UM-SJTU JOINT INSTITUTE

Introduction to Algorithms
(VE477)

Homework #8

Prof. Manuel

Xinmiao Yu
518021910792

Dec. 4, 2020

Q1.

Fast multi-point evaluation

1. The binary tree by splitting U is

$$u_0, \cdots, u_{n-1}$$

$$u_0, \cdots, u_{n/2-1} \qquad u_{n/2}, \cdots, u_{n-1}$$

$$u_0, \cdots, u_{n/4-1} \quad u_{n/4}, \cdots, u_{n/2-1} \quad u_{n/2}, \cdots, u_{3n/4-1} \quad u_{3n/4}, \cdots, u_{n-1}$$

... ...

2. From $M_{i,j} = \prod_{l=0}^{2^i-1} m_{j2^i+l}$

$$\begin{aligned} M_{i+i,j} &= \prod_{l=0}^{2^{i+1}-1} m_{j2^{i+1}+l} \\ &= \prod_{l=0}^{2^i-1} m_{j2^{i+1}+l} \cdot \prod_{l=2^i}^{2^{i+1}-1} m_{j2^{i+1}+l} \\ &= \prod_{l=0}^{2^i-1} m_{2j2^i+l} \cdot \prod_{l=0}^{2^i-1} m_{(2j+1)2^i+l} \\ &= M_{i,2j} \cdot M_{i,2j+1} \end{aligned}$$

$$M_{0,j} = \prod_{l=0}^{2^0-1} m_{j2^0+l} = m_j$$

3. The polynomial $M_{i,j}$ represents that for the node at height $k-i$, j nodes from left, the product of all the leaves underneath. As each leaf represents $M_{0,j} = x - u_j$.

$$M_{k,0} \prod_{i=1}^{n-1}(x-u_i)$$

$$M_{k-1,0} \prod_{i=1}^{n/2-1}(x-u_i) \qquad M_{k-1,1} \prod_{i=n/2}^{n-1}(x-u_i)$$

... ...

$M_{0,0} = (x-u_0)$ ... ... $M_{0,n-1} = (x-u_{n-1})$

4. a)

Algorithm 1: Build up the subproduct tree

Input : $n = 2^k$, $k \in \mathbb{N}$. $u_0, \dots, u_{n-1} \in R$
Output: $M_{i,j}$ for $1 \le i \le k$ and $0 \le j < 2^{k-i}$

for $j \leftarrow 0$ **to** $n-1$ **do**
  $M_{0,j} \leftarrow (x - u_i)$
end for
for $i \leftarrow 1$ **to** k **do**
  for $j \leftarrow 0$ **to** $2^{k-i} - 1$ **do**
    $M_{i,j} \leftarrow M_{i-1,2j} \cdot M_{i-1,2j+1}$
  end for
end for
return $M_{i,j}$

b)

Algorithm 2: Go down the subproduct tree

Input : $n = 2^k$, $k \in \mathbb{N}$. $P \in R[x]$ of degree less than n. $u_0, \dots, u_{n-1} \in R$. subproducts $M_{i,j}$
Output: $P(u_0), P(u_1), \dots, P_{u_{n-1}} \in R$

1 **if** $n == 1$ **then**
2   **return** $P \in R$
3 **end if**
4 $A_0 \leftarrow P \bmod M_{k-1,0}$
5 $A_1 \leftarrow P \bmod M_{k-1,1}$
6 $A_0(u_0), \dots, A_0(u_{n/2-1}) \leftarrow$ call algorithm with input $A_0, n/2$, subtree rooted at $M_{k-1,0}$
7 $A_1(u_{n/2}), \dots, A_1(u_{n-1}) \leftarrow$ call algorithm with input $A_1, n/2$, subtree rooted at $M_{k-1,1}$
8 **return** $A_0(u_0), ..., A_0(n/2-1), A_1(u_{n/2}), ..., A_1(u_{n-1})$

5. a) Proof by induction

When $k = 0$, **Algorithm 2** will return at line 2 directly,which satisfies the hypothesis.

When $k \ge 1$, evaluate P at u_i. Assume q_0 as the quotient of P divided by $M_{k-1,0}$ and q_1 be the quotient of P divided by $M_{k-1,1}$.

Then it gives

$$P(u_i) \begin{cases} q_0(u_i) \cdot M_{k-1,0}(u_i) + A_0(u_i) = A_0(u_i) & \text{if } 0 \le i < n/2 \\ q_1(u_i) \cdot M_{k-1,1}(u_i) + A_1(u_i) = A_1(u_i) & \text{if } n/2 \le i < n \end{cases}$$

b)

$$T(n) = 2T(n/2) + O(M(n))$$

From Master Theorem, the time complexity is $\mathcal{O}(M(n)\log n)$

Problem 1 continued on next page...

Fast interpolation

1. Give distinct $u_0, \ldots, u_{n-1} \in R$ and the arbitrary $v_0, \ldots, v_{n-1}$ in a field F, Lagrange interpolation says that the unique polynomial $P \in F[x]$ which solves the

$$P(u_0), \ldots, P(u_{n-1}) = (v_0, \ldots, v_{n-1})$$

takes the form

$$P = \sum_{0 \le i < n} v_i s_i \frac{m}{(x - u_i)},$$

where $m = (x - u_0) \cdots (x - u_{n-1})$ and

$$s_i = \prod_{j \ne i} \frac{1}{u_i - u_j}$$

2.

$$m^{'} = \left(\prod_{i=0}^{n-1} (x - u_i) \right)^{'} = \sum_{i=0}^{n-1} (x - u_i)^{'} \frac{m}{x - u_i} = \sum_{i=0}^{n-1} \frac{m}{x - u_i}$$

$\frac{m}{x-u_i}$ vanishes at all points u_j with $i \ne j$, so

$$m^{'}(u_i) = \frac{m}{x - u_i}|_{x=u_i} = \frac{1}{s_i}$$

3. Algorithm

Algorithm 3: Interpolation

Input : $n = 2^k$, $k \in \mathbb{N}$. $u_0, \ldots, u_{n-1} \in R$, subproducts $M_{i,j}$ for $1 \le i \le k$ and $0 \le j < 2^{k-i}$

Output: $P(u_0), P(u_1), \ldots, P_{u_{n-1}}$

if $n == 1$ **then**
  return $P \in R$
end if
$left \leftarrow$ call algorithm with input $k - 1, 2i$
$right \leftarrow$ call algorithm with input $k - 1, 2i + 1$
return $left \times M_{k,1} + right \times M_{k,0}$

4. a)

   b) s_i could be computed by evaluating $m^{'}$ at the n evaluation points $u_0, \ldots, u_{n-1}$. which could be done in $\O(M(n)\log n)$.

   c) Once the s_i is computed, the polynomial to solve the problem is created. To output a polynomial of the form $P(x) = a_0 + a_1 x + \ldots a_{n-1} x^{n-1}$, the subproduct tree will be used to compute the sum. Then from the previous discussion, we could know the time complexity is $\O(M(n)\log n)$.

5. It is possible because we could retrieve polynomials from the precomputed tree.

Q2.

1.

2. - S_2 :"No idea what those two numbers could be..."
     The only way of taking xy student to know the two numbers directly is that the product could only be decomposed into two prime numbers. Such as $15 = 3 \times 5$. Then conclude that xy is not a unique result of two prime numbers.
   - S_1 : "I'm not surprised, I knew you couldn't know!"
     For student taking $x + y$, he knows the only way for another student to know the two numbers is two prime numbers product. Then conclude that $x + y$ could not be decomposed into two prime numbers. Then all the even numbers could be ignored. The remaining odd numbers that satisfy this condition could be generated by program: $11, 17, 23, 27, 29, 35, 37, 41, 47, 53$
   - S_2 :"Uhm...so now I know..."
     At this points, S_2 knows $x + y$ must be in the above list.
   - S_1 : "So do I!"
     For S_1 to know the number, the sum must only corresponds to only one possible solution.

   Then $x = 4, y = 13$.