CSCI 5454 Spring 2023: Assignment #3
Due Date: Thursday, February 16, 2023.

Topics: Divide-And-Conquer Algorithms and Fast-Fourier Transform

---

P1. (30 points) In class, we showed a divide and conquer algorithm for Fast Fourier Transforms (FFT) wherein given a sequence $\langle a_0, \ldots, a_{n-1}\rangle$, we divided into two subsequences of length $n/2$ to wit: $a_0, a_2, \ldots, a_{n-2}$ and $a_1, a_3, \ldots, a_{n-1}$. Assume n is some power of two.

In this problem, we ask you to consider dividing the sequence of n numbers into two subsequences: $\langle a_0, \ldots, a_{n/2-1}\rangle$ and $\langle a_{n/2}, \ldots, a_{n-1}\rangle$.

Part A (2 points) Let polynomial $p(x) = \sum_{k=0}^{n-1} a_k x^k$ and consider polynomials $p_1(x) = \sum_{k=0}^{n/2-1} a_k x^k$ and $p_2(x) = \sum_{k=0}^{n/2-1} a_{k+n/2} x^k$. Express $p(x)$ in terms of p_1 and p_2.

Part B (10 points) Consider the sequence $\langle b_0, \ldots, b_{n/2-1}\rangle$ wherein

$$b_0 = a_0 + a_{n/2}, b_1 = a_1 + a_{n/2+1}, \cdots, b_j = a_j + a_{n/2+j}, \cdots, b_{n/2-1} = a_{n/2-1} + a_{n-1}\,.$$

Let $\langle B_0, \ldots, B_{n/2-1}\rangle$ be the FFT for that sequence. Show that $A_{2k} = B_k$ for any $k = 0, \ldots, n/2-1$.

Part C (10 points) Consider the sequence $\langle c_0, c_1, \ldots, c_{n/2-1}\rangle$ wherein $c_k = (a_k - a_{k+n/2}) \times \omega_n^k$. Let $\langle C_0, \ldots, C_{n/2-1}\rangle$ be its Fourier coefficients. Show that $A_{2k+1} = C_k$.

Part D (8 points) Propose an alternative divide and conquer scheme for computing FFT by putting together the two observations from the previous two parts. Write the pseudo-code and derive the complexity. In the pseudo code there is no need to write down the code for base case: just add a comment to the effect that the "base case code goes here" or assume there is a function `naive-dft` that would compute DFT coefficients in $O(n^2)$ time.

P2 (20 points) We have a directory in a computer with n files where n is a large even number. These files are numbered from 1 to n. Design an efficient algorithm to check if a majority of the files are identical and if so which file it is.

- If n is even then majority means at least one more than $n/2$. If n is odd, majority is simply $\lceil n/2 \rceil$.
- You can check if two files are the same (presumably by rapid bitwise comparison).
- You cannot sort the files based on their contents since there is no $\leq$ comparison between files.
- No hashing, bitwise operations or arithmetic operations over file contents allowed.

Part A (10 points) Design a divide and conquer algorithm for this problem that runs in $O(n \log n)$ time. Justify the time complexity briefly.

Part B (10 points) Design an even more efficient divide and conquer algorithm that runs in $O(n)$ time. Justify why it runs in O(n) time.

Hint: Pair up the first file with the second, third with the fourth and so on. Test for each of the pair of files if they are the same. If not, discard both files in the pair. If yes, keep just one of the files in the pair. If there was a majority of identical files, what can you say about the remaining files? If there was no majority, then what? Rinse and repeat the process. Think of modifications if n were an odd number?

P3 (10 points) In class, we demonstrated the algorithm to find the closest pair of points on a plane. We obtained the recurrence

$$T(n) = 2T(n/2) + O(n \log(n)) \,,$$

and a suitable base case. In the previous assignment, we proved that this leads to a overall time complexity of $O(n(\log(n)^2))$.

Explain how you would modify the algorithm to compute the closest pair of points in $O(n \log(n))$ time. Recall the outline of the algorithm:

- Sort the points using their x coordinates.
- Recursively carry out the following steps:
  - If number of points is less than 10 just do pairwise comparisons.
  - Otherwise, split the list into two lists of size $n/2$ along the median x coordinate.
  - Recursively compute the closest pair for both halves.
  - Take the minimum of the two distances obtained from the two calls.
  - Sort the points in the "central strip" along the y coordinate.
  - Iterate through the list and compare each point in the sorted list from previous step to the eight previous points.

(i) Explain which steps contribute to the $n(\log(n))^2$ complexity; (ii) explain how that step can be made faster to obtain $n \log(n)$ complexity.