Computer Science 331

Solutions to Selected Tutorial #5 Questions

Question 1

Use asymptotic notation to state bounds on the worst-case running times for the Bubble Sort and the gcd algorithms

We have computed the number of operations for the worst-case of the Bubble Sort algorithm in Tutorial 4, namely $T(n) = 5n^2 - 2$. We can prove the following claims about $T(n)$.

1. $T(n) \in O(n^2)$. To show this, note that

   $$5n^2 - 2 \leq 5n^2$$

   for all $n \geq 1$. Thus, the definition of big-O notation is satisfied for $c = 5$ and $N_0 = 1$.

2. $T(n) \in \Omega(n^2)$. To show this, note that

   $$5n^2 - 2 \geq 4n^2$$

   if $n^2 \geq 2$, or $n \geq \sqrt{2}$. Thus, the definition of Ω is satisfied for $c = 4$ and $N_0 = 2$.

3. $T(n) \in \Theta(n^2)$. Follows from the previous two claims.

Notice that the third statement, which says that the growth rate of $T(n)$ is the same as n^2, is as precise as we can be. Thus, results using little-o and ω are not required here.

For the gcd algorithm, we begin by determining the worst-case number of operations

```
                                    Cost
function gcd (a, b: integer)
   if (a==0) then                   1 operation
       return b
   elsif (b==0) then                1 operation
       return a
```

```
  else
      p = abs(a);                          1 operation
      q = abs(b);                          1 operation
      while (q <> 0) do                    1 operation
            r = p mod q;                   2 operation
            p = q;                         1 operation
            q = r;                         1 operation
       end do;
       return p                            1 operation
   end if
end function;
```

Since we are considering the worst case, $a \neq 0$ and $b \neq 0$, and the program enters into the while loop. To find how many times the while loop executes we have to find a loop variant for this loop. Following Question 2b of Tutorial 4, we define the size of the input to be the sums of the bit lengths of p and q and want to show that this function decreases by at least 1 after every iteration of the loop. The three cases indicated in Question 2b of Tutorial 4 cover all the possibilities for an iteration of the loop (case 2 involves showing that subtracting two numbers of equal bit length yields an answer with smaller bit length because the high-order bits will be one, case 2 follows because $r = p \bmod q$ is strictly less than q). Putting this together, we have that

$$f(p,q) = \begin{cases} \log_2 p + \log_2 q & \text{if } p \geq q \\ \log_2 p + \log_2 q + 1 & \text{if } p < q. \end{cases}$$

serves as a loop variant. The second case is required to ensure that the value of $f(p,q)$ decreases after the first iteration, because if $p < q$, then $r = p$ and we just swap p and q. The arguments from Question 2b show that $f(p,q)$ decreases by at least one after each iteration so, in the worst case where $\gcd(p,q) = 1$, we'll have $f(p,q) = 0$ when $p = 1$ and $q = 0$, implying that the loop terminates. As a result, the number of iterations (found by plugging in the initial values of p and q) will be at most

$$(\log_2 p) + (\log_2 q) + 1 \leq 2(\log_2 q) + 1$$

since the worst case occurs when $p < q$.

To count the worst-case number of operations, we note that the loop body costs 4 steps, the while-loop test costs 1 operation, and 5 steps are required for initialization before the loop and termination of the function. Thus, the total worst case number of operations $T(q)$ satisfies

$$\begin{aligned} T(q) &\leq 5 + (1)(2 + 2\log_2 q) + (4)(1 + 2\log_2 q) \\ &= 11 + 10\log_2 q \ . \end{aligned}$$

We can prove the following about $T(q)$.

1. $T(q) \in O(\log_2 q)$. To show this, note that
$$11 + 10\log_2 q \leq 11\log_2 q + 10\log_2 q = 21\log_2 q$$
if $\log_2 q \geq 1$, or $q \geq 2$. Thus, the definition of big-O is satisfied for $c = 21$ and $N_0 = 2$.
2. $T(q) \in \Omega(\log_2 q)$. To show this, note that
$$11 + 10\log_2 q \geq 10\log_2 q$$
if $\log_2 q \geq 0$, or $q \geq 1$. Thus, the definition of Ω is satisfied for $c = 10$ and $N_0 = 1$.
3. $T(q) \in \Theta(\log_2 q)$. Follows from the previous two claims.

Question 2d

To prove that $x \in o(x^2)$, we need to show that for every constant $c > 0$, there exists a constant $N_0 \geq 0$ such that $x \leq cx^2$. Note that we have $x \leq cx^2$ whenever $x \geq 1/c$, so for every constant c, the definition is satisfied with $N_0 = 1/c$.

Question 3

a: Prove that $2x^3 + 4 \in \Theta(x^3)$

First, note that
$$2x^3 + 4 \leq 2x^3 + 4x^3 = 6x^3$$
if $x^3 \geq 1$, or $x \geq 1$. Also, note that
$$2x^3 + 4 \geq 2x^3$$
also holds if $x \geq 1$. Thus, we have
$$0 \leq c_L x^3 \leq 2x^3 + 4 \leq c_U x^3 \quad \text{for } x \geq N_0$$
holds for $c_L = 2$, $c_U = 6$, and $N_0 = 1$, and by definition $2x^3 + 4 \in \Theta(x^3)$.

b: Prove that $2x^3 + 4 \notin O(x^2)$

Assume that $2x^3 + 4 \in O(x^2)$. By definition there exist constants $c > 0$ and $N_0 \geq 0$ such that
$$2x^3 + 4 \leq cx^2$$
for $x \geq N_0$. As $2x^3 < 2x^3 + 4$ for $x \geq 0$ we have
$$2x^3 \leq cx^2 \ .$$
Dividing both sides by x^2 yields
$$2x \leq c$$
for all $x \geq N_0$, a contradiction. Thus, our assumption that $2x^3 + 4 \in O(x^2)$ must be false.

c: Prove that $2x^2 \notin \omega(x^2)$

Assume that $2x^2 \in \omega(x^2)$. By definition, for every constant $c > 0$, there exists a constant $N_0 \geq 0$ such that

$$2x^2 \geq cx^2$$

for all $x \geq N_0$. Dividing both sides by x^2 implies that for every constant $c > 0$ we have

$$2 \geq c \ ,$$

a contradiction. Thus, our assumption that $2x^2 \in \omega(x^2)$ must be false.

Question 6.a

If $f(n) \in \Theta(g(n))$, then by definition there exist constants $c_L, c_U > 0$ and $N_0 \geq 0$ such that $0 \leq c_L g(n) \leq f(n) \leq c_U g(n)$ for all $n \geq N_0$. To show that $g(n) \in \Theta(f(n))$ we need to show that there exist constants $c'_L, c'_U > 0$ and $N'_0 \geq 0$ such that $0 \leq c'_L f(n) \leq g(n) \leq c'_U f(n)$ for all $n \geq N_0$.

- By assumption we have that $f(n) \leq c_U g(n)$ for all $n \geq N_0$. Thus, $g(n) \geq (1/c_U) f(n)$ for all $n \geq N_0$.
- By assumption we have that $f(n) \geq c_L g(n)$ for all $n \geq N_0$. Thus, $g(n) \leq (1/c_L) f(n)$ for all $n \geq N_0$.

Hence, we can take $c'_L = 1/c_U$, $c'_U = 1/c_L$, and $N'_0 = N_0$, and by definition $g(n) \in \Theta(f(n))$.