Comp 2140 - Data Structures

Midterm

Shahin Kamalli
University of Manitoba

March 4th, 2019

Write your name and student id here:

‘It Isn’t the Mountain Ahead That Wears You Out; It Is the Grain of Sand in Your Shoes.’ Robert W. Service

Do not open this booklet until instructed.

• You are not allowed to use any printed/written material, laptops/cell-phones. Please turn off your cell phones and put them in your bags.
• Manage your time. We start the exam at 9:30 and end the exam at 10:20. You have 50 minutes.
• Questions are NOT ordered by their difficulty. If you are stuck in one question, skip it and go to the next.
• There are 6 pages (including this cover page). Write your answers in the provided boxes.
• In the unlikely case that you find the exam too long/hard, do not panic. The marks will be scaled so that the highest mark gets the full mark.
1. Short Answer (14+5+5=24 marks)

Provide your short answers in the provided boxes.
There is no need to justify your answers for the True/False questions.

1. True or False: An interface can implement an abstract class.
   **Answer:** False.
   An abstract can implement an interface but not the other way around.

2. True or False: A protected method of class A can be called in classes inherited from A.
   **Answer:** True;
   the definition of being “protected” implies that the descendants of a class can access its protected methods and variables.

3. True or False: $2019n \log n \in O(n)$
   **Answer:** False.
   $n \log n$ grows faster than $n$.

4. True or False: $n^2 \log n \in O(n^3)$
   **Answer:** True.
   $n^2 \log n$ grows slower than $n^3$.

5. True or False: merge-sort runs in $O(n \log n)$ in the best case
   **Answer:** True;
   we learned in the class that the time complexity of merge-sort stays the same in the best and worst case.

6. Consider the following figure which plots three functions $f$ and $g$.
   True or False: $f(n) \in O(g(n))$.
   **Answer:** False;
   $f(n)$ is a non-linear function which grows faster than $f$. So, we have $g(n) \in O(f(n))$ but not the other way around.

7. Consider the following pseudocode:
   ```c
   void foo ( int n)
   {
     int i = 1;
     int q = 1;
     while (i < n)
     {
       for (int w=1; w < 1398; w++)
         for (int j=1; j < 2019; j++)
           q = q*j;
       i = i+2;
     }
   }
   ```
What is the worst-case running time of \( \text{foo}(n) \)? Express your answer using \( O \)-notation in terms of \( n \), and be as precise as possible. No justification is needed.

\[
O(n)
\]

**Answer:** The answer is \( O(n) \). The while loop iterates \( O(n) \) time and the other loops iterate constant times (the constant is \( 1398 \times 2019 \)).

8. Consider the following recursive algorithm.

```c
int foo(int n)
{
    if(n <= 0)
        return 0;
    else
    {
        int A = foo(n - 1);
        return n*n + A;
    }
}
```

a) In one or two sentences indicate what the algorithm does.

**Answer:** It returns sums of squares on the first \( n \) numbers, i.e., the value of \( n^2 + (n-1)^2 + (n-2)^2 + \ldots + 9 + 4 + 1 \).

b) Indicate whether the code uses a tail recursion or not. In either case, shortly explain why.

**Answer:** It is not a tail recursion because a computation is made after the recursive call (in the last line).

9. Assume you want to sort a set of non-negative real numbers whose fractional part is 0, 0.1, or 0.2. An example is \{0.0, 2.1, 3.2, 0.1, 5.2, 8.0, 9.1\}. Assume all numbers are smaller than \( \log n \). Indicate whether it is possible to sort these numbers in \( O(n) \) or not. Justify your answer in a few sentences (be precise).

**Answer:** Yes, it is possible. Multiply all numbers by 10. Now, we have \( n \) non-negative numbers in the range \( [0, 10 \log n] \). Use counting sort to sort the updated numbers; counting sort takes \( O(n+10 \log n) = O(n) \). After sorting the numbers, divide them by 10.
2. Recursion (7 marks)

For a given integer \( n \) such than \( n \geq 2 \), let’s define \( OF(n) \) as the product of all positive, odd integers strictly smaller than \( n \). For example, \( OF(2) = 1 \), \( OF(3) = 1 \), \( OF(4) = 1 \times 3 \), \( OF(5) = 1 \times 3 \), and \( OF(10) = 1 \times 3 \times 5 \times 7 \times 9 \).

Write a recursive code to compute and return \( OF(n) \). You need to complete the following code:

```java
public static int OF(int n) {
    if (n < 2)
        return -1;
    if (n == 3)
        return 1;
    // complete the code in the space below
    if (n == 2)
        return 1;
    if (n % 2 == 0)
        return OF(n + 1);
    return (n - 2) * OF(n - 2);
}
```

**Answer:** Note that there are many, many ways to correctly answer this question. The above is only one of the many.
3. More Recursion (9 marks)

We want to write a code that **recursively** reverses the odd numbers in an array of positive integers. The position of even numbers does not change.

For example, an array 6, 1, 2, 4, 5, 3, 7, 12, 9, 10, 18 will become 6, 9, 2, 4, 7, 3, 5, 12, 1, 10, 18. Note that the odd numbers (highlighted) are reversed while the even number are not changed.

You should complete the following code:

```java
public static void reverse_odd(int[] A) {
    reverse_odd_rec(A, 0, A.length -1);
}

private static void reverse_odd_rec(int[] A, int lo, int hi) {
    // implement this. You should recognise what lo and hi are
    while (lo < A.length && A[lo] % 2 == 0) 
        lo ++;
    while (hi >= 0 && A[hi] % 2 == 0)
        hi --;
    if (lo < hi)
        {
            int temp = A[lo];
            A[lo] = A[hi];
            A[hi] = temp;
            reverse_odd_rec(A, lo+1, hi-1);
        }
}
```
4. Sorting (3+5+4=12 marks)

Consider the quick sort algorithm for sorting \( n \) integers. Assume you are given a pivot selection algorithm that selects a pivot \( x \) such that at least \( n/3 \) numbers are guaranteed to be smaller than \( x \) and at least \( n/4 \) numbers are guaranteed to be larger than \( x \).

a) write a recursive function for the best-case time complexity of the quick-sort with this pivot-selection algorithm.

**Answer:** In the best case, the pivot is always in the middle. Note that the middle item is among the possible pivots for the pivot selection algorithm. The recursion is similar to the best-case of any other pivot-selection algorithm. We have \( T(n) = 2T(n/2) + cn \) for some constant \( c \) and \( T(1) = d \) for some constant \( d \).

b) write a recursive function for the worst-case time complexity of the quick-sort with this pivot-selection algorithm.

**Answer:** Recall that in the worst-case, the pivot takes a value so that the size of the recursions are as un-balanced as possible. In this case, the worst case is when the pivot is always at \( 3n/4 \). The recursion will be \( T(n) = T(3n/4) + T(n/4) + cn \), \( T(1) = d \) for some constant values of \( c \) and \( d \).

c) (bonus) What is the time complexity of the algorithm in the worst case? To get the complete mark, you need to justify your answer. It is not sufficient to provide a correct time without a correct justification.

**Answer:** For solving recursions likes this, we can use a recursion tree method. At the \( i \)'th level of the tree, we have the results of recursion when we have replaced its components (initially \( T(n/4) \) and \( T(3n/4) \) for \( i \) times. For example, at level 0 we have \( T(n) \), at level 1 we have \( T(n/4) + T(3n/4) \) and so on. Since the recursive parameter decrease by at least a factor of \( 3/4 \) after each replacement, after \( \log_{4/3} n \) replacements, we get a constant value. That means the height of the tree is at most \( \log_{4/3} n \). The value of \( T(n) \) can be described as the total cost for all levels. In this example, the cost for each level is \( cn \) (see the figure below). So, we have \( T(n) < \log_{4/3} n \times cn = O(n \log n) \).