CS6033 – Design and Analysis of Algorithms I

Homework 8

Manuel — NYU (Spring 2021)

Reminders

- Write in a neat and legible handwriting or use LaTeX
- Clearly explain the reasoning process
- Write in a complete style (subject, verb, and object)
- Be critical on your results

*Questions preceded by a * are optional. Although they can be skipped without any deduction, it is important to know and understand the results they contain.*

Ex. 1 — *Fast multi-point evaluation and interpolation*

Let R be a commutative ring, $u_0, \cdots, u_{n-1}$ be n elements in R, and $m_i = X - u_i$, with $0 \leq i < n$, be n degree 1 polynomials in $R[X]$. Without loss of generality we assume n to be a power of 2.

Part I — Fast multi-point evaluation

In order to perform fast multi-point evaluation the set of points $U = \{u_0, \cdots, u_n\}$ is recursively split into two halves of equal cardinality.

1. Draw the binary tree resulting from the recursive split of the set U.
2. Denote the depth of the binary tree by k and for all $0 \leq i \leq k$ and $0 \leq j < 2^{k-i}$, define $M_{i,j} = \prod_{l=0}^{2^i-1} m_{j2^i+l}$. Prove that for each i, j

$$\begin{cases} M_{i+1,j} &= M_{i,2j} M_{i,2j+1} \\ M_{0,j} &= m_j. \end{cases} \tag{1.1}$$

3. How do the $M_{i,j}$ relate to the binary tree?
4. Fast multi-point evaluation.
   a) Write an algorithm that builds the subproduct tree and returns the polynomials $M_{i,j}$ as defined in (1.1).
   b) Write an recursive algorithm which takes a polynomial P of degree less than $n = 2^k$ as input as well as $u_0, \cdots, u_{n-1}$ and the subproducts $M_{i,j}$. It should go down the subproduct tree and return $P(u_0), \cdots, P(u_{n-1})$.
5. Correctness and complexity.
   a) By induction on k, prove the correctness of the previous algorithm.
   b) Show that the complexity of the algorithm is $\mathcal{O}(\mathsf{M}(n) \log n)$ operations in R.

Part II — Fast interpolation

Reusing the notations from part I, let m be the product of all the m_i, i.e. $m = \prod_{i=0}^{n-1}(X - u_i)$.

* 1. Explain how to perform Lagrange interpolation.

Hint: an element a in R is invertible if there is a b in R such that $ab = e$, with e a unit in R.

2. Let $s_i = \prod_{i \neq j} 1/(u_i - u_j)$. Prove that m', the derivative of m, is $m' = \sum_{j=0}^{n-1} m/(x - u_j)$ and that $m'(u_i) = 1/s_i$.

3. Devise a divide and conquer algorithm which proceeds from the leaves to the root of the binary tree from part I question 1, in order to return the interpolation of P at the points $u_0, \cdots, u_{n-1}$. *Hint:* use the M_i, j to apply a recursive approach to Lagrange interpolation.

4. Correctness and complexity.

   * a) By induction on k, prove the correctness of the previous algorithm.

   b) Prove that computing the s_i in question 2, amounts to $\mathcal{O}(\mathsf{M}(n) \log n)$ operations in R.

   c) Conclude that the interpolation problem can be solved in $\mathcal{O}(\mathsf{M}(n) \log n)$ ring operations.

5. Discuss the possibility of pre-computing the subproducts M_i, j.

Ex. 2 — *Critical thinking*

* 1. Let G be a group such that for all x, y in G, $(xy)^2 = (yx)^2$, and for any $x \neq e$, $x^2 \neq e$, where e is a unit element. Prove that G is abelian.

2. After passing CS6033 two students, s_1 and s_2, are asked to determine two integers x and y such that $1 < x < y$ and $x + y < 100$. Student s_1 is told that $x + y$, while s_2 is given xy. Remembering the importance of critical thinking they start discussing:

   S_2 : "No idea what those two numbers could be..."

   S_1 : "I'm not surprised, I knew you couldn't know!"

   S_2 : "Uhm...so now I know..."

   S_1 : "So do I!"

   What about you?

* **Ex. 3 —** *Beyond CS6033*

Explain what the Swype keyboard is and propose some hints on how it could be implemented.

* **Ex. 4 —** *Interview problem*

1. Write a short program allowing to expand a binary tree into a linked list with all elements in increasing order.

2. Given a string, find the longest substring without duplicate, i.e. each character should appear no more than once in the substring.

   Example. For input `abcabcbb`, return `abc` with length 3, and for `?????`, return `?` with length 1.