Assignment 6

Algorithms, Spring 2024

Honor code: *Work on this assignment alone or with one partner (highly encouraged). Partner policy: You and your partner will work together on the assignment throughout the whole process, you will write it and review it together, and will submit one assignment. You can talk to anyone in the class (collaboration level 1). It is not allowed to search online for the specific problems in this assignment—doing so violates academic honesty for the class.*

---

1. **Majority element via divide-and-conquer:** 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 $> \lfloor \frac{n}{2} \rfloor$ times). The elements are so that you cannot check whether an element is $>$ or $<$ to another, you can only check if an element is $=$ to another (imagine for e.g. images). Use divide-and-conquer to design an algorithm that finds the majority element in $O(n \lg n)$ time.

   We expect: a brief English description of the main idea of the algorithm, high-level pseudocode, justification of correctness (ie why it finds the majority), and running time analysis.

   Hint: To start, you may remember that you solved a similar problem using the selection algorithm — using that if a majority element exists, it must be the median. But the Selection algorithm needs to compare elements using $<$ or $>$ in order to place them before or after the pivot, so you cannot use it in this case.

   To use a divide-and-conquer strategy start by formulating and a claim like so: If an element x is the majority element in A, then x must be ..... in the first half of A ... . in the second half of A. Justify that your claim is correct using a contradiction argument.

2. **Stock market problem:** You are given an array of numbers representing a stock's prices over n days. Your goal is to identify the longest consecutive number of days during which the stock's value does not decrease.

   For example, consider the stock value below:

   ```
   Day:      1   2   3   4   5   6   7   8
   Value:  113 111 119  116 116 119 115 127
   ```

   In this example the length of the longest consecutive non-decreasing run is 3 (and goes from day 4 to day 7).

3. **The inversion problem:** Let A 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. Design an algorihm that determines the number of inversions in an array in $O(n \lg n)$ time.

   We expect: a brief description of what the algorithm is doing, high-level pseudocode, a brief informal justification of why the algorithm correctly counts the inversions, and runtime analysis.

   Hint: modify merge sort.

   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 refered to as the "straightforward" solution).