UNIVERSITY OF LIVERPOOL

Department of Computer Science
Dr. Sebastian Wild · Ben Smith

Date: 2023-10-16
Version: 2023-10-14 19:35

Tutorial 4 for COMP 526 – Efficient Algorithmics, Fall 2023

Problem 1 (Sorted with outliers)

In this task, we consider a different type of presortedness:

Given an array $A[0..n)$, we say A is *d-deletion-sortable* if there are indices $0 \le i_1 < i_2 < \cdots i_d < n$, so that after deleting the positions $i_1, \ldots, i_d$ from A, the resulting sequence is sorted. For example

$$2, 4, 1, 6, 7, 5, 8, 12, 0$$

is 3-deletion-sortable (by removing elements $1, 5, 0$), but it is not 2-deletion-sortable.

In the following, we always assume that we are given an array $A[0..n)$ that is d-deletion-sortable. For simplicity, you may assume that the elements in A are pairwise different.

a) Design an adaptive sorting algorithm for A when you are also *given* (a sorted array $D[0..d)$ of) the *indices* $i_1, \ldots, i_d$ to delete.

   Assuming $d \ll n$, your algorithm should run in time $o(n \log n)$; more precisely, a full solution would sort a $\sqrt{n}$-deletion-sortable $A[0..n)$ in $O(n)$ time.

   Describe your algorithm (in clear prose or pseudocode) and analyze its running time (as a Θ-class).

b) How large can d be before your solution requires $\omega(n)$ time?
   How large can d be before your solution requites $\Omega(n \log n)$ time?

c) *Bonus problem:*
   Design an algorithm as in a), but this time you are neither given d, nor the indices to delete.

   Hint: Can you find a set of indices I, so that A is sorted after removing those indices (without making I too big)? You may not be able to achieve $|I| \le d$ easily, but it is sufficient to be "not too far" from d.