

CSE 101 Homework 3

Speeding up algorithms with restructuring, preprocessing and data structures.

Due Tuesday, April 23

80 points total %

Solve each problem. For algorithm problems, if the problem only specifies that you need to give a proof of correctness, then no time analysis is required. If it specifies that you need to give an efficient implementation, then you do not need to give a correctness proof for the basic strategy (just explain why your version actually carries out the strategy). If it says to do both, or doesn't specify what parts you need, you need to give both a proof of correctness and time analysis.

- True/False (10 points each: explain your answer)**
1. If Alg1 takes time $\Theta(n^2)$ and produces an output of length $n \log n$, and Alg2 takes time $\Theta(n^2)$, what is the time complexity of an algorithm that runs Alg1 and then runs Alg2 on the output of Alg1? Explain your answer.
 2. If on a graph with n nodes and m edges, Alg1 takes time $O(nm)$ and Alg2 takes time $O(n^2 \log n)$, when is it better to use Alg1? Explain.

Merging k sorted lists (40 pts) You are given k sorted linked lists, and want to merge them into a single sorted list containing all of the elements in all of the lists. (Pointers to the heads of the sorted lists are arranged in an array of size k .) There are n elements total in all of the lists, but the lists are not necessarily of the same size. Give an efficient algorithm for this problem. Be sure to give a correctness proof and proof of the time analysis.

Implementation-20 points Implement a naive $O(n^2)$ time sorting algorithm (such as bubble sort) and heap-sort. You can use heaps from a standard library to implement heap-sort. Plot their performance on random arrays of n integers with values between 1 and n , for $n = 2^6, 2^8, 2^{10}, 2^{12}, 2^{14}, 2^{16}$. Plot their performance on a log-log scale. Is heap-sort always better than bubble-sort? Why or why not?