Practice problems (don't turn in):

1. [DPV] Problem 2.8 (FFT practice)
2. [DPV] Problem 2.17 (fixed point)

Problem 1 [DPV] Problem 2.9 part (b)

Do polynomial multiplication by FFT for the pair of polynomials $1 + x + 2x^2$ and $2 + 3x$. (You might try part (a) for practice.)

Answer: (Yes, show your work!)

$n = 4$ $a = (1, 1, 2, \emptyset)$ $\omega_n = 1, i, -1, -i$

$b = (2, 3, \emptyset, \emptyset)$ $\omega_n^2 = 1, -1$

$a_e = (1, 2)$, $a_o = (1, \emptyset)$, $b_e = (2, \emptyset)$ $b_o = (3, \emptyset)$

$FFT(A, i) = FFT(A_e, -1) + i\, FFT(A_o, -1)$

$FFT(A_e, -1) = (\overset{1}{1} + 2, \overset{-1}{1} - 2) = (3, -1)$

$FFT(A_o, -1) = (1, 1)$

$FFT(A, i) = (\overset{1}{3} + \overset{i}{1} \cdot 1, \overset{-1}{-1} + \overset{-i}{i} \cdot 1, 3 + (-1) \cdot 1, -1 + (-i) \cdot 1)$

$= (4, -1 + i, 2, -1 - i)$

$FFT(B_e, -1) = (2, 2)$, $FFT(B_o, -1) = (3, 3)$

$FFT(B, i) = (5, 2 + 3i, -1, 2 - 3i)$

$C_v = (20, -5 - i, -2, -5 + i)$

$(-1 + i)(2 + 3i) = -2 - 3i + 2i + 3i^2$

$-5 - i \quad -3$

$(-1 - i)(2 - 3i)$

$-2 + 3i - 2i - 3$

$-5 + i$

Mazahet Mendiola
P9 2.95

$$C = \frac{1}{n} FFT(C_v, v^{-1}) = \frac{1}{4} FFT(C_v, -i)$$

$C_e = (20, -2)$, $C_o = (-5-i, -5+i)$

$FFT(C_e, -1) = (18, 22)$, $FFT(C_o, -1) = (-10, -2i)$

$FFT(C_v, -i) = (8, 22 + (-i)(-2i), 18 + (-1)(-10), 22 + i(-2i))$

$= (8, 20, 24, 24)$

$C = (2, 5, 7, 6)$

$C(x) = 2 + 5x + 7x^2 + 6x^3$

Problem 2 [DPV] Problem 2.16 (find x in an infinite array)

Answer: (Explain your algorithm in words and analyze its running time)

Find i where $A[2^i] = \infty$ by incrementing i. This takes $O(\log n)$. Search the array $A[1 \ldots 2^i]$ with binary search (compare center value, prune left or right half, etc.) This takes $O(\log n)$. Total time is $O(\log n)$.

Problem 3

You are given an array $A = [1, \ldots, n]$ of n positive numbers that represent the price of a particular stock on n days. You want to buy the stock at one day and sell on a later day. Your goal is to determine what would have been the best pair of days for buying/selling. You simply have to output the difference in price. You can assume n is a power of 2. Here is an example: $A = [10, 15, 6, 3, 7, 12, 2, 9]$. Then the optimal decision would be to purchase on day 4 for \$3 and sell on day 6 for \$12, so your algorithm should output \$9.

3. Part A:

Suppose you want to buy in the first $n/2$ days and you want to sell in the last $n/2$ days. Give an $O(n)$ time algorithm for finding the best pair of days for buying/selling under this restriction. Explain your algorithm in words and justify/explain its running time.

Answer:

Iterate through first n/2 days to find the min value, O(n). Iterate through last n/2 days to find the max value, O(n). Return the difference O(1). Total time is O(n).

3. Part B:

Give a divide and conquer algorithm with running time $O(n \log n)$ for finding the best pair of days for buying/selling. (There are no restrictions on the days, except that you need to buy at an earlier date than you sell.) Explain your algorithm in words and analyze your algorithm, including stating and solving the relevant recurrence.
Hint: Your algorithm should use your solution to part a.

Answer:

Split A into the first and last $n/2$ days.
Get solution from algorithm in part a.
Run the two sub arrays through the algorithm recursively and get the max differences from each. Return the max of $MaxProfit(A_F)$, $MaxProfit(A_L)$, $max(A_L) - min(A_F)$.

This Algorithm splits the problem into 2 $n/2$ subproblems each of $O(n)$ time. The recurrance is $T(n) = 2T(\frac{n}{2}) + O(n)$.
This runs in $O(n \log n)$ time.

3. Extra Credit:

Give a divide and conquer algorithm with running time $O(n)$ for finding the best pair of days for buying/selling.
Hint: Modify the problem to obtain additional info from the subproblems so that you can do the "merge" part in $O(1)$ time.

Answer:

MaxProfit(A):
If A is of size 2:
Return Min(A), Max(A), $A_1 - A_0$
Split A into A_F and A_L
$Min_F, Max_F, P_F = MaxProfit(A_F)$
$Min_L, Max_L, P_L = MaxProfit(A_L)$
Return $Min(Min_F, Min_B)$, $Max(Max_F, Max_L)$, $Max(P_L, P_F, Max_L - Min_F)$

By returning the min and max along with the max profit, a $O(n)$ search at each level is replaced with a $O(1)$ computation. This makes the recurrance $T(n) = 2T(\frac{n}{2}) + O(1)$ which has a runtime of $O(n)$.

Problem 4 Integer multiplication using FFT

4. Part A:

Given an n-bit integer number a where $a = a_0 a_1 \dots a_{n-1}$, define a polynomial $A(x)$ where $A(2) = a$.

Answer:

$$A(2) = a_0 2^{n-1} + a_1 2^{n-2} + \dots a_{n-2} 2 + a_{n-1}$$

$$A(x) = a_0 x^{n-1} + a_1 x^{n-2} + \dots a_{n-2} x + a_{n-1}$$

4. Part B:

Given 2 n-bit integers a and b, give an algorithm to multiply them in $O(n \log n)$ time. Use the FFT algorithm from class as a black-box (i.e. don't rewrite the code, just say run FFT on ...). Explain your algorithm in words and its running time.

Answer:

(1) Convert integers a and b into polynomial form of $A(x) = a_0 x^{n-1} + \dots a_{n-1} x^0$. This takes $O(n)$ time to iterate over each bit.

(2) Convert each polynomial to value form with FFT algorithm in $O(n \log n)$ time.

(3) Multiply the value form of each polynomial in $O(n)$ time.

(4) Convert back to the polynomial form with inverse FFT in $O(n \log n)$ time.

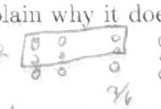
(5) Convert polynomial coefficients back to binary representation in $O(n)$ time.

Total running time is $O(n \log n)$

Problem 5 Deterministic Algorithm for finding the Median

For the deterministic $O(n)$ time algorithm for finding the median presented in class on Thursday, February 2, suppose we broke the array into groups of size 3 or 7 (instead of 5).
Does groups of 3 or 7 work? Why? Make sure to state the recurrence $T(n)$ for the modified algorithm for each case and explain why it does or does not solve to $O(n)$.

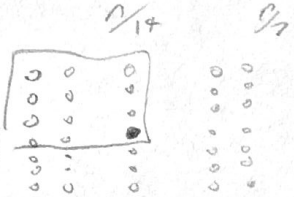
Answer:



When $|G_i| = 3$, $A_{\leq p} = \frac{2n}{6}$, so $A_{>p} = \frac{4n}{6} = \frac{2n}{3}$

The recurrence becomes $T(n) = T(\frac{2n}{3}) + T(\frac{n}{3}) + O(n)$

$\frac{2}{3} + \frac{1}{3} = 1$ so runtime $\neq O(n)$.



$|G_i| = 7$, $A_{\leq p} = \frac{4n}{14}$, $A_{>p} = \frac{10n}{14} = \frac{5n}{7}$

Recurrence is $T(n) = T(\frac{5n}{7}) + T(\frac{n}{7}) + O(n)$.

$\frac{5}{7} + \frac{1}{7} < 1$, so runtime is $O(N)$.