CS2040S Semester 1 2023/2024

Data Structures and Algorithms

Tutorial+Lab 01

Basic Java, Basic OOP, Analysis, Hands-on 1

For Week 03

Document is last modified on: August 21, 2023

MODAL ANSWER IS FOR OUR CLASS ONLY; NOT TO BE DISTRIBUTED IN PUBLIC

1 Introduction and Objective

The purpose of this first tutorial+lab session is to recap the first four lecture sessions of CS2040S: Introduction, basic Java (especially Java ArrayList and basic OOP: Java class), basic analysis of algorithm, and to ensure that all students can code a simple Java program using their own computer at home (and/or laptop that he/she intends to use for Week 11 F2F Practical Exam). The first half of the session is generally the 'tutorial' part and the second half of the session is generally the 'hands-on/lab' part. The tutors will control the timings and they don't have to divide the sessions exactly by half. There will be a short break during the transition.

As this is the first session, we will do a quick ice breaking at the start of the session. Your TA is your main contact person of CS2040S related queries this semester. Only contact Prof Halim for questions that you are sure that your TA cannot (or has no privilege to) answer, e.g., questions about class policy, appeal for plagiarism verdict (if caught beyond reasonable doubt and you want to plead your case), etc.

To get the most out of the tutorial part of these sessions, please try out all the questions in the tutorial component and give some answer even if you encounter difficulties in answering some of them. Before, during, or after the tutorial session, don't hesitate to clear up all doubts and questions you might have, with the tutor.

Every week, you will try to solve one medium (or two easy) selected Kattis problem during the 'hands-on/lab' component. The tutors already know the selected Kattis problems for this semester. However, these selected problems will be revealed to you on the spot each week (if you happen to already solve it, then you are free to just leave the session or actually you can stay back to help your

peers – you can learn more things by observing how others approached the problem that you have solved – possibly with just one way). Tutor will guide all students to get (near) Accepted solution for each problem. These problems are not graded but attempting them during the hands-on time (and possibly to fully complete them afterwards) is beneficial to better understand CS2040S material.

The tutorial/lab participation marks returns to encourage class participation. These marks will be given by the tutor **at the end of the semester** using the following guideline:

- 0% if you only attend ≤ 5 out of 10 tutorial/lab sessions (we lose Monday of Week 13, due to Deepavali PH in-lieu),
- 1% for **at most the bottom three** most-passive students (assuming these students attend > 5 tutorial/lab sessions),
- 3% for **at least the top three** most-active students (answering questions when asked by TA – the correctness of your answers are secondary; or even just by asking your own questions to TA before/during/after class/during consultation); in each tutorial group, and
- 2% for the rest.

2 Tutorial 01 Questions

Q1). You are given a simple Java program below:

```
class ListArray<T> { // question 1a
  private int N;
  private Object[] A = new Object[100]; // question 1b
  public ListArray() {
    N = 0;
  }
  @SuppressWarnings("unchecked")
  public T get(int i) {
    return (T) A[i]; // question 1c
  }
  public int indexOf(T v) {
    for (int i = 0; i < N; ++i)
      if (A[i].equals(v))
        return i;
    return -1;
  }
  public void add(T v) { // question 1d
    if (N == 100)
      return;
    A[N++] = v;
  }
  public void add(int i, T v) {
    if (N == 100 || i < 0 || i > N) // question 1e
      return;
    for (int j = N-1; j >= i; --j)
    // for (int j = i; j <= N-1; ++j) // question 1f
      A[j+1] = A[j];
    A[i] = v;
    ++N;
  }
  public void remove(int i) {
    for (int j = i; j < N-1; ++j) // question 1g
      A[j] = A[j+1];
    --N;
  }
```

```
  public void printList() {
    for (int i = 0; i < N; ++i)
      System.out.print((i > 0 ? " " : "") + A[i]); // question 1h
    System.out.println();
  }
  public void sortList() {
    // sort array A, question 1i
  }
}

class ListArrayTest {
  public static void main(String[] args) {
    ListArray<Integer> LA = new ListArray<>();
    LA.add(5);
    LA.add(0, 1);
    LA.add(0, 4);
    LA.add(0, 7);
    LA.add(0, 2); // we should have A = {2, 7, 4, 1, 5} by now
    System.out.println(LA.get(3)); // 1, A[3] = 1
    System.out.println(LA.indexOf(4)); // 2, A[2] = 4
    System.out.println(LA.indexOf(6)); // not found, -1
    LA.remove(1); // we should have A = {2, 4, 1, 5} by now
    System.out.println(LA.indexOf(4)); // 1, A[1] = 4 now
    System.out.println(LA.indexOf(7)); // not found, -1
    LA.printList(); // unsorted
    LA.sortList(); // we should have A = {1, 2, 4, 5} by now
    LA.printList(); // sorted
  }
} // please copy paste the code above, test compile, and run it yourself
```

This code will be revisited soon during discussion of List ADT (read `https://visualgo.net/en/list?slide=2-1` until 2-8). For now, please answer the following sub-questions (see the comments):

(a) What does this line means?

(b) Anything wrong with this line?

(c) Any potential issue with this line?

(d) What is the difference of this 'add' versus the other 'add'?

(e) What does this line means?

(f) What if we use this commented line instead of the line before it? Any potential issue?

(g) Any potential issue with this line?

(h) What does this line means?

(i) Implement this routine using any sorting algorithm that you know!

(a) This is called Java Generics, see `https://www.programiz.com/java-programming/generics`.

(b) We cannot have ListArray that contains more than 100 items (of type T). This is the limitation of (fixed-size) array for List ADT that will be explained again soon.

(c) There is no safeguard (see explanation of the next sub-question below). If we access `i` that is not inside [0..`N`-1], we will cause a crash/wrong answer.

(d) The first 'add' adds 'v' to the back of the ListArray, the second 'add' (polymorphism) adds 'v' at specific index 'i'.

(e) It is a safeguard mechanism to prevent unnecessary runtime error.

(f) If we use the commented line instead of the correct one, we will accidentally overwrite the values wrongly, making all values in `A[i..N-1]` to be all `A[i]` (before `A[i]` is overwritten by `v`).

(g) Everything is correct. Notice that if we have a list of `N` items, we can have `N+1` possible insertion points, at index [0..`N`]. Again we are limited to 100 items in this example class that uses fixed-size array.

(h) This is a ternary operation. If `i` is positive (greater than 0), i.e., the second index onwards, we print a space before printing the next item. This way, we have spaces only in-between the integers and we do not have a trailing space before newline... Kattis online judge that we use in this course usually forgives this.

(i) You can use bubble, selection, or insertion sort for this case. As `N` is not more than 100, these slow $O(N^2)$ sorting algorithms are OK. Writing your own $O(N \log N)$ Merge Sort or Quick Sort is overkill. But the best is to just use sorting library for this.

Analysis/Order of Growth

Q2). What is the *tightest* bound of the following function? $\mathbf{F(}n\mathbf{)} = \log(\mathbf{2}^n) + \sqrt{n} + \mathbf{100\,000\,000}$

1. O(n)
2. O($n \log n$)
3. O(n^2)
4. O(1)
5. O(2^n)

Ans: **1. O(n).**
$log(2^n)+ \sqrt{n}+100\,000\,000$ is dominated by the term $log(2^n)$, which is n.
Note: The base of this logarithm is not specified, but we can multiply this term with a constant value $\frac{1}{\log 2}$ to have $\log_2(2^n) = n$.

Q3. What is the *tightest* bound of the following functions below: **F**(n), **G**(n), and **H**(n)?
The options:

1. O(2^n)
2. O(n^2)
3. O($n \log n$)
4. O(n)
5. O($\log^2 n$)
6. O($\log n$)
7. O(1)
8. none of the above

Q3.a). $\mathbf{F}(n) = n + \frac{1}{2}n + \frac{1}{3}n + \frac{1}{4}n + \frac{1}{5}n + \frac{1}{6}n + \frac{1}{7}n + \frac{1}{8}n + \mathbf{... + 1}$

Q3.b). $\mathbf{G}(n) = n + \frac{1}{2}n + \frac{1}{4}n + \frac{1}{8}n + \frac{1}{16}n + \frac{1}{32}n + \frac{1}{64}n + \frac{1}{128}n + \mathbf{... + 1}$
G(n) is basically the sum of the reciprocals of powers of two up to n.
To simplify the analysis, assume that n is a powers of two.

Q3.c). $\mathbf{H}(n) = n + \frac{1}{2}n + \frac{1}{3}n + \frac{1}{5}n + \frac{1}{7}n + \frac{1}{11}n + \frac{1}{13}n + \frac{1}{17}n + \mathbf{... + 1}$
H(n) is basically the sum of the reciprocals of prime numbers up to n.
To simplify the analysis, assume that n is a prime number.
The analysis is very mathematical...
For CS2040S level, just estimate the time complexity of **H**(n) w.r.t **F**(n) and **G**(n).

Ans: **3. O($n \log n$) for F(n)**
4. O(n) for G(n).
8. none of the above for H(n) as it is O($n \log \log n$).

F(n) = $n+\frac{1}{2}n+\frac{1}{3}n+\frac{1}{4}n+\frac{1}{5}n+\frac{1}{6}n+\frac{1}{7}n+\frac{1}{8}n+...+1$
= n*($\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+...+\frac{1}{n}$) **>>> there are n terms here**
= n*$\sum_{x=1}^{n} \frac{1}{x}$ **>>> this is a divergent Harmonic series (we can memorize this)**
= n*O(ln n)
= $O(n \log n)$

***Proof** that $\sum_{x=1}^{n} \frac{1}{x} = O(\log n)$

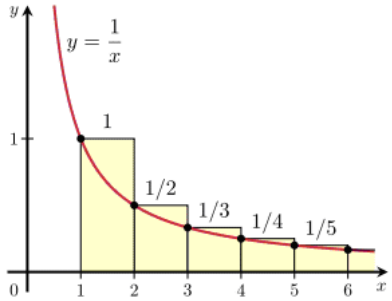

Figure 1:

$\sum_{x=1}^{n} \frac{1}{x}$ is the sum of the area of the rectangles which is bounded by the area under the curve $y=\frac{1}{x}$ as shown in fig. **??** (stolen from Wikipedia).
Area under the curve $y=\frac{1}{x}$ can be expressed as the integral $\int_1^n \frac{1}{x}dx$, and $\int_1^n \frac{1}{x}dx=\ln n$. Thus $\sum_{x=1}^{n} \frac{1}{x}$ $< c*\ln n$ for some constant c (we can choose a c such that the curve is shifted upwards to completely cover the rectangles).
Thus $\sum_{x=1}^{n} \frac{1}{x} = O(ln\ n) = O(\log n)$
Try it in WolframAlpha
`http://www.wolframalpha.com/input/?i=1%2B1%2F2%2B1%2F3%2B1%2F4%2B...`

Alternative explanation (possibly easier to understand):
$F(n) = n+\frac{1}{2}n+\frac{1}{3}n+\frac{1}{4}n+\frac{1}{5}n+\frac{1}{6}n+\frac{1}{7}n+\frac{1}{8}n+...+1$
$= n*(\frac{1}{1}+$
$\frac{1}{2}+\frac{1}{3}+$
$\frac{1}{4}+\frac{1}{5}+\frac{1}{6}+\frac{1}{7}+$
$\frac{1}{8}+...+\frac{1}{n})$ >>> **split the n terms here into 1, 2, 4, 8, ... (powers of two) terms**
$\leq n*(\frac{1}{1}+$
$\frac{1}{2}+\frac{1}{2}+$
$\frac{1}{4}+\frac{1}{4}+\frac{1}{4}+\frac{1}{4}+$
$\frac{1}{8}+...+\frac{1}{n})$ >>> **notice this upper-bounding technique**
$\leq n*(1+$
$1+$
$1+$
$1)$ >>> **each row sums to 1, there are $\log_2 n + 1$ rows**
$= O(n(\log_2 n + 1)) = O(n \log n)$.

$G(n) = n+\frac{1}{2}n+\frac{1}{4}n+\frac{1}{8}n+...+1$
$= n*(\frac{1}{1}+\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+...+\frac{1}{2^{log_2 n}})$ >>> **there are only $\log_2 n$ terms here, contrast this with Q3.a**
$= n*2$ >>> **this is a geometric series that converges to 2**
$= O(n)$
Try it in WolframAlpha
`http://www.wolframalpha.com/input/?i=1%2B1%2F2%2B1%2F4%2B1%2F8%2B...`

$H(n) = n+\frac{1}{2}n+\frac{1}{3}n+\frac{1}{5}n+\frac{1}{7}n+\frac{1}{11}n+...+1$
$= n*(\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\frac{1}{5}+\frac{1}{7}+\frac{1}{11}+...)$
$= n*(1+O(\log\log n))$
$= O(n \log\log n)$ The analysis is very involved and far beyond this module, see `https://en.wikipedia.org/wiki/Divergence_of_the_sum_of_the_reciprocals_of_the_primes#Proof_that_the_series_exhibits_log-log_growth`. What is probably suitable for CS2040S level is for students to *estimate* what is the time complexity of $H(n)$. Stare at the summations and convince yourself that $G(n)$ (Geometric series, simplified to $O(n)$) $< H(n)$ (what we want to estimate; involving reciprocals of primes) $< F(n)$ (Harmonic series, simplified to $O(n \log n)$).

Hands-on 1

TA will run the second half of this session with one chosen Kattis problem involving List ADT.

Kattis /cutinline, a List ADT (with array (or ArrayList, but we don't need the resize-able feature). N is 'small' so $O(N^2)$ solution should still pass. Help students derive the required solutions.

Problem Set 1+2

We will end the tutorial with a short PS1 debrief and an overview of PS2.

PS1 A (/tenis) is not a hard problem, just that it has so many corner cases and can turn into a reading comprehension problem. Just be very very careful.

For PS1 B (/falcondive), we need to use three 2D arrays (frame 1, frame 2, and frame 3). Frame 1 and 2 are the inputs and frame 3 is the output. If a cell is identical in frame 1 and 2, it will also be the same in frame 3. If a cell is different, we can always infer what is behind the Falcon cell via the frame 1 or 2. Finally, find the top left corner of the Falcon in frame 1 and similarly find the top left corner of the Falcon in frame 2. This displacement 'vector' is then applied to all cells of the Falcon for its location in frame 3.

For PS2 A (/universityzoning) requires (several rounds of) sorting. For the first week of PS2, TAs will just describe the problem statement and give a few more sample test cases to ensure all students understand the task (sample 1 and 2 are too trivial).

For PS2 B (/jobbyte) has a naive solution (that will get TLE). Just run $O(N^2)$ selection sort on it. To solve this problem, you will need to come up with an $O(N \log N)$ solution.

3 Note

Remember that outside the official tutorial+lab hours, there will be tutor(s) who will stand by at his/their designated (e-)consultation time slot each week on Fri 8-10pm or Sat 10am-12nn (unless they

mention some exceptions) to answer CS2040S related queries. This information can be found at `https://www.comp.nus.edu.sg/~stevenha/cs2040s.html`, scroll to 'registration' section. Remember that you do NOT have to be in that tutor's class to join a tutor's consultation slot. It is a free-and-easy unstructured session. Anyone who wants to study together with the tutor or have any CS2040S related question(s) can join.