3 3 3 1

1 1 1

12

UNIVERSITY OF ALBERTA
CMPUT 204

8 + 4

Term Test 1 (Solutions), October 16, 2019

Question 1. [10 MARKS]

Part (a) [6 MARKS]

8/1/1/3 :

27

Consider the following code:

9

12 +

3

1

```
procedure Test(n)
y ← 1
for (i from 1 to n)
    y ← y × 2
return y
```

(i) Give a loop invariant (LI) for the loop in the code.

Solution. LI: At the beginning of each iteration i $(1 \leq i \leq n+1)$, $y = 2^{i-1}$.

(ii) Prove the maintenance, i.e., prove that each iteration of the loop maintains the LI. State clearly what is assumed, what is to be proved, and how you prove it.

Solution. [3 marks] For any $1 \leq i \leq n$, assume LI holds, i.e., $y = 2^{i-1}$, at the beginning of the iteration (I.H.) and we show that $y = 2^{i'-1}$, where $i' = i+1$, at the beginning of the next iteration. By I.H., $y = 2^{i-1}$, and by the effect of the code $y \leftarrow y \times 2$, we get $y = 2^{i-1} \times 2 = 2^i$. At the beginning of the next iteration, i is incremented by 1, i.e., $i' = i + 1$. It follows $y = 2^{i'-1}$.

Part (b) [4 MARKS]

- Complete the following definition. A function $g(n) \in \Omega(f(n))$ if ...

  Solution: $\exists c > 0, n_0 \in \mathbb{N}$, such that for all $n \geq n_0$ we have $g(n) \geq cf(n)$.

- Prove that $g(n) \in \Omega(f(n))$, where $g(n) = 8n^2 + 5n + 7$ and $f(n) = \frac{n^2}{5}$.

  Solution: $c = 5$, $n_0 = 0$, for all $n \geq 0$, $8n^2 + 5n + 7 \geq 5 \times \frac{n^2}{5} = n^2$, i.e. $7n^2 + 5n + 7 \geq 0$, thus for all $n \geq 0$, $g(n) \geq 5f(n)$. $g(n) \in \Omega(f(n))$.

Question 2. [12 MARKS]

Part (a) [4 MARKS]

Consider array $A = [6, 2, 4, 7, 1, 3, 5]$ which has size 7. Recall the procedure *Partition* in the Quick-Sort algorithm with pivot being the last key. Show the result of calling Partition$(A, 1, 7)$. (It is sufficient to show the resulting array; you do not need to show how you derive it.)

Solution: [2,4,1,3,5,7,6]

Part (b) [4 MARKS]

Suppose that Priority Queues are implemented by max-heap. Consider array $A = [9, 7, 5, 6, 2, 3]$, which is a max-heap. Show the resulting array after the following two calls in the order in which they are called:

`Extract-Maximum(`A`)` **** return the max element of A and extract it from A
`Insert(`$A, 4$`)` **** insert new key 4 into A

Show your work.

Solution: After Extract-Maximum, we get a max-heap in $A = [7, 6, 5, 3, 2]$ (They can answer $A = [7, 6, 5, 3, 2, 3]$ with heap-size $= 5$). After Insert, we get $A = [7, 6, 5, 3, 2, 4]$. If they get the final answer right, you can ignore "show your work", which can only be useful for partial marks.

Part (c) [4 MARKS]

For input lists that are already sorted, for each sorting algorithm below, indicate its running time in terms of the big-O notation. Your bound should be tight.

- InsertionSort: $O(n)$
- MergeSort: $O(n \log n)$
- HeapSort: $O(n \log n)$
- QuickSort: $O(n^2)$

Question 3. [18 MARKS]

Part (a) [7 MARKS]

Indicate (without proof) which of the following statements are true and which are false.

(T) $a)$ $(2n^2 \log^4 n) \in O(n^{2.1})$	(T) $d)$ $(n+1)! \in O(n^n)$
(T) $b)$ $1000 + \frac{7}{\log n} \in O(1)$	(F) $e)$ $\sqrt{n} + \frac{n}{\log n} \in \Theta(n)$
(F) $c)$ $\log(n^3) \in \Omega((\log n)^2)$	(T) $f)$ $(\log \log n)^{\log \log n} \in O(\sqrt{n})$

Part (b) [11 MARKS]

Find asymptotic tight bounds (i.e. Θ) for the following recurrences. Assume that in each case, $T(n)$ is some positive constant for sufficiently small n. If you use the master theorem, indicate which case applies. If you use iterated substitution or a recurrence tree to obtain your guess, show your work. But you do *not* need to prove that your closed-form is correct.

a) $T(n) = 9T(\frac{n}{3}) + 2n \log n$.

Answer: $\Theta(n^2)$, apply case 1 of Master Theorem: $n^{\log_3 9} = n^2$ and $2n \log n \in o(n^{2-\epsilon})$

b) $T(n) = 15T(\frac{n}{4}) + n^2\sqrt{\log n}$.

Answer: $\Theta(n^2\sqrt{\log n})$, apply case 3 of MT: $n^{log_4 15} < n^2$, and further verify $\exists \delta$, s.t. $15(\frac{n}{4})^2\sqrt{\log \frac{n}{4}} = \frac{15}{16}n^2\sqrt{\log \frac{n}{4}} < \delta n^2 \sqrt{\log n}$

c) $T(n) = T(n-2) + n\log n$.

Solution for c: Using iterated substitution, we have

$$\begin{aligned}
T(n) &= T(n-2) + n\log n \\
&= T(n-4) + (n-2)\log(n-2) + n\log n \\
&\vdots \\
&= T(0) + \sum_{k=1}^{\frac{n}{2}} 2k\log(2k) \\
&\le T(0) + \frac{n^2}{2}\log n \\
&\in O(n^2\log n)
\end{aligned}$$

On the other hand, we have

$$\begin{aligned}
T(n) &= T(0) + \sum_{k=1}^{\frac{n}{4}} 2k\log(2k) + \sum_{k=\frac{n}{4}+1}^{\frac{n}{2}} 2k\log(2k) \\
&\ge T(0) + \sum_{k=\frac{n}{4}+1}^{\frac{n}{2}} 2k\log(2k) \\
&\ge T(0) + \frac{n^2}{8}\log\frac{n}{2} \\
&\in \Omega(n^2\log n)
\end{aligned}$$

So $T(n) \in \theta(n^2\log n)$.

Question 4. [10 MARKS]

An array A is called *even-odd* if (i) A has at least one odd number and (ii) any odd number in A appears after all even numbers of A. E.g., $A = [6, 8, 4, 2, 3, 1]$ is an even-odd array. Note that if A has no even numbers, then the first odd number is $A[1]$, e.g., $A = [3, 1, 5]$ is a valid input. Give an efficient algorithm with running time $O(\log n)$ to determine the index of the first odd number of an even-odd array. For the first example above, index 5 should be returned.

You should describe your algorithm in pseudocode. Briefly justify the running time.

Solution: Two possible implementations of *even-odd*

```
Procedure even-odd(A,p,r)
       **** Pre-condition: A has at least one odd number and odd numbers
                           in it appear after all even numbers, if there is any.
   while (p < r) do
       m = floor[(r-p)/2] + 1
       if A[m] is odd
              then r = m
       else p = m+1
   return p     ***at the end, p=r
```

```
Procedure even-odd(A,l,r)   *** initially l = 1, r = n (length of A)
    *** base case
    if l == r
        then return l    *** l is index of first odd
    mid = floor[(l + r) / 2]
    if A[mid] is even
        *** search in the upper half of array
        then return even-odd(A, mid + 1, r)
    else    *** A[mid] is odd
        *** search in the lower half of array
        then return even-odd(A, l, mid)
```

The recurrence for the second implementation is $T(n) = T(n/2) + C$. By case 2 of the Master Theorem, $T(n) \in \Theta(\log n)$.

Total Marks = 50