ADS- COURSEWORK 1 - s1725018

1) a) Solve the recurrence $T(n) = 3T(\sqrt{n}) + \lg n$.

Let $m = \lg n \leftrightarrow 2^m = n$

$$\therefore \quad T(n) = 3T(\sqrt{n}) + \lg(n)$$
$$= T(2^m) = 3T((2^m)^{1/2}) + m$$
$$= T(2^m) = 3T(2^{m/2}) + m$$

now let $S(m) = T(2^m)$

$$\therefore \quad T(2^m) = S(m) = 3S\left(\frac{m}{2}\right) + m$$

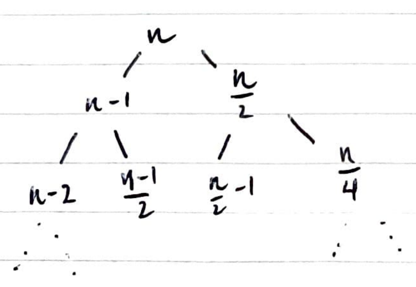
by using master theorem on $S(m) = 3S(m/2) + m$

we have $a = 3$ $\quad b = 2$ $\quad k = 1$

$c = \lg(3)$ since $c > 1$ then $S(m) = \Theta(m^{\lg 3})$

$$T(n) = T(2^m) = S(m) = \Theta(m^{\lg 3})$$
$$= \Theta((\lg n)^{\lg 3}) \quad \square$$

b) Proove the asymptotic upper bound on:
$T(n) = T(n-1) + T(n/2) + n$



It is clear to see that the worst case scenario is when both sides decrease by 1. $\therefore$ at each node we get 2^i elements. Since it decrease by one our height will be n so as a result at the leaf node we will have 2^n eleafs. as a result this is the worst case run time $O(2^n)$

Similarly the best case run time is when we exclusively divide by $n/2$ and so our tree will have by letting $n = 2^k \to 2^k$ $\therefore$ $k = \lg n$.

$$\begin{aligned}
T(2^k) &= T\left(\tfrac{2^k}{2}\right) + 1 \\
&= T(2^{k-1}) + 1 \\
&= (T(2^{k-2} + 1)) + 1 \\
&= T(2^{k-2}) + 2 \\
&= \vdots \\
&= T(2^{k-k}) + k \\
&= T(2^0) + k \\
&= T(1) + k \\
&= 1 + \lg n
\end{aligned}$$

$\therefore T(n) = O(1 + \lg n) = O(\lg n)$. $\square$

c) we have that $T(n) = a\,T(n/4) + \Theta(n^2)$

in the worst case by the master theorem, $T(n) = O(n^{\log_4 a})$

$$\begin{aligned}
\therefore \quad \log_4 a &< \log_2 7 \\
\log_2 \sqrt{a} &< \log_2 7 \\
\sqrt{a} &< 7 \\
a &< 49
\end{aligned}$$

(so it is better than Strassen's)

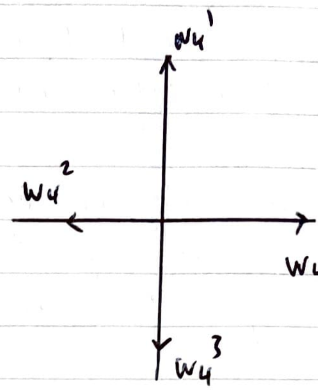
$\therefore$ a can be at most $\underline{48}$. $\square$

2) a) Compute DFT of vector $(0, 1, 2, 5)$

$DFT_4(0, 1, 2, 5)$

$A(x) = 0x^0 + 1x^1 + 2x^2 + 5x^3$
$A(x) = x + 2x^2 + 5x^3$

$w_4^0 = (e^{2\pi i/4})^0 = e^0 = 1$
$w_4^1 = (e^{2\pi i/4})^1 = e^{\pi i/2} = i$
$w_4^2 = (e^{2\pi i/4})^2 = e^{\pi i} = -1$
$w_4^3 = (e^{2\pi i/4})^3 = e^{\frac{3\pi i}{2}} = -i$



$A(w_4^0) = 1 + 2(1)^2 + 5(1)^3 = 1 + 2 + 5 = 8$
$A(w_4^1) = i + 2(i)^2 + 5(i)^3 = i - 2 - 5i = -2 - 4i$
$A(w_4^2) = -1 + 2(-1)^2 + 5(-1)^3 = -1 + 2 - 5 = -4$
$A(w_4^3) = (-i) + 2(-i)^2 + 5(-i)^3 = -i - 2 + 5i = -2 + 4i$

$DFT_4((0, 1, 2, 5)) = (8, -2-4i, -4, -2+4i)$

b) We know that $A(x) = q(x)(x - x_0) + r$ and so as a result we can calculate r by horner's method in $O(n)$ time. We only need to calculate the coefficients of $q(x)$

$A(x) = q(x)(x - x_0) + r$

$a_0 + a_1 x^1 + \dots + a_n x^n = (b_0 + b_1 x^1 + \dots + b_{n-1} x^{n-1})(x - x_0) + r$

by construction: we have:

$a_0 + a_1 x^1 + \dots + a_n x^n = b_0 x + b_1 x^{1+1} + \dots + b_{n-1} x^n - x_0(b_0 + b_1 x^1 + \dots + b_n x^{n-1}) + r$

$= a_0 + a_1 x^1 + \dots + a_n x^n = b_0 x + b_1 x^2 + \dots + b_{n-1} x^n - x_0 (q(x)) + r$

as a result we can see that:

$a_0 = -x_0 b_0 + r$
$a_1 = b_0 - x_0 b_1$
$a_2 = b_1 - x_0 b_2$
$\vdots$
$a_{n-1} = b_{n-2} - x_0 b_{n-1}$
$a_n = + b_{n-1}$

$\therefore \quad b_0 = \dfrac{r - a_0}{x_0} \quad$ and $\quad a_n = b_{n-1}$

and $\quad b_i = \dfrac{b_{i-1} - a_i}{x_0}$

which we can calculate directly for $i = 2$ upto n and as a result this takes $O(n)$ time. We can do this since we have a_i, x_0 and r.

$\therefore$ total time would be $O(n)$ for evaluating r and $O(n)$ for the coefficients $\therefore \quad O(n) + O(n) = O(n)$ $\square$

c) Since $P(x) = 0 \quad \forall \, z_0, z_1 \dots z_{n-1}$

We can construct a polynomial of degree $n-1$ with $(x - z_0)(x - z_1) \dots (x - z_{n-1}) = Q(x)$

$\therefore$ we know $P(x) = Q(x) + r(x)$ $r(x)$ of degree 2.

$\therefore$ by polynomial division $P(x) / Q(x)$ using FFT we can get $r(x)$ in $O(n \lg n)$ time.

now we need to solve $Q(x)$ to find the coefficients of $A(x)$

since $Q(x) = (x - z_0)(x - z_1) \cdots (x - z_{n-1})$

we can use fft to ~~pro~~ multiply it out ~~gi~~ using polynomial multiplication where we use a divide and conquer strategy to recursively split the polynomial into 2 and multiply the resulting polynomial with FFT

$\therefore$ Our recurrence looks like this:

$$T(n) = 2T(n/2) + O(n \lg n)$$

by master theorem $a = 2$ $b = 2$ $c = \log_2(2) = 1$

since $c = k$ of n $\therefore$ our recurrence has

$O(n \lg^{k+1} n)$ complexity $= O(n \lg^2 n)$

which then we can use to read coefficients of ~~[illegible]~~ the polynomial so total time $= O(n \lg^2 n) + O(n \lg n)$

$= O(n \lg^2 n)$ $\square$

d) let $A(x) = \sum_{i=0}^{10n} a_i x^i$ where i is the value between 0 and $10n$

~~a~~ $= a\_i =$ n° of times value i occurred and $i =$ value

likewise for $B(x) = \sum_{i=0}^{10n} b\_i x^i$

by using FFT to multiply $A(x)$ and $B(x)$ we get it done in $O(n \log n)$ time.

where the coefficients of this new function are the number of times each element occurs and the power is the element

This is due to properties of matrix multiplication.

Since $A(x) \cdot B(x) = \sum_{i=0}^{2n}$ for example the first term would be:

$b_0 \cdot a_0 x^{0+0}$ ← value in c.

n° of times it occurs in C

$\square$