HW 1

Due on **Wednesday, February 24, 2021 at 10am CST**

CS 498: Topics in Algorithms, Spring 2021 Version: **1.12**

- You can work with other people, and submit as a group. The groups can be of arbitrary size.
- For each of the following questions (or subquestions) provide a short concise description of the solution. You do not need to provide the details (but you should figure them out for yourself, naturally).
- Solutions should be typed using LaTeX. Please submit via email both the latex file and the pdf file.
- In the top of your solution include the netid, and names of people in the group submitting the homework.
- Submission of homework is via email.
- You are allowed to use any source in your solution, but (A) you need to write your own solutions (no cut and paste), and (B) you need to cite any such source you used.
- Sub-questions/questions that require you to prove something can be ignored (you might want to verify you know how to prove the claim).

---

1 (100 PTS.) Absolutely not subset sum.

Let $B = \{b_1, \ldots, b_m\} \subseteq [\![U]\!] = \{1, 2, \ldots, U\}$. A number $t \leq U$ is ***n*-representable** by B, if there exists integer numbers $\alpha_i \geq 0$, for $i = 1, \ldots, m$, such that

(i) $\sum_{i=1}^{m} \alpha_i = n$, and

(ii) $\sum_{i=1}^{m} \alpha_i b_i = t$.

Show how to compute, as fast as possible, if t is n-representable by B by an algorithm with running time close to linear in m and U (the dependency of the running time on n should be polylogarithmic in n).

[Hint: Use FFT.]

To make life easy for you, I broke it into three steps:

1.A. (30 PTS.) Show how to solve the case $n = 2$.

1.B. (20 PTS.) Show how to solve the case that n is a power of 2.

1.C. (50 PTS.) Show how to solve the general problem.

(As usual, the solutions to (A) and (B) are much simpler than (C), and are useful in solving (C).)

2 (100 PTS.) Computing Polynomials Quickly

In the following, assume that given two polynomials $p(x), q(x)$ of degree at most n, one can compute the polynomial remainder of $p(x) \bmod q(x)$ in $O(n \log n)$ time. The ***remainder*** of $r(x) = p(x) \bmod q(x)$ is the unique polynomial of degree smaller than this of $q(x)$, such that $p(x) = q(x) * d(x) + r(x)$, where $d(x)$ is a polynomial.

Let $p(x) = \sum_{i=0}^{n-1} a_i x^i$ be a given polynomial.

2.A. (25 PTS.) Prove that $p(x) \bmod (x - z) = p(z)$, for all z.

2.B. (25 PTS.) We want to evaluate $p(\cdot)$ on the points $x_0, x_1, \ldots, x_{n-1}$. Let

$$P_{ij}(x) = \prod_{k=i}^{j}(x - x_k)$$

and

$$Q_{ij}(x) = p(x) \bmod P_{ij}(x).$$

Observe that the degree of Q_{ij} is at most $j - i$.
Prove that, for all x, $Q_{kk}(x) = p(x_k)$ and $Q_{0,n-1}(x) = p(x)$.

2.C. (25 PTS.) Prove that for $i \le k \le j$, we have

$$\forall x \quad Q_{ik}(x) = Q_{ij}(x) \bmod P_{ik}(x)$$

and

$$\forall x \quad Q_{kj}(x) = Q_{ij}(x) \bmod P_{kj}(x).$$

2.D. (25 PTS.) Given an $O(n \log^2 n)$ time algorithm to evaluate $p(x_0), \ldots, p(x_{n-1})$. Here $x_0, \ldots, x_{n-1}$ are n given real numbers.

3 (100 PTS.) Lower bound on sorting network

Prove that an n-input sorting network must contain at least one comparator between the ith and $(i+1)$st lines for all $i = 1, 2, ..., n-1$.

4 (100 PTS.) First sort, then partition

Suppose that we have $2n$ elements $< a_1, a_2, ..., a_{2n} >$ and wish to partition them into the n smallest and the n largest. Prove that we can do this in constant additional depth after separately sorting $< a_1, a_2, ..., a_n >$ and $< a_{n+1}, a_{n+2}, ..., a_{2n} >$.

5 (100 PTS.) Easy points.

Let $S(k)$ be the depth of a sorting network with k inputs, and let $M(k)$ be the depth of a merging network with $2k$ inputs. Suppose that we have a sequence of n numbers to be sorted and we know that every number is within k positions of its correct position in the sorted order, which means that we need to move each number at most $(k-1)$ positions to sort the inputs. For example, in the sequence 3 2 1 4 5 8 7 6 9, every number is within 3 positions of its correct position. But in sequence 3 2 1 4 5 9 8 7 6, the number 9 and 6 are outside 3 positions of its correct position.
Show that we can sort the n numbers in depth $S(k) + 2M(k)$. (You need to prove your answer is correct.)

6 (100 PTS.) Matrix Madness

We can sort the entries of an $m \times m$ matrix by repeating the following procedure k times:

(I) Sort each odd-numbered row into monotonically increasing order.
(II) Sort each even-numbered row into monotonically decreasing order.
(III) Sort each column into monotonically increasing order.

6.A. (8 PTS.) Suppose the matrix contains only 0's and 1's. We repeat the above procedure again and again until no changes occur. In what order should we read the matrix to obtain the sorted output ($m \times m$ numbers in increasing order)? Prove that any $m \times m$ matrix of 0's and 1's will be finally sorted.

6.B. (8 PTS.) Prove that by repeating the above procedure, any matrix of real numbers can be sorted. [Hint:Refer to the proof of the zero-one principle.]

6.C. (4 PTS.) Suppose k iterations are required for this procedure to sort the $m \times m$ numbers. Give an upper bound for k. The tighter your upper bound the better (prove you bound).