The University of Melbourne
Department of Computing and Information Systems

COMP90038 Algorithms and Complexity, Semester 1, 2016

Assignment **2** (handed out on 4 May 2016)

Total marks: **20** (which count for **10% of your final grade**).

Due date: **20 May 2016 at 8:59 am.**

Objectives

The objectives of this assignment are to:

- improve your understanding of the time complexity of algorithms, the development of efficient algorithms and data structures, and the transform-and-conquer paradigm;
- develop skills in analysis and formal reasoning about complex concepts;
- improve written communication skills; in particular the ability to present algorithms clearly, precisely and unambiguously.

Note

You may use standard operations on the data structures as discussed in lectures (you need to list the lecture week and slide number where the operation you use is defined). For question 2, you may also assume that any geometric function that you might need is already available, for example, you might assume the existence of a function to compute the area of a triangle and any other geometric function you might need. However, you need to document precisely what you expect these functions to do, i.e., what is their input and what do they return, but you do not need to implement them or write the pseudocode for them.

Question 1. [4 marks]

A. What is the main difference between a BST and a (max) heap?

B. Can we traverse the n nodes of a (max) heap in sorted order in O(n) runtime? If this is possible, outline an algorithm. If this is not possible, explain why.

Question 2. [8 marks]

Let P be a finite set of n distinct points in $\mathbb{R}^2$. The set P is called *completely triangulable* if for any three points p, q and r of P the area of their triangle is always positive. Develop an algorithm that determines if a finite set P is completely triangulable. The algorithm returns TRUE if P is completely triangulable and FALSE otherwise. You need to write the pseudocode of the algorithm. Your algorithm has to run in $O(n^2 \log n)$ time for full marks. Show that your algorithm achieves $O(n^2 \log n)$ runtime.

Question 3. [8 marks]

Develop an algorithm with $O(n\log k)$ runtime that sorts k sorted lists into a single list. Assume that $n>k$, where n is the total number of elements over all lists. You may assume that all n elements are pairwise distinct. First, explain your idea for your algorithm, then write its pseudocode. Show that your algorithm is correct and that you achieve $O(n\log k)$ runtime.

Submission and Evaluation

- Submit a PDF document via the LMS. Note: Microsoft Word submissions are not acceptable — if you use Word, create a PDF version for submission.
- Marks are primarily allocated for correctness, but elegance of algorithms and how clearly you communicate your thinking will also be taken into account. Where indicated, the complexity of algorithms also matters.
- We expect your work to be neat—parts of your submission that are difficult to read or decipher will be deemed incorrect. Make sure that you have enough time towards the end of the assignment to present your solutions carefully.

If you questions, email the Head Tutor, Toby Davis <davies@student.unimelb.edu.au> or the Lecturer Lars Kulik <lkulik@unimelb.edu.au>, with a precise description of the problem, bring it up at a lecture, or use the LMS discussion board. Late submission will be possible, but a late submission penalty will apply: a flagfall of 2 marks, and then 1 mark per 12 hours of being late.