COMP 9101

Assignment 2

Student ID : z5174741

Student name : GUANQUN ZHOU

Q1

And

We can let and in this way we can get another two polynomials:

And

We let

The degree of is 2 and the degree of is 3.

So the degree of should be and we can use 6 coefficients to represent

In other word,

In this way, we can multiply those two polynomials using only 6 large number multiplications.

We can use 6 small numbers, for example, -3,-2,-1,0,1,2 , to evaluate all coefficients of .

If we just calculate the product of :

We can also get the values of < and multiply those two polynomials using only 6 large number multiplications.

Q2

1. We can multiply and we can get:

We can find that , which means if we already know the value of products of , then we can get the value of with only one real number multiplications.

So the three real number multiplications we need to multiply are

.

1. We can calculate and we can get:

And we can find that

The product of could be equal to

So the two multiplications of real numbers we need to calculate are .

1. We can let

We can find that .

According to conclusion of (a), we can multiply with only three real number multiplications. After calculation, we can get a new complex number, suppose we let the new complex number is .

So now and our job is to calculate the result of

According to conclusion of (b), we can calculate using only ﬁve real number multiplications.

So in total, we can find the product using only ﬁve real number multiplications.

Q3

1. Assume two n-degree polynomials are:

And

We let .

The degree of both and is n, so the degree of is . Thus we will be able to use distinct coefficients to uniquely determine .

First we calculate DFT for every polynomials using the roots of unity.

Since we need distinct coefficients to uniquely determine , the DFT of and should be like :

And

We use FFT to evaluate these DFT, and this can be done in .

The second step is to evaluate the multiplication of

and

The product is

And this can be done in

Third step is to evaluate with IDFT, and this can be done in .

Thus, the product of and can be compute in with the Fast Fourier Transform (FFT)

1. We are given K polynomials and deg()+…+deg() = S

In order to ﬁnd the product of these K polynomials, we can compute them in this way:

We recursively multiply every with the product of previous calculation and in every time we multiply with the product of previous calculation, since degree of and degree of the product of previous calculation are both less than S, so we can say that every multiplication can be done in using FFT.

As we need to compute K polynomials, which means we need to do multiplications to get the product of these K polynomials. multiplications can be done in and .

Thus, we can say that the product of these K polynomials can be found in

1. We are given K polynomials and deg()+…+deg() = S

We can put every polynomials and their relative products in a complete binary tree.

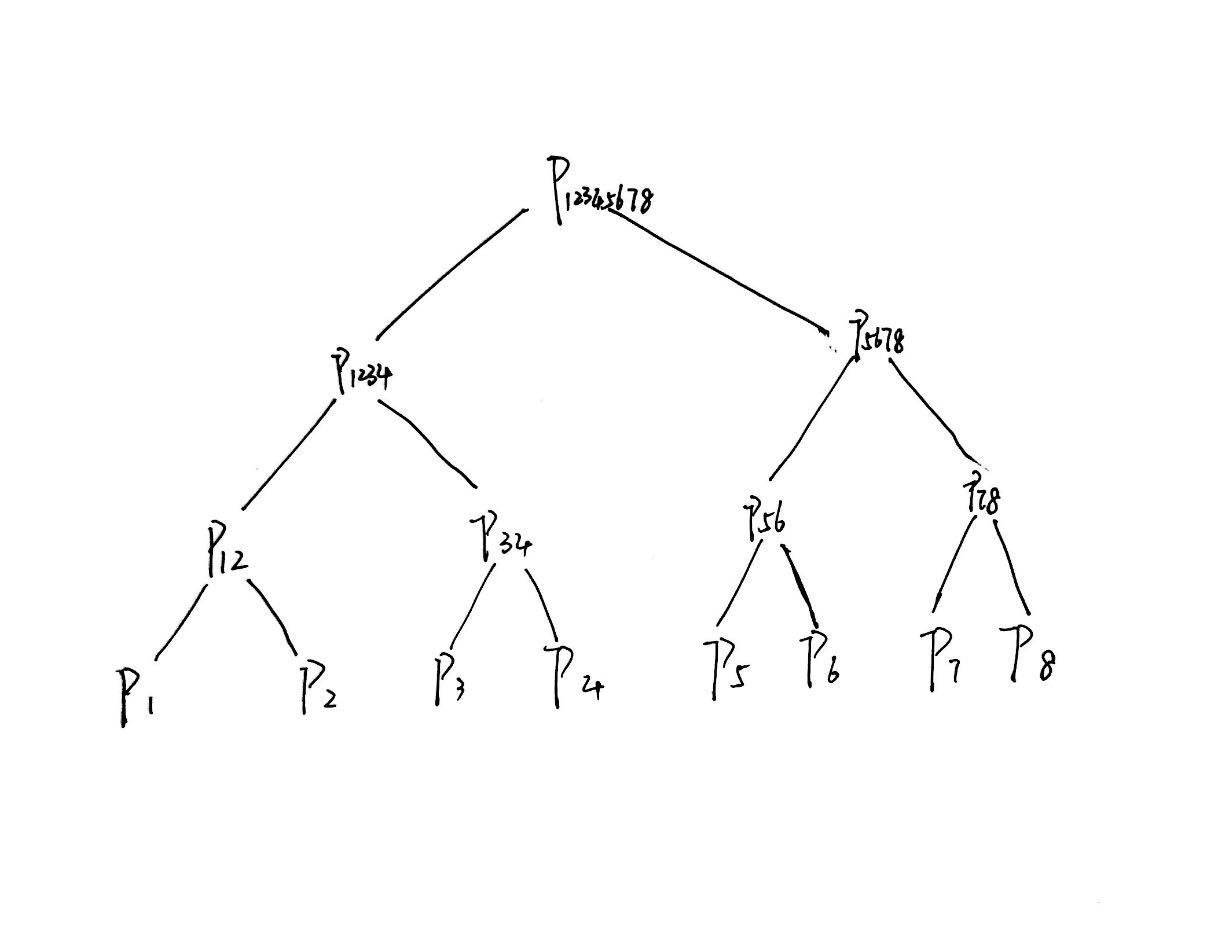
First step : we need to make sure all polynomials are assigned to one leaf. We need to compute the value of , we let . If , then we can build a complete binary tree directly . If , then we need to add two children to the left most leaves. In this way, we can build a complete binary tree in any condition.

Second step: If , we can treat the two lowest levels as one level, because in this way, the amount of the degrees of polynomials on each level of the binary tree is always equal to S. If , the amount of the degrees of polynomials on each level of the binary tree is always equal to S.

We let to represent the degree of any two children polynomials of a node at some level. The degree of product of these two polynomials is . So we can compute the two polynomials in .

Since , so , thus we can say these two children polynomials can be calculated in . In this way, the amount tine of computing all pairs of polynomials on this level could be because the amount of the degrees of polynomials on each level of the binary tree is always equal to S.

Third step: The height of the tree would be or . According to second step, the calculation for every level can be done in . And no matter height of the tree would be or , the whole calculation can be done in .



The graph above can show an example.

Suppose we have 8 polynomials and the amount of the degrees of polynomials is equal to S.

According to the method mentioned in (ii), we can compute every level in and the height of the tree is 3 which is . So this algorithm can help us to solve this problem in

Q4

1. Step1. we compute when :

Step2. we assume the equation is true for n = k (),

Step3. We will prove n = k+1 can satisfy the equation:

In this way, given formula can be proved for all integers .

1. According to (a), , then we find via computing

We can let

Then we can proceed with divide and conquer.

. . .

. . .

This graph shows how to compute with divide and conquer.

If n is even, we just recursively calculate and square it. Every time of recursive calculation can be done in .

If n is odd, we just recursively calculate ,then square it and time it with a . Every time of recursive calculation can also be done in .

We can see from the graph, the height of tree is , so we need to compute times and every time calculation can be done in

Thus, we can find in time.

Q5

1. First step: we need a variety to record the number of elements which are no less than T. the initial value of  is zero.

Second step: We need to find the first element which is no less than T. we let the index of this element is . Then we plus with 1.

Third step: Then we need to check the element whose index is (.

If the value of this element is no less than T, we plus with 1 and update with the index of this element . Then we continue to skip giants to check the new element whose index is new (.

If the value of element is less than T, then we need to check the next neighbour of element . In other word, we will check the element whose index is ( If the value neighbour is less than T, then we continue to plus current index with 1 until we found next element whose value is no less than T. Then we plus with 1 and update with the index of this element whose value is no less than T. Then we continue to skip giants to check the new element whose index is new (.

We use method in third step to check over the entire list . If ,return , which means there exists some valid choice of leaders satisfying the constraints whose shortest leader has height no less than T. If , return , which means there is no any valid choice of leaders satisfying the constraints whose shortest leader has height no less than T.

In this way, the whole work can be done in .

1. The optimisation version of this problem is to find the largest T which can return according to the decision version of this problem.

Since if one T can return in the decision version of this problem, which mean that all candidates whose value is less than T will all return in the decision version of this problem.

In other word, the result of the decision version of this problem is monotonously relative to T.

First step: we sort list using Merge sort and this can be done in .

Second step: we use binary search to find the optimal T.

We let initial value of = 0(which is the index of the first element of list ) and initial value of = the largest index of list .

Then we recursively compute: , if for the decision version of this problem returns , then . if for the decision version of this problem returns then .

In this way, we can find the optimal T.

Since binary search can be done in and for every the decision version of this problem need compute in .

Thus, the whole work can be done in .