COT 3100 HW5 solution

Question 1  [Section 3.1 Exercise 12, p.202 (10 points)]

Four assignment statements are needed, one for each of the variables and a temporary assignment to get started so that we do not lose one of the original values.

\[
\begin{align*}
temp &:= x \\
x &:= y \\
y &:= z \\
z &:= temp
\end{align*}
\]

Question 2  [Section 3.1 Exercise 32, p.203 (10 points)]

The following algorithm will find all terms of a finite sequence of integers that are greater than the sum of all the previous terms. We put them into a list \( L \), but one could just as easily have them printed out, if that were desired. It might be more useful to put the indices of these terms into \( L \), rather than the terms themselves(i.e. their values), but we take the former approach for variety. As usual, the empty list is considered to have sum 0, so the first term in the sequence is included in \( L \) if and only if it is positive.

**procedure** find all biggies \((a_1, a_2, ..., a_n : \text{integers})\)

set \( L \) to be the empty list

\( sum := 0 \)

\( i := 1 \)

while \( i \leq n \)

if \( a_i > sum \) then append \( a_i \) to \( L \)

\( sum := sum + a_i \)

\( i := i + 1 \)

**return** \( L \) \{ the list of all the values that exceed the sum of all the previous terms in the sequence \}

Question 3  [Section 3.1 Exercise 34, 38, p.203 (10 points)]

(a) There are five passes through the list. After one pass the list reads 2, 3, 1, 5, 4, 6, since the 6 is compared and moved at each stage. During the next pass, the 2 and the 3 are not interchanged, but the 3 and the 1 are, as are the 5 and the 4, yielding 2, 1, 3, 4, 5, 6. On the third pass, the 2 and the 1 are interchanged, yielding 1, 2, 3, 4, 5, 6. There are two more passes, but no further interchanges are made, since the list is now in order.

(b) We start with 6, 2, 3, 1, 5, 4. The first step inserts 2 correctly into the sorted list 6, producing 2, 6, 3, 1, 5, 4. Next 3 is inserted into 2, 6, and the list reads 2, 3, 6, 1, 5, 4. Next 1 is inserted into 2, 3, 6, and the list reads 1, 2, 3, 6, 5, 4. Next 5 is inserted into 1, 2, 3, 6, and the list reads 1, 2, 3, 5, 6, 4. Finally 4 is inserted into 1, 2, 3, 5, 6, and the list reads 1, 2, 3, 4, 5, 6. At each insertion, the element to be inserted is compared with the elements already sorted, starting from the beginning, until its correct spot is found, and then the previously sorted elements beyond that spot are each moved one position toward the back of the list.

Question 4  [Section 3.2 Exercise 22, p.216 (10 points)]

The ordering is straightforward when we remember that exponential grow faster than polynomial functions, that factorial functions grow faster still, and that logarithmic functions grow very slowly. The order is \((\log n)^3, \sqrt{n \log n}, n^{99} + n^{98}, n^{100}, 1.5^n, 10^n, (n!)^2\).

Question 5  [Section 3.2 Exercise 26, p.217 (10 points)]

The approach in these problems is to pick out the most rapidly growing term in each sum and discard the rest(including the multiplication constants).

a) This is \( O(n^3 \cdot \log n + \log n \cdot n^3) \), which is the same as \( O(n^3 \cdot \log n) \).

b) Since \( 2^n \) dominates \( n^2 \), and \( 3^n \) dominates \( n^3 \), this is \( O(2^n \cdot 3^n) = O(6^n) \).

c) The dominant terms in the two factors are \( n^n \) and \( n! \), respectively. Therefore this is \( O(n^n \cdot n!) \).

Question 6  [Section 3.2 Exercise 44, p.217 (10 points)]

The definition of "\( f(x) \) is \( \Theta(g(x)) \)" is that \( f(x) \) is both \( O(g(x)) \) and \( \Omega(g(x)) \). That means that there are positive constants \( C_1, k_1, C_2 \) and \( k_2 \) such that \( |f(x)| \leq C_2 |g(x)| \) for all \( x > k_2 \) and \( |f(x)| \geq C_1 |g(x)| \) for all \( x > k_1 \). Similarly, we have that there are positive constants \( C_1', k_1', C_2' \) and \( k_2' \) such that \( |g(x)| \leq C_2' |h(x)| \) for all \( x > k_2 \) and \( |g(x)| \geq C_1' |h(x)| \) for all \( x > k_1' \). We can combine these inequalities to obtain \( |f(x)| \leq C_2 C_2' |h(x)| \) for all \( x > \max(k_2, k_2') \) and \( |f(x)| \geq C_1 C_1' |h(x)| \) for all \( x > \max(k_1, k_1') \). This means that \( f(x) \) is \( \Theta(h(x)) \).
**Question 7**  [Section 3.3 Exercise 14, p.230 (10 points)]

a) Initially \( y := 3 \). For \( i = 1 \) we set \( y \) to \( 3 \cdot 2 + 1 = 7 \). For \( i = 2 \) we set \( y \) to \( 7 \cdot 2 + 1 = 15 \), and we are done.

b) There is one multiplication and one addition for each of the \( n \) passes through the loop, so there are \( n \) multiplications and \( n \) additions in all.

**Question 8**  [Section 3.3 Exercise 39, p.231 (10 points)]

a) doubles  
b) increases by 1

**Question 9**  [Section 3.3 Exercise 44, p.231 (10 points)]

We have two choices: \((AB)C\) or \(A(BC)\). For the first choice, it takes \(3 \times 9 \times 4 = 108\) multiplications to form the \(3 \times 4\) matrix \(AB\), and then \(3 \times 4 \times 2 = 24\) multiplications to get the final answer, for a total of \(168\) multiplications. For the second choice, it takes \(9 \times 4 \times 2 = 72\) multiplications to form the \(9 \times 2\) matrix \(BC\), and then \(3 \times 9 \times 2 = 54\) multiplications to get the final answer, for a total of \(126\) multiplications. The second method uses fewer multiplications and so is the better choice.

**Question 10**  [10 points]

a) When the smallest element is in the last position, it is the best case, with complexity \(O(n^2)\). (Of course, an array in reversely sorted order is a worst case.) When the original data is already-sorted in increasing order, it is the best case, with complexity \(O(n)\).

b) When the original data is already-sorted in decreasing order, it will be the worst case for bubble sort algorithm as stated in textbook, with complexity \(O(n^2)\). When the original data is already-sorted in increasing order, it is the best case, with complexity \(O(n)\).

**Bonus Question**  (1 extra point)

a) False. For example, \(f(n) = n\), \(g(n) = n^2\), \(f(n) = O(g(n))\), but \(g(n) \neq O(f(n))\).

b) False. For example, \(f(n) = n\), \(g(n) = n^2\), \(f(n) = O(g(n))\), but \(g(n) = \Theta(n^2) = \Theta(max(n, n^2))\)

c) True. Because \(f(n) = O(g(n))\), there exist positive constants \(C, k\) such that \(f(x) \leq Cg(x)\) for all \(x > k\).

\[
\begin{align*}
  f(x) &\leq Cg(x) \\
  \lg(f(x)) &\leq \lg(Cg(x)) \\
  \lg(f(x)) &\leq \lg C + \lg g(x) \quad \text{(because } \lg g(x) \geq 1) \\
  \lg(f(x)) &\leq (\lg C + 1)\lg g(x) \quad \text{(} C' = \lg C + 1\text{)} \\
  \lg(f(x)) &\leq C' \lg g(x) \quad \text{thus, } \lg(f(x)) = O(\lg(g(x)))
\end{align*}
\]

d) False. For example, \(f(n) = 2n\), \(g(n) = n\). \(4^n = \Omega(2^n)\), \(2^{O(f(x))} = \Omega(2^{g(x)})\)

e) False. For example, \(f(n) = \frac{1}{n}\), \((f(n))^2 = \frac{1}{n^2}\), but \(\frac{1}{n} \neq O(\frac{1}{n^2})\).

f) True. By definition.

g) False. For example, \(f(n) = 2^n\), \(f(n/2) = (\sqrt{2})^n\), \(f(n) = \Omega(f(n/2))\) but \(f(n) \neq O(f(n/2))\)