Antonio Scalfaro

CMSC 451

05/20/2024

**Homework 1**

1. Int square(int n)

If(n ==1) return 1

Return square(n-1) + 2 \* n – 1

//Precondition – n > 0

//Postcondition – n2 is returned

Proof by Induction on n:

Base case: n = 1

Returns 1 = 12 By function base case

Inductive case: We assume that it is true for n = k -1, so we must show it is true for n = k, where n > 1.

Return value = square(n – 1) + 2 \* n – 1 From program

= (n-1)2 + 2 \* n – 1 By Inductive Hypothesis

= n2 – 2n + 1 + 2n – 1 By Algebra

= n2 By Algebra

1. F(n) = 3n2 + 9

G(n) = 51n + 17

G(n) < F(n)

51n + 17 < 3n2 + 9

3n2 – 51n + 9 - 17 > 0

3n2 – 51n – 8 > 0

Solve by quadratic formula:

N = 17.15, n = -.15

Since negative answers are removed from solution set, it must be n = 17.15. Also, due to this counting n as an int, we round up to n = 18. When n ≥ 18, f(n) becomes more efficient than g(n).

1. Inner loop runs from i = 0 to i = n -1, n = length of the array, for a total of n times.

Outer loop runs from j = n – 1 to j = i + 1 for a total of (n – 1) – i times.

Worst case total swaps (S):

S = (n – 1) + (n – 2) + (n – 3) + . . . + 1 + 0

Therefor the sum of the natural numbers n from 1 (exclude 0) to n -1 is:

Therefor the final sum for S is:

1. Initial condition is : T(0) = C, C = some constant time for checking the base case.

Recurrence relation for T(n): T(n) = T(n-2) + O(1), O(1) refers to constant time operation performed before recursive call.

Therefor it is T(n) = T(n-2) + C.

Because it is only even integers, we substitute (n-2) + C with (n – 2k) + kC, where k ≥ 1.

Dropping the constant gives us T(n) = n – 2k, n ≥ 1, k ≥ 1 for the recurrence. This shows the algorithm grows linearly as the size of the array grows.