**Lingnan Liu (ll656)**

**Asymptotic complexity problems**

**1. Prove formally that function is .**

Ans: We know that if . Thus is for and .

**2. Prove formally that function is . Yes, this seems to contradict exercise 1. Give an explanation for this.**

Ans: We know that if . Thus is for and .

This conclusion is reasonable. As shown in the formal definition of Big-O, the terms including higher order of n is growing faster than those with lower order of n. Thus, the higher Big-O expression will cover the lower Big-O expression. However, we usually choose the lowest possible order Big-O expression to precisely express the asymptotic complexity of programs.

**3. Prove formally that function is .**

Ans: We know that if . Thus is for and .

**4. Prove formally that function is .**

Ans: We know that , for .

Thus, is for and .

**5. Ans:**

The worst-case order of execution should be . In the worst case, there are no occurrence of x in s, thus the iteration part will be executed by times. In each iteration, the function will compare between x and the substring of s which consumes execution times. Therefore, the altogether execution time order is .

In the average case, I think the time consuming of string comparison in the iteration part is a constant. That is, the expected time order should be .

**Induction problems**

**1. Ans:**

Base case: , .

Assume that holds, .

Then , therefore holds.

To sum up, for all , holds.

**2. Ans:**

Base case: , .

Assume that , holds, .

Then, , therefor holds.

To sum up, for all , holds.

**3. Ans:**

Base case: is divisible by 3.

Assume that , holds, is divisible by 3.

Then for : .

Because and are both divisible by 3, is also divisible by 3 as the addition of those two items. holds.

Thus, for all , is divisible by 3.

**4. Ans:**

Base case: , .

Assume that , holds: .

Then for : .

Thus, holds and for any , holds: .

**5. Ans:**

Base case: . In this case, there is only one node in the tree, thus .

From the source code, the function returns 1, which satisfies its specification.

Assume that , the function satisfies its specification, a.k.a it returns .

Now we add one more node to top of the tree, making . The original tree becomes the left (or right) subtree of the new node.

From the source code, the return value is . Thus , the function still satisfies its specification.

To sum up, the function satisfies its specification for all .