VE 477 Homework8

Wang Yichao, ID: 517370910011

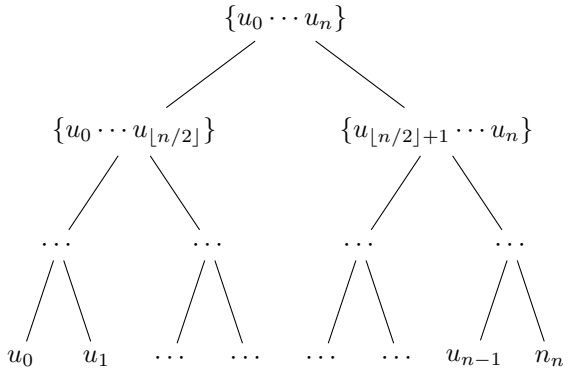
- **Exercise 1.**

  This problem comes from a paper Fast Multipoint Evaluation On n Arbitrary Points and we need to learn FastEval algorithm.

  `http://www.cecm.sfu.ca/CAG/theses/justine.pdf`

  Part 1:

  1. The split is similar to merge sort.



  2. It is obvious that when subscript is 0 $M_{0,j} = m_{j+0} = m_j$.

When subscript is greater than 0,

$$\begin{aligned} M_{i+1,j} &= \prod_{l=0}^{2\cdot 2^i-1} m_{j\cdot 2^i+l} \\ &= \prod_{l=0}^{2^i-1} m_{j\cdot 2^i+l} \cdot \prod_{l=2^i}^{2\cdot 2^i-1} m_{j\cdot 2^i+l} \\ &= \prod_{l=0}^{2^i-1} m_{j\cdot 2^i+l} \cdot \prod_{l=0}^{2^i-1} m_{j\cdot 2^i+2^i+l} \\ &= M_{i,2j} \cdot M_{i,2j+1} \end{aligned}$$

  3. Since in tree the children of i is $2i$ and $2i+1$, we can learn that parent is the product of its two children.

  4. (a) The basic idea is using the product property and build this tree from bottom to the top.

Algorithm 1 Build Tree

Input: an integer $n = 2^k$, $u_0, \cdots, u_{n-1}$ in commutative Ring R
Output: The polynomials $M_{i,j}$ for $0 \le i \le k$ and $0 \le j < 2^{k-i}$

```
for j = 0, ..., n - 1 do
    M_{i,j} = x - u_i
end for
for i = 0, ..., k do
    for j = 0, ..., 2^{k-i} - 1 do
        M_{i,j} = M_{i-1,2j} × M_{i-1,2j+1}
    end for
end for
```

  (b) Use divide and conquer.

Algorithm 2 *Recursive*

Input: an integer $n = 2^k$, $f \in R[x]$ of degree less than n, $u_0, \cdots, u_{n-1}$ in commutative Ring R, and polynomials $M_{i,j}$

Output: $f(u_0), \ldots, f(u_{n-1})$

1: **if** $n == 1$ **then**
2: return f
3: **end if**
4: $r_0 \equiv f \mod M_{k_1,0}$
5: $r_1 \equiv f \mod M_{k_1,1}$
6: $r_0(u_0), \ldots, r_0\left(u_{n/2-1}\right) = Recursive(n/2, r_0, subtree\ rooted\ at\ M_{k_1,0})$
7: $r_1\left(u_{n/2}\right), \ldots, r_1(u_{n-1}) = Recursive(n/2, r_1, subtree\ rooted\ at\ M_{k_1,1})$
8: **return** $r_0(u_0), \ldots, r_0\left(u_{n/2-1}\right), r_1\left(u_{n/2}\right), \ldots, r_1(u_{n-1})$

5. (a) We have already proved that parent is the product of its two children. For $k = 0$, it is trivial since it is constant function and it will return f. Now we assume this is true for k and we need to prove that it is also true for $k+1$. We now do division with remainder. In the *Recursive* algorithm, for $0 \le i < n/2$, we have $f(u_i) \equiv q_0(u_i) \cdot M_{k-1,0}(u_i) + r_0(u_i) \equiv r_0(u_i)$. For $n/2 \le i < n$, we have $f(u_i) \equiv q_1(u_i) \cdot M_{k-1,1}(u_i) + r_1(u_i) \equiv r_1(u_i)$. Qed.

(b) For building the tree, the time complexity would be $\frac{n}{2}\mathrm{M}(2) + \frac{n}{4}\mathrm{M}(4) + \cdots + 4\mathrm{M}\left(\frac{n}{4}\right) + 2\mathrm{M}\left(\frac{n}{2}\right) = n \log n^2 = \log n M(n)$.

For *Recursive* algorithm, we have $T(n) = 2T(n/2) + 2D(n)$, where $D(n)$ is $\mathcal{O}(n \log n)$. Using master theorem, we can get the complexity of $T(n) = \mathcal{O}(n \log n^2) = \mathcal{O}(M(n) \log n)$.

Part 2.

1. Strict definition: Reference: `https://www.math.usm.edu/lambers/mat772/fall10/lecture5.pdf`

Readable one: Reference: `https://brilliant.org/wiki/lagrange-interpolation/`

Briefly, if we are given n discrete points, Lagrange interpolation can be used to determine a polynomial that cross these n points, which is fitting.

2. Since $(uv)' = u'v + uv'$, we can get

$$m' = m \times \sum_{j=0}^{n-1} \frac{(x - u_j)'}{x - u_j} = m \times \sum_{j=0}^{n-1} \frac{1}{x - u_j}$$

(where, $x \notin \{u_0, \cdots, u_{n-1}\}$).

When $x = u_i$, only the ith term remains, since for other terms in the summation, m will be 0. Thus $m'(u_i) = 1/s_i$.

3. Use the similar divide and conquer algorithm as in part 1 question 4.

Algorithm 3 *Divide*

Input: $u_0, \cdots, u_{n-1}$, P

Output: the interpolation of P at the points $u_0, \cdots, u_{n-1}$

1: **if** *degree* $== 1$ **then**
2: **return** P
3: **end if**
4: $r_0 \equiv P \mod M_{k_1,0}$
5: $r_1 \equiv P \mod M_{k_1,1}$
6: $r_0(u_0), \cdots, P(u_n/2) = Divide(n/2, r_0, subtree\ rooted\ at\ M_{k_1,0})$
7: $r_1(u_0), \cdots, P(u_n/2) = Divide(n/2, r_1, subtree\ rooted\ at\ M_{k_1,1})$
8: **return** $r_0(u_0), \ldots, r_0\left(u_{n/2-1}\right), r_1\left(u_{n/2}\right), \ldots, r_1(u_{n-1})$

4. (a) Optional and not interested.

(b) Similar to part 1, to build up the tree, we need $n \log n^2 = \log n M(n)$.

(c) Similar to part 1, for the recursion, we have $T(n) = 2T(n/2) + 2D(n)$, where $D(n)$ is $\mathcal{O}(n \log n)$. Using master theorem, we can get the complexity of $T(n) = \mathcal{O}(n \log n^2) = \mathcal{O}(M(n) \log n)$. SO it can be solved within $\mathcal{O}(M(n) \log n)$ operations.

5. I do not think so, since it will require $\mathcal{O}(n^2)$ polynomial space complexity, which is really costly when more points are chosen for fitting. Space is more expensive than time. However, we can do pre-calculation when the size is small to save time.

- **Exercise 2.**

  1. Life will be better if we focus on theory instead of coding.

  Since $xyxy = yxyx$, assume $xy = m$, we will get $mm = ymx = x^{-1}mmx$, that is $xmm = mmx$ for any x and m, we denote this equation as lemma.

  Next we prove that $(xyx^{-1}y^{-1})^4 = e$. We rewrite it as

  $(xyx^{-1}y^{-1})^2(xyx^{-1})y^{-1}xyx^{-1}y^{-1}$
  $= (xyx^{-1})(xyx^{-1}y^{-1})^2y^{-1}xyx^{-1}y^{-1}$ (by taking $xyx^{-1}y^{-1}$ as m in lemma)
  $= xy^2x^{-1}y^{-1}xyx^{-1}(y^{-1})^2xyx^{-1}y^{-1}$ (then we take advantage of two squares in this line)
  $= y^2xx^{-1}y^{-1}xyx^{-1}(y^{-1})^2xyx^{-1}y^{-1}$
  $= yxyx^{-1}(y^{-1})^2xyx^{-1}y^{-1}$
  $= yxyx^{-1}x(y^{-1})^2yx^{-1}y^{-1}$
  $= e$.

  Finally since $(xyx^{-1}y^{-1})^4 = e$, we can get $(xyx^{-1}y^{-1})^2 = e$, then $(xyx^{-1}y^{-1}) = e$, which means $xy = yx$. Qed.

  2. Talk is cheap, show me the code. I reckon that it is impossible to calculate this question by hand. I am really curious how can our smart ATIJ solve this problem by hand.

  From what S_2 says, there are at least two ways to divide the xy. So this problem is about factorization. From what S_1 says, we can learn that there is no way to divide $x+y$ into to numbers such that the multiple of them has only one way of factorization into two integer greater than 1. Now we assume $x + y = m$ and $x \times y = n$. Then according to our reasoning. $n = p_1 \times p_2 \times q$ and $m = prime + composite$ or $m = composite + composite$. There is only one way for n to be divided.

  I use brute force to solve this problem in python. the runtime is OK for small input like this, but further **optimization** is required for larger input!!!

  type **./run.sh** in terminal, the answer will be redirected to stdout.txt the answer is in the last line, all the lines above are the key processes

- **Exercise 3.**

  Swype keyboard is a special way of typing English letters(also support other languages). Instead of pressing each letter once a time, you draw fold lines between each letter you want to type in and you do not need to move your fingers off the screen. There is an advanced algorithm to solve the accuracy problem since the path of your finger is hard to interpret directly. This Swype service is not available after 2018. My roommate is lucky to have tried this technology before on his phone.

  Reference: `https://www.youtube.com/watch?v=SJ-RAefCG_c`

- **Exercise 4.**

  I have completed the course survey.