CS2040C Semester 1 2018/2019

Data Structures and Algorithms

Tutorial 01 - C++, Complexity Analysis

For Week 3 (Week Starting 27th August 2018)

Document was last modified on: August 21, 2018

1 Introduction and Objective

The purpose of this first tutorial is to recap the first two weeks of CS2040C: Introduction to C++ and basic analysis of algorithm.

As this is the first tutorial session, we will do a quick ice breaking at the start of the session.

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

The tutorial participation marks to be given by the tutor **at the end of the semester** are as follows:

- 0% if you only attend ≤ 5 out of 10 tutorial sessions (Tuesday of Week 12, 06 Nov 2018 is a Public Holiday - Deepavali. Alternative arrangements will be made nearing that period.),
- 1% for **at most the bottom three** most-passive students (assuming these students attend > 5 tutorial 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); in each tutorial group, and
- 2% for the rest.

2 Tutorial 01 Questions

C++ OOP (basic)

Q1). You are given a simple C++ program that will be revisited soon during discussion of List ADT:

```
#include <iostream> // you can change <iostream> with <bits/stdc++.h>
// (this is basically a header file that includes every standard library)
using namespace std;

class ListArray {
private:
  int N;
  int A[10]; // question 1a
public:
  ListArray() : N(0) {} // question 1b
  int get(int i) {
    return A[i]; // question 1c
  }
  int search(int v) {
    for (int i = 0; i < N; i++)
      if (A[i] == v)
        return i;
    return -1;
  }
  void insert(int i, int v) {
    if ((N == 10) || (i < 0) || (i > N)) return; // question 1d
    for (int j = i; j <= N-1; j++) // question 1e
      A[j+1] = A[j];
    A[i] = v;
    N++;
  }
  void remove(int i) {
    for (int j = i; j < N-1; j++) // question 1f
      A[j] = A[j+1];
    N--;
  }
  void printList() {
    for (int i = 0; i < N; i++) {
      if (i) cout << " ";
      cout << A[i];
    }
    cout << endl;
```

```
  }
  void sortList() { // sort array A, question 1g
    for (int i = 0; i < N-1; i++) {
      for (int j = 1; j < N-i; j++) {
        if (A[j-1] > A[j]) {
          int tmp = A[j];
          A[j] = A[j-1];
          A[j-1] = tmp;
        }
      }
    }
  }
};

int main() {
  ListArray* LA = new ListArray(); // question 1h
  LA->insert(0, 5);
  LA->insert(0, 1);
  LA->insert(0, 4);
  LA->insert(0, 7);
  LA->insert(0, 2); // we should have A = {2, 7, 4, 1, 5} by now
  cout << LA->get(3) << endl; // 1, A[3] = 1
  cout << LA->search(4) << endl; // 2, A[2] = 4
  cout << LA->search(6) << endl; // not found, -1
  LA->remove(1); // we should have A = {2, 4, 1, 5} by now
  cout << LA->search(4) << endl; // 1, A[1] = 4 now
  cout << LA->search(7) << endl; // not found, -1
  LA->printList(); // unsorted
  LA->sortList(); // we should have A = {1, 2, 4, 5} by now
  LA->printList(); // sorted
  return 0;
} // please copy paste the code above, test compile, and run it yourself
```

Now answer the following sub-questions (please refer to the comments in the code above):

(a) Is there anything wrong with this line? 

(b) What does this line mean? 

(c) Is there any potential issue with this line? (also see question 1d below) 

(d) What does this line mean? 

(e) Is there any potential issue with this line? 

(f) Is there any potential issue with this line?

(g) What is the time complexity of this function, in terms of the number of elements in the list, N.

(h) Can we just write `ListArray LA;` in this line?

Analysis/Order of Growth

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

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

Q3.a). What is the bound of the following function? $\mathbf{F}(n) = n + \frac{1}{2}n + \frac{1}{3}n + \frac{1}{4}n + \mathbf{...} + \mathbf{1}$
Q3.b). What about $\mathbf{G}(n) = n + \frac{1}{2}n + \frac{1}{4}n + \frac{1}{8}n + \mathbf{...} + \mathbf{1}$

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. $O(1)$

Sorting (part 1)

Q4). At `https://visualgo.net/en/sorting?slide=1-2`, Visualgo describes a few array applications that become easier/more efficient if the underlying array is sorted. In this tutorial, we will discuss application 1-4 and if time permits, we will discuss application 5-6 (otherwise we will continue the discussion during the next Tut02). Although tutorial timeslots are spread across the week, you should still be able to answer the questions before the sorting lecture is conducted. You may just assume that you are provided with an sorted array of integers as input.