Lingnan Liu (ll656)

Asymptotic complexity problems

1. Prove formally that function $g(n) = 2n + 2$ is $O(n)$.

Ans: We know that $2n + 2 \leq 3n$ if $n \geq 2$. Thus $g(n) = 2n + 2$ is $O(n)$ for $c = 3$ and $N = 2$.

2. Prove formally that function $g(n) = 2n + 2$ is $O(n^2)$. Yes, this seems to contradict exercise 1. Give an explanation for this.

Ans: We know that $2n + 2 \leq n^2$ if $n \geq 3$. Thus $g(n) = 2n + 2$ is $O(n^2)$ for $c = 1$ and $N = 3$.

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 $h(n) = 7n^2 + 2n + 1000$ is $O(n^2)$.

Ans: We know that $7n^2 + 2n + 1000 \leq 8n^2$ if $n \geq 33$. Thus $h(n) = 7n^2 + 2n + 1000$ is $O(n^2)$ for $c = 8$ and $N = 33$.

4. Prove formally that function $k(n) = 10^n + n^2$ is $2^{O(n)}$.

Ans: We know that $10^n + n^2 < 10^n + 10^n = 2 \cdot 10^n \leq (2 \cdot 10)^n < 32^n = 2^{5n}$, for $n \geq 1$.

Thus, $k(n) = 10^n + n^2$ is $2^{O(n)}$ for $c = 5$ and $N = 1$.

5. Ans:

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

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 $O(|s|)$.

Induction problems

1. Ans:

Base case: $n = 0$, $1 + 2 \cdot 0 - 1 = 0 = 0^2$.

Assume that $n = k, P(k)$ holds, $1 + 3 + 5 + \cdots + 2k - 1 = k^2$.

Then $1 + 3 + 5 + \cdots + 2k - 1 + 2(k + 1) - 1 = k^2 + 2k + 1 = (k + 1)^2$, therefore $P(k + 1)$ holds.

To sum up, for all $n \geq 0$, $P(n)$ holds.

2. Ans:

Base case: $n = 0$, $0 = \frac{0(0+1)}{2}$.

Assume that $n = k$, $P(k)$ holds, $1 + 2 + 3 + \cdots + k = \frac{k(k+1)}{2}$.

Then, $1 + 2 + 3 + \cdots + k + (k + 1) = \frac{k(k+1)}{2} + (k + 1) = \frac{(k+1)(k+2)}{2}$, therefor $P(k + 1)$ holds.

To sum up, for all $n \geq 0$, $P(n)$ holds.

3. Ans:

Base case: $n = 1, 1^3 + 2 \cdot 1 = 3$ is divisible by 3.

Assume that $n = k$, $P(k)$ holds, $k^3 + 2k$ is divisible by 3.

Then for $P(k + 1)$: $(k + 1)^3 + 2(k + 1) = k^3 + 3k^2 + 3k + 1 + 2k + 2 = k^3 + 2k + 3(k^2 + k + 1)$.

Because $k^3 + 2k$ and $3(k^2 + k + 1)$ are both divisible by 3, $(k + 1)^3 + 2(k + 1)$ is also divisible by 3 as the addition of those two items. $P(k + 1)$ holds.

Thus, for all $n > 0$, $n^3 + 2n$ is divisible by 3.

4. Ans:

Base case: $n = 1$, $3^1 > 1^2$.

Assume that $n = k$, $P(k)$ holds: $3^k > k^2$.

Then for $P(k + 1)$: $3^{k+1} = 3^k + 3^k + 3^k > k^2 + k^2 + k^2 > k^2 + 2k + 1 = (k + 1)^2$.

Thus, $P(k + 1)$ holds and for any $n \geq 1$, $P(n)$ holds: $3^n > n^2$.

5. Ans:

Base case: $r.size = 1$. In this case, there is only one node in the tree, thus $r.left == r.right == null$.

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

Assume that $r.size = k > 1$, the function satisfies its specification, a.k.a it returns k.

Now we add one more node to top of the tree, making $r.size = k + 1$. The original tree becomes the left (or right) subtree of the new node.

From the source code, the return value is $size(new.left) + 1 = size(r) + 1 = k + 1$. Thus $r.size = k + 1$, the function still satisfies its specification.

To sum up, the function satisfies its specification for all $r.size \geq 1$.