Problem set 2

Date : Jan 10, 2022 Algorithms-S22

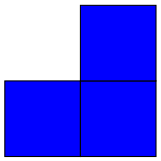
Instructions

- The problem sets are not for submission and will not be graded. However, you are strongly encouraged to spend some time thinking about the questions. This might be helpful in internalizing some of the things that we would discuss in the lectures.
- To get the most out of problem sets, you are strongly encouraged to think about the problems on your own before discussing with others, consulting any references or looking at hints (that some of the problems might have).

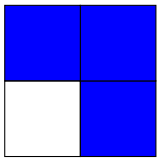
Problems

1. Given an array A of distinct integers an inversion is a pair $(i, j) \in \mathbb{N} \times \mathbb{N}$ of indices such that $i < j$ and $A[i] > A[j]$. Design an $O(n \log n)$ time algorithm for counting the number of inversions in an n length (possibly unsorted) input array. Prove its correctness and the bound on the running time.
2. Design and analyse the fastest algorithm you can for the following problem: the input to the algorithm is an array of distinct integers of length n and the goal is to output the two largest integers in the array.
3. Consider a simpler version of the previous problem where the goal is to output the largest (and not the two largest) elements in the input array. Prove that any algorithm for this problem requires at least $n - 1$ comparisons.

   Note that this problem is not just asking you to prove that *your algorithm* for computing the maximum element requires at least $n - 1$ comparisons, but *any* algorithm requires at least $n - 1$ comparisons. [1]
4. Given below A is a building block. Show that for every $k \in \mathbb{N}, k \geq 1$, a square with side length 2^k units can be filled using A such that only bottom left corner is left.



A



$k = 1$

5. An array of integers is said to be *unimodal* if all its entries are distinct and its entries are in increasing order up until its maximum element, after which its elements are in decreasing order. Design (and analyse) an algorithm to compute the maximum element of a unimodal array that runs in $O(\log n)$ time.

[1] Arguments like these are called *lower bounds* in the study of algorithms. We will not see many such lower bounds in this course, but proving better and improved lower bounds is an important, active and deep area of research in computer science and mathematics and is discussed in a bit more detail in advanced courses on algorithms or in courses related to computational complexity.

6. You are given a sorted (in increasing order) array A of n distinct integers which can be positive, negative or zero. Design the fastest algorithm you can for deciding whether or not there is an index i such that $A[i] = i$.