Recitation Class 2

Q1 – If $f(n) = 100n^2 + log(n)$ and $g(n) = n^2 + log^2(n)$, which ones are correct

(a) $f(n) = O(g(n))$
(b) $f(n) = \Omega(g(n))$
(c) $f(n) = \Theta(g(n))$

Answer

Assuming the base of logarithm is a:

(a) True. Take $c = 100$ and $n_0 = a$.
(b) True. Take $c = 1$ and and $n_0 = a$.
(c) True. Correctness of (a) and (b) directly implies (c).

Q2 – Prove or disprove the following statement

$$f_1(n) = \Theta\big(g_1(n)\big)\ and\ f_2(n) = \Theta\big(g_2(n)\big) \Longrightarrow f_1(n) + f_2(n) = \Theta(g_1(n) + g_2(n))$$

Answer

By definition we have

$$c_1 g_1(n) \leq f_1(n) \leq c_1' g_1(n) \quad for\ n \geq n_1$$

And

$$c_2 g_2(n) \leq f_2(n) \leq c_1' g_2(n) \quad for\ n \geq n_2$$

Take $c = \min\{c_1, c_2\}$ and $c' = \max\{c_1', c_2{}'\}$ and $n' = \max\{n_1, n_2\}$, then we have

$$c(g_1(n) + g_2(n)) \leq f_1(n) + f_2(n) \leq c'(g_1(n) + g_2(n)) \quad for\ n \geq n'$$

This implies that

$$f_1(n) + f_2(n) = \Theta((g_1(n) + g_2(n))$$

Q3 – The pseudocode for the well-known insertion sort algorithm is

```
INSERTION-SORT(A)
for j = 2 to A.length
    key = A[j]
    // Insert A[j] into the sorted sequence A[1 .. j − 1].
    i = j − 1
    while i > 0 and A[i] > key
        A[i + 1] = A[i]
        i = i − 1
    A[i + 1] = key
```

What is the best-case running time of Insertion Sort? On what input array does it occur? What is the worst-case running time of Insertion Sort? On what input array does it happen?

Answer

The best-case running time of Insertion sort is $O(n)$, and it occurs on the sorted array $< 1, 2, \ldots, n >$. The worst-case running time of Insertion sort is $O(n^2)$, which occurs on the array $< n, n-1, \ldots, 1 >$.

Q4 – Let $A[1..n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is an inversion of A. Now,

(A) List the inversions of $\langle 2,3,8,6,1 \rangle$.

(B) What array with elements from $\{1,2, \ldots, n\}$ has the most inversions? How many?

(C) What is the relation between the running time of Insertion Sort and the number of inversions?

Answer

(A) List of inversions: (2,1), (3,1), (8,1), (6,1), (8,6)

(B) Array <n, n-1, …, 3, 2, 1> has the most number of inversions, which is $n(n-1)/2$.

(C) Let T(n) denote the running time of insertion sort. $T(n) = c_1 n + c_2(\#inversions)$