COMP9101 ASS2

Z5141180 YIZHENG YING

1. $P_A(x) = A_0 + A_3x^3 + A_6x^6$ and $P_B(x) = B_0 + B_3x^3 + B_6x^6 + B_9x^9$, so we assume that $y = x^3$, so we can get $P_A(x) = A_0 + A_3y + A_6y^2$ and $P_B(x) = B_0 + B_3y + B_6y^2 + B_9y^3$ so P_A is degree 2 and P_B is degree 3, it is able to multiply any degree 2 polynomial by a degree 3 polynomial using 6 such multiplications. Since the result is a polynomial of degree 5, we can uniquely determine it by determining its values at 6 points. For this we can choose $\{-3,-2,...,2\}$. We evaluate P at these 6 points and Q at these 6 points. We then multiply the results pointwise: these require precisely 6 large number multiplications. We can then determine the coefficients from these values by setting up a system of linear equations. Solving this is done by inverting a constant matrix, so this inversion can even be done by hand. We then multiply the matrix by the vector formed by the pointwise multiplications.

2. (a) $(a+ib)(c+id) = ac + iad + ibc - bd$

$ad + bc = (a+b)(c+d) - ac - bd$

$(a+ib)(c+id) = ac + ((a+b)(c+d) - ac - bd)i - bd$

Because we have already got ac and bd before, so we just need to multiply (a+b) and (c+d). So we can multiply two complex numbers using only 3 real number multiplications with

$ac + ((a+b)(c+d) - ac - bd)i - bd$

(b) $(a+ib)^2 = a^2 - b^2 + 2abi = (a+b)(a-b) + 2abi$, we just need to calculate (a+b)(a-b) and ab, so we can sue only 2 multiplications of real numbers.

(c) Because $ac + ((a+b)(c+d) - ac - bd)i - bd$ we use 3 multiplications, so

$(a+ib)^2(c+id)^2 = (ac + ((a+b)(c+d) - ac - bd)i - bd)^2$,

We can assume $(ac - bd) + ((a+b)(c+d) - ac - bd)i$ as

$(A+Bi)^2$, as we get in question b we only need to use 2 multiplications, so in total we can use 5 real number multiplications.

3. (a) $P_A(x) = A + A_1x^1 + ...A_nx^n$ to calculate DFT we use O(nlogn) times, we can get $\mathrm{P_A}' = \{\mathrm{P_A}(1), \mathrm{P_A}(\omega_{2n+1})...\mathrm{P_A}(\omega_{2n+1}^{2n})\}$, the same $P_B(x) = B + B_1x^1 + ...B_nx^n$ we can get

$\mathrm{P_B}' = \{\mathrm{P_B}(1), \mathrm{P_B}(\omega_{2n+1})...\mathrm{P_B}(\omega_{2n+1}^{2n})\}$ in O(nlogn) times. Then we multiply $\mathrm{P_A}'$ and $\mathrm{P_B}'$ in O(n) times to get

$\mathrm{P_C} = \{\mathrm{P_A}(1)\mathrm{P_B}(1), \mathrm{P_A}(\omega_{2n+1})\mathrm{P_B}(\omega_{2n+1})...\mathrm{P_A}(\omega_{2n+1}^{2n})\mathrm{P_B}(\omega_{2n+1}^{2n})\}$, then we use IDFT in O(nlogn) times to get

$P_c'(x) = \sum_{j=0}^{2n}(\sum_{i=0}^{j} A_i B_{j-i})x^j = \sum_{j=0}^{2n} c_j x^j = P_A(x)P_B(x)$, so we can calculate convolution in time O(n log n).

(b)(i) We suppose that each $P_i (1 \le i \le K)$ has S items, if there is no item, the coefficient of the item will be 0. We make each P_i represent by point value and the complexity is O(SlogS). Then multiply the results pointwise and then IDFT. So the total complexity is O(KSlogS).

(ii) We arrange S polynomials present as leaves of a balanced full tree, i.e., a tree in which every node has either 2 or 0 children and the depth of the tree is as small as possible. To do that compute $\mathrm{m} = \lfloor \log n \rfloor$ and construct a perfect binary three with $2^m \le S$ leaves. If $2^m < S$ add two children to each of the leftmost $S - 2^m$ leaves of such a perfect binary tree. We can get $\sum (d(p_i) + d(p_{i+1})) \log(d(p_i) + d(p_{i+1})) \le \sum (d(p_i) + d(p_{i+1})) \log(S)$ $\le S \log S$,so the complexity is O(SlogS) and there are logK layer of the tree so the total complexity is O(SlogKlogS).

4. (a) We assume that when n=1, we have $A_1 =$

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

When n=k, we have $A_k =$

$$\begin{bmatrix} F_{k+1} & F_k \\ F_k & F_{k-1} \end{bmatrix}$$

So when n=k+1,we have $A_{k+1} = A_1 A_k$

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} F_{k+1} & F_k \\ F_k & F_{k-1} \end{bmatrix}=\begin{bmatrix} F_{k+1}+F_k & F_k+F_{k-1} \\ F_{k+1} & F_k \end{bmatrix}=\begin{bmatrix} F_{k+2} & F_{k+1} \\ F_{k+1} & F_k \end{bmatrix}$$

In conclusion, for all integers n $\geqslant$ 1, the equation is right.

(b) If n is odd, we can get $A^n = AA^{2\frac{n-1}{2}}$, we can calculate $F(\frac{n-1}{2}), F(\frac{n-1}{4})...F(1)$ and multiply them.

If n is even we can get $A^n = (A^2)^{\frac{n}{2}}$, we can calculate $F(\frac{n}{2}), F(\frac{n}{4})...F(1)$ and multiply. The complexity is O(logn).

So the total complexity is O(logn).

(a) We compare $H[i]_{(0\le i<N)}$ with T and intial count=0. If $H[i] < T$ then we compare next one. If $H[i] \ge T$, we make count=count+1, and compare the H[i+K+1] one because every pair of leaders must have at least K $\geqslant$ 0 giants standing in between them. After comparing the last one , if count $\geq$ L, there exists some valid choice of leaders satisfying the constraints, otherwise there does not exist and the complexity is O(N).

(b)Firstly we do merge sort to make the list into a non-

decreasing order.

The second step we can do binary search and then we use the function in question (a).

Then we do the second step recursively.

So the total complexity is O(NlogN).