ j ] > A[ j + 1]:  Swap(A[ j ], A[ j + 1]) | My Observations:  Operations:   1. Iterate through A (the array) times 2. Another loop that iterates times 3. Compare the current element with the next element and see if the current is greater than the latter. 4. Swap the current element with the next element if the condition stated in *Step 3* is True.   Derivations:  1st for loop:  2nd for loop:      3rd Operation: times (assuming the condition for each iteration evaluates to True).  4th Operation: times (again, assuming that the condition in *Operation 3* evaluates to True).  Therefore, we have: |

1. Prove that is . Is this a tight upper bound? Why or why not? Use textbook definition for the proof, for example state the general definition, then state what T(n) and F(n) are in this case, and what needs to be shown in this case.

What we know:

1. . **T(n) represents the runtime of an algorithm.**
2. T(n) is O(F(n)) means there is a fixed threshold n0 and some positive constant multiplier such that
3. c(F(n)) is an asymptotic upper bound on T(n).
4. F(n) is n6

Given:

WTS:

is .

Proof:

|  |  |
| --- | --- |
| Let | Definition of Big-O  Substitute given values for T(n) and F(n)  Let n0 = 1 and substitute value into equation |

**Is this a tight upper bound? Why or why not?** No, this is not a tight upper bound. The reason being is that this is not the smallest function F(n) where T(n) is O(F(n)) – the smallest function F(n) where T(n) is O(F(n)) is n5.