COMP 3804 — Assignment 2

Due: Sunday February 28, 23:59.

Assignment Policy:

- Your assignment must be submitted as one single PDF file through cuLearn.

> Use the following format to name your file:
>
> `LastName_StudentId_a2.pdf`

- **Late assignments will not be accepted. I will not reply to emails of the type "my internet connection broke down at 23:57" or "my scanner stopped working at 23:58", or "my dog ate my laptop charger".**
- You are encouraged to collaborate on assignments, but at the level of discussion only. When writing your solutions, you must do so in your own words.
- Past experience has shown conclusively that those who do not put adequate effort into the assignments do not learn the material and have a probability near 1 of doing poorly on the exams.
- When writing your solutions, you must follow the guidelines below.
  - You must justify your answers.
  - The answers should be concise, clear and neat.
  - When presenting proofs, every step should be justified.

Question 1: Write your name and student number.

Question 2: You are given a sequence S of n numbers. An element x in S is called a *majority element* if it occurs more than $n/2$ times in S.

This question asks you to describe two algorithms that decide if the sequence S contains a majority element; if it does, the algorithm returns it; otherwise, the algorithm returns the message "there is no majority element".

You are highly encouraged to use any algorithm and any result that was discussed in class. In other words, try to make your algorithms as short as possible by using algorithms discussed in class as "black boxes".

(2.1) How many majority elements can there be? Justify your answer.

(2.2) Show how the majority problem can be solved in $O(n \log n)$ time. Argue why your algorithm is correct and why the running time is $O(n \log n)$.

(2.3) Show how the majority problem can be solved in $O(n)$ time. Argue why your algorithm is correct and why the running time is $O(n)$.

Question 3: Some textbooks contain statements of the form "The running time of algorithm $\mathcal{A}$ is at least $O(n^2)$". Explain why such a statement does not make sense.

Question 4: You are given k lists, each one containing a sorted sequence of numbers. Let n denote the total number of elements in all these lists. Give an $O(n \log k)$–time algorithm that merges these k lists into one sorted list. Explain why the running time of your algorithm is $O(n \log k)$.

Hint: Use a min-heap. If $k = 2$, this problem should look familiar to you.

Question 5: You are given a min-heap $A[1 \ldots n]$ on n elements and an integer k with $1 \leq k \leq n$. Give an algorithm that computes the k smallest elements in A in sorted order. The running time of your algorithm must be $O(k \log k)$. Explain why your algorithm is correct and why its running time is $O(k \log k)$.

Hint: It is easy to get an $O(k \log n)$–time algorithm. In a min-heap, the root of any subtree contains the minimum of all elements in that subtree. You may think of forming a min-heap consisting of $O(k)$ elements using the elements of the given heap A of n elements. As always, you may use any algorithm that was discussed in class.

Question 6: Let $G = (V, E)$ be an undirected graph, which is given to you in the adjacency list format. Recall that $\mathit{degree}(u)$ denotes the degree of vertex u. Let $\mathit{twodegree}(u)$ be the sum of the degrees of u's neighbors, i.e.,

$$\mathit{twodegree}(u) = \sum_{v:\{u,v\} \in E} \mathit{degree}(v).$$

Describe an algorithm that computes $\mathit{twodegree}(u)$ for all vertices u, in $O(|V| + |E|)$ total time. Justify your answer.

Question 7: Let $G = (V, E)$ be a directed acyclic graph. In class, we have seen the following algorithm for topologically sorting the vertices of G:

- Set $k = 1$.
- While the graph is non-empty:
  - Find a vertex v of indegree zero.
  - Assign the number k to v.
  - Remove v from the graph.
  - Increase k by one.

Show how this algorithm can be implemented such that its running time is $O(|V| + |E|)$.