CS 170 HW 2

Due on 2018-09-09, at 11:59 pm

1 (★) Study Group

List the names and SIDs of the members in your study group.

2 (★★) Counting inversions

This problem arises in the analysis of *rankings*. Consider comparing two rankings. One way is to label the elements (books, movies, etc.) from 1 to n according to one of the rankings, then order these labels according to the other ranking, and see how many pairs are "out of order".
We are given a sequence of n distinct numbers $a_1, \cdots, a_n$. We say that two indices $i < j$ form an inversion if $a_i > a_j$, that is if the two elements a_i and a_j are "out of order". Provide a divide and conquer algorithm to determine the number of inversions in the sequence $a_1, \cdots, a_n$ in time $O(n \log n)$ (Four part solution required. *Hint:* Modify merge sort to count during merging).

3 (★★★) Two sorted arrays

You are given two sorted arrays, each of size n. Give as efficient an algorithm as possible to find the k-th smallest element in the union of the two arrays. What is the running time of your algorithm as a function of k and n? (Four part solution required.)

4 (★★★★) Majority Elements

An array $A[1 \ldots n]$ is said to have a *majority element* if more than half of its entries are the same. Given an array, the task is to design an efficient algorithm to tell whether the array has a majority element, and if so to find that element. The elements of the array are not necessarily from some ordered domain like the integers, so there can be **no** comparisons of the form "is $A[i] > A[j]$?". (Think of the array elements as GIF files, say.) The elements are also not hashable, i.e., you are *not* allowed to use any form of sets or maps with constant time insertion and lookups. However you *can* answer questions of the form: "is $A[i] = A[j]$?" in constant time. Four part solutions are required for each part below.

(a) Show how to solve this problem in $O(n \log n)$ time. (Hint: Split the array A into two arrays A_1 and A_2 of half the size. Does knowing the majority elements of A_1 and A_2 help you figure out the majority element of A? If so, you can use a divide-and-conquer approach.)

(b) Can you give a linear-time algorithm? (This algorithm would run in $O(n \log n)$ time, but you *should not* reuse this algorithm to answer part a)

5 (★★★★★) Merged Median

Given k sorted arrays of length l, design a *deterministic* algorithm (i.e. an algorithm that uses *no* randomness) to find the median element of all the $n = kl$ elements. Your algorithm should run asymptotically faster than $O(n)$.

(You need to give a four-part solution for this problem.)

6 (★★) Fourier Transform Basics

Answer the following. Do any required matrix multiplications by hand, using e.g. a Fourier transform calculator will not get you full credit.

(a) What is the Fourier transform of $(3, i, 2, 4)$?

(b) Find x and y such that $\text{FT}(x) = (5, 2, 1, -i)$ and $\text{FT}(y) = (4, 4, i, i)$.

(c) *Using part b*, find z such that $\text{FT}(z) = (1, -2, 1 - i, -2i)$. (*Hint:* Observe that $\text{FT}(v)$ is a linear transform of v, and recall the properties of linear transforms).

(d) Compute $(2x^2 + 1)(x + 4)$ using a Fourier transform. (*Hint:* Recall that to multiply two polynomials, first, they must both be converted by the Fourier transformation, then multiplied pointwise, and finally converted back to coefficient form)

7 (★★) Modular Fourier Transform

Fourier transforms (FT) have to deal with computations involving irrational numbers which can be tricky to implement in practice. Motivated by this, in this problem you will demonstrate how to do a Fourier transform in modular arithmetic, using modulo 5 as an example.

(a) There exists $\omega \in \{0, 1, 2, 3, 4\}$ such that all the powers $\omega, \omega^2, \ldots, \omega^4$ are distinct (modulo 5). When doing the FT in modulo 5, this ω will serve a similar role to the primitive root of unity in our standard FT. Find such an ω (there are multiple, you only need to find one), and show that $\omega + \omega^2 + \omega^3 + \omega^4 = 0 \pmod 5$. (Interestingly, for any prime modulus there is such a number.)

(b) Using the matrix form of the FT, produce the transform of the sequence $(0, 1, 0, 2)$ modulo 5; that is, multiply this vector by the matrix $M_4(\omega)$, for the value ω you found earlier. Be sure to explicitly write out the FT matrix you will be using (with specific values, not just powers of ω). In the matrix multiplication, all calculations should be performed modulo 5.

(c) Write down the matrix necessary to perform the inverse FT. Show that multiplying by this matrix returns the original sequence. (Again all arithmetic should be performed modulo 5.)

(d) Now show how to multiply the polynomials $2x^2 + 3$ and $-x + 3$ using the FT modulo 5.

8 (★★★) Polynomial from roots

Given a polynomial with exactly n distinct roots at $r_1, \ldots, r_n$, compute the coefficient representation of this polynomial in time. Your runtime should be $O(n \log^c n)$ for some constant c (you should specify what c is in your runtime analysis). There may be multiple possible answers, but your algorithm should return the polynomial where the coefficient of the highest degree term is 1. You can give only the main idea and runtime analysis, a four part solution is not required.

Note: A root of a polynomial p is a number r such that $p(r) = 0$. The polynomial with roots $r_1, ..., r_k$ can be expressed as $\prod_i (x - r_i)$.