Assignment 7

csci2200, Algorithms

Instructions:

- Honor code: Work on this assignment with at most one partner. Between different teams, Collaboration is at level 1 [verbal collaboration only]
- Write each problem on a separate page; If a problem has multiple parts, you can write all parts on the same page, as long as you leave space in between them.

---

1. **Majority element:** Suppose we are given an array A of length n with the promise that there exists a majority element (i.e. an element that appears $> \frac{n}{2}$ times). Additionally, we are only allowed to check whether two elements are equal (no $>$ or $<$ comparisons between the elements). Design an $On \lg n)$ algorithm to find the majority element, using divide-and-conquer. Explain the correctness and runtime of your algorithm.

   [*We expect: pseudocode, an English description of the main idea of the algorithm, an (informal) explanation of why it's correct; and running time analysis.*]

2. **The inversion problem:** Let $A[1..n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called an inversion of A. Give an algorihm that determines the number of inversions in an array in $O(n \lg n)$ time worst-case (Hint: modify merge sort).

   We expect: pseudocode and a clear English description of what the algorithm is doing; A brief informal justification of why the algorithm is correct, and its runtime analysis.

   Helper exercises (do not turn in).

   (a) List the inversions of the array $< 2, 3, 8, 6, 1 >$.

   (b) What array with elements from the set $\{1, 2, ..., n\}$ has the most inversions? How many does it have?

   (c) Give an algorithm that determines the number of inversions in an array in $O(n^2)$ time (this is the straightforward solution).

Evaluation

This assignment (and all subsequent assignments) will be evaluated along four general criteria:

1. **Algorithm:** Is the algorithm clearly described ? Is the general idea included? Is high-level pseudocode included?[1]
2. **Correctness**: Does the algorithm solve the problem?
3. **Analysis**: Is the running time of your algorithm analysed?
4. **Style**: Does it look professional and neat? Is the explanation written carefully in complete sentences, and well-organized logic? Is it easily human-readable? Is it compleete yet concise? Is it easy to understand? These kinds of questions do not affect correctness but greatly affect how readable the algorithm is.

[1]Pseudocode should be clear enough that a student who took 1101 can understand what your algorithm is doing and could implement it in a language of their choice, without thinking too hard. At the same time, pseudocode is **not actual code**, and should not include details that are straightforward and make the ideas hard to follow. For e.g.it is preffered to say "find the max element in the array" (basic straightforward process) rather then spell it out.