National University of Singapore

School of Computing

CS2040S - Data Structures and Algorithms

Midterm Test

(Semester 1 AY2024/25)

Time Allowed: 70 minutes

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this assessment paper until you are told to do so.
2. This assessment paper contains ONE (1) section with a few short questions.
   It comprises TEN (10) printed pages, including this page.
3. This is an **Open Book Assessment**.
   Only non-programmable calculator is allowed in this assessment.
4. Answer **ALL** questions within the **boxed space** of the answer sheet (page 07-10).
   There are a few starred (*) boxes: free 1 mark if left blank but 0 for wrong answer (no partial).
   The answer sheet is at page 07-10, just hand in those pages.
   You can use either pen or pencil. Just make sure that you write **legibly**!
5. Important tips: Pace yourself! Do **not** spend too much time on one (hard) question.
   Read all the questions first! Some (subtask) questions might be easier than they appear.
6. You can use **pseudo-code** in your answer but beware of penalty marks for **ambiguous answer**.
   You can use **standard, non-modified** classic algorithm in your answer by just mentioning its name, e.g., run Merge Sort on ArrayList A, etc.
7. The total marks is 70. All the best :)

A Short Questions (70 marks)

A.1 Java Code Comprehension and Analysis (15 marks)

You are given the following Java code that correctly solves one programming contest problem in Kattis:

```
import java.util.*;

public class A1 {
  public static void main(String[] args) throws Exception {
    Scanner sc = new Scanner(System.in);

    // input
    int n = sc.nextInt(), m = sc.nextInt(); sc.nextLine();
    ArrayList<Integer> A = new ArrayList<>();
    for (int i = 0; i < n; ++i)
      A.add(sc.nextInt());

    // the computation
    int ans = 0;
    for (int i = 0; i <= n-m; ++i) {
      int something = 0;
      for (int j = 0; j < m; ++j)
        if (A.get(i+j)%2 == 0)
          ++something;
      if (something >= 2)
        ++ans;
    }

    // output
    System.out.println(ans); // what is actually computed?
  }
}
```

1. (2 marks) If we enter `8 3` as the first line of input and then
   `1 2 3 4 5 6 7 8` as the second line of input,
   what will this Java code produce?

2. (2 marks) If we enter `9 4` as the first line of input and then
   `1 2 3 5 7 8 9 11 5` as the second line of input,
   what will this Java code produce?

3. (5 marks) Describe, in high-level, what is being computed in the "the computation" section? Any logical explanation will be accepted.

4. (3 marks) What is the tightest Big O time complexity of this solution, in terms of n and m? PS: You are given additional information that $1 \leq m < n$.

5. (3* marks) ~~Can you~~ Please rewrite the Java code so that it solves the same problem, but in a faster (i.e., tighter) Big O time complexity *(the time complexity should **not** involve variable m).*

A.2 Big-O Time Complexity Analysis (10 marks)

There is an unknown algorithm that process a Java ArrayList A (containing N integers). This algorithm has been correctly analyzed to have time complexity of $O(N \log N)$. There is no randomized component in this unknown algorithm.

For the five statements below, which statement(s) is/are **always True for all cases|can be False on at least one case** given the information above (1 mark per correct decision)? Give a short one sentence explanation for your decision (1 mark for each logical explanation).

1. If $N = 2^{17} = 131\,072$, then the total number of operations is **exactly** $2\,228\,224$ operations.
2. This algorithm can be improved to $O(N)$.
3. We can also say that this algorithm runs in $O(N^2)$.
4. If we change ArrayList A into PriorityQueue PQ, the algorithm will run in $O(N)$.
5. *Asymptotically, as N becomes very large,* there is no test-case can make this algorithm executes N^2 operations or more.

A.3 Another Merge Sort Variant (5 marks)

In class, you learn about the following standard Merge Sort algorithm:

```
method mergeSort(array A, integer low, integer high)
  // the array to be sorted is A[low..high]
  if (low < high) // base case: low >= high (0 or 1 item)
    int mid = (low+high) / 2
    mergeSort(a, low   , mid ) // divide into two halves
    mergeSort(a, mid+1, high) // then recursively sort them
    merge(a, low, mid, high) // conquer: the O(n) merge subroutine
```

Suppose, you modify it into the following

```
method mergeSort(array A, integer low, integer high)
  // the array to be sorted is A[low..high]
  if (low < high) // base case: low >= high (0 or 1 item)
    int mid = (low+high) / 2
    mergeSort(a, low   , mid ) // divide into two halves
    mergeSort(a, mid+1, high) // then recursively sort them
    call an O(n log n) sort to sort A[low..high] //<-- the change
         // can be anything, perhaps call Arrays.sort(A, low, high)
         // that only sort this subarray A[low..high]
```

1. (3 marks) Just choose one of the following (2 marks):
   This modification of Merge Sort is [faster|no different|slower]
   than the original version of Merge Sort discussed in class.
   Give a short one sentence explanation for your decision (1 mark)

2. (2* mark) What is the time complexity of this modification of Merge Sort?

A.4 Analyze These Statements (20 marks)

For each of the five statements below, choose whether it is **correct**|**incorrect** (1 mark). If you say it is correct, write a short one sentence explanation for your decision. If you say it is incorrect, just show one counterexample (the other 1 mark).

1. This Java code runs in worst-case time complexity of $O(n^2)$ and this analysis is tight.

```
int counter = 0;
for (int i = n; i >= 1; --i)
  for (int j = 1; j <= n/i; ++j)
    ++counter;
```

2. (Randomized) Quick sort is the *best* sorting algorithm to sort *any* set of n Integers.

3. Java `Collections.sort` *can* run *slower* than $O(n \log n)$ (and more than $O(n^2)$) if we use it to sort a collection of n Java Strings (assume each String has m Characters).

4. We *can* compute the number of inversions (a.k.a. number of Bubble Sort swaps) of an ArrayList A with n integers faster than $O(n^2)$.

5. Singly Linked List discussed in Lecture 4a+4b is a *better* data structure to implement the Stack ADT `push(key)` and `last_key = pop()` than using Java ArrayList, i.e., a resizeable array.

6. Doubly Linked List discussed in Lecture 4a+4b is a *better* data structure to implement the Deque ADT `offerFirst(key)`, `offerLast(key)`, `key1 = pollFirst()`, and `key2 = pollLast()` than using Java ArrayList, i.e., a resizeable array. Important: For this question, the maximum number of elements in the Deque is already known beforehand.

7. A Stack/Queue is a Last-In-First-Out/First-In-First-Out data structure, respectively. Thus, there is *no way* we can insert a sequence of n ($n > 1$) Integers into a Stack and (the same sequence of n Integers into) a Queue and when we peek the top/front of the Stack/Queue, respectively, we see the same Integer (value).

8. Suppose that you have two Linked Lists La and Lb. La/Lb contains n/m unsorted alphabets ['A'..'Z'], respectively ($100 < n, m < 10\,000; n \neq m$). As we cannot get an element at index i in $O(1)$ if we use Linked List, to check whether an alphabet is inside *both* La and Lb, we need an $O(n \times m)$ algorithm that is roughly like this:

```
for (Character a:  La)
..for (Character b:  Lb)
....if (a.equals(b)) return true;
return false;
```

9. The smallest element in a Binary *Max* Heap is always located at the bottommost, rightmost leaf, i.e., at index n of a compact array A that describes the Binary *Max* Heap.

10. Suppose that we need to use a *special kind* of ADT Priority Queue where all enqueue (Insert(v)) operations will be performed first *before* all subsequent dequeue (ExtractMax()) operations (from highest priority to lowest priority). For this kind of ADT PQ, we *can* use another data structure and/or algorithm to achieve similar time complexities as if we use Binary (Max) Heap.

A.5 An Algorithm that uses Priority Queue (15 marks)

You are not yet told what this algorithm does, but it is from a real algorithm that you will eventually learn in a future algorithm course (if there is no change in curriculum).

```
import java.util.*;

public class A5 {
  public static void main(String[] args) {
    int n = 5;
    int[] charFreq = { 5, 1, 6, 10, 3 };

    ArrayList<Integer> ds = new ArrayList<>();
    for (int i = 0; i < n; ++i)
      ds.add(charFreq[i]);

    while (ds.size() > 1) {
      Collections.sort(ds);
      Integer x = ds.get(0);
      ds.remove(0);
      Integer y = ds.get(0);
      ds.remove(0);
      ds.add(x+y);
      System.out.println(x.toString() + "+" + y.toString() + "=" + (x+y));
    }
  }
}
```

1. (4 marks) Output 4 lines that will be the output of that code, if executed.

2. (3 marks) If we change $n = 7$ and $charFreq = \{1, 2, 3, 4, 5, 6, 7\}$
   What will be the *last line* of the output?

3. (3 marks) What is the time complexity of that code in terms of n?
   You need to assume that n can be a very big number, not just $n = 5$ in this example.

4. (5* marks) ~~Can you~~ Please rewrite the Java code so that it solves the same problem, but in a faster (i.e., tighter) Big O time complexity. Hint: Do not call `Collections.sort` repeatedly. There is a better data structure for this problem.

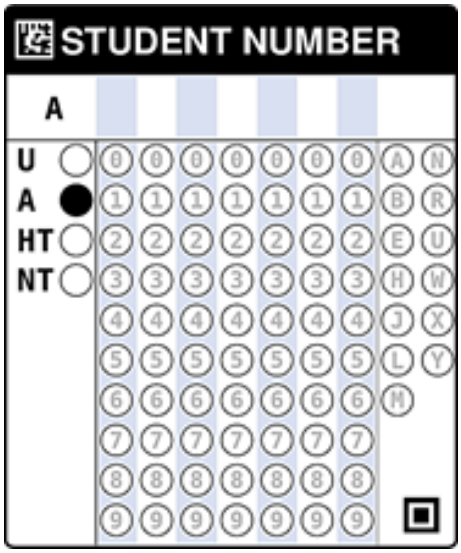
A.6 The Last Question (5 marks)

To qualify for up to easy 5 marks, you need to **write both full names correctly**.
My CS2040S lecturer is _____ and Teaching Assistant (TA) is _____,

Write a **short** (maybe limit yourself to around 2 minutes to do this and about 3-4 sentences) **but honest (and not anonymous)** feedback on what you have experienced in the first 6 weeks of CS2040S in Semester 1 AY 2024/25 (including Week -02/-01 experience, if any). Feedback that are shared by *majority* (**not a one-off**) and can be easily incorporated (e.g., Prof Halim, do not travel again on Week 09 like Week 04 and 06 is very hard to change) to make the next 7 weeks of CS2040S better will be done. Grading scheme: 0-blank, 3-considered trivial feedback but not blank, 5-good and constructive feedback, thanks. (Penalty -1 mark for each wrong name above...).

The Answer Sheet

Write your Student Number in the box below using **(2B) pencil**.
Do NOT write your name.



Box A.1.1 Check Your Understanding 1 (write one integer)

Box A.1.2 Check Your Understanding 2 (write one integer)

Box A.1.3 Explain that Java code in high-level

Box A.1.4 What is the time complexity in terms of n and m?

Box A.1.5* (1 if blank, 0 if wrong) Design a faster algorithm, regardless of m

Box A.2 always True|can be False; and why? (5 statements)

Box A.3.1. faster|no different|slower; and why?

Box A.3.2* (1 if blank, 0 if wrong) What is the time complexity?

Box A.4 1st-5th boxes, correct|incorrect; and why? (first 5 statements)

Box A.4. 6th-10th boxes, correct|incorrect; and why? (next 5 statements)

Box A.5.1 The output of the given code

Box A.5.2 The output of the next test case

Box A.5.3 The time complexity in terms of n

Box A.5.4* (1 if blank, 0 if wrong) Design a faster algorithm that does not sort each time

Box A.6 The Last Question: Lecturer and TA name, plus honest feedback

– END OF PAPER; All the Best –