VE477 Introduction to Algorithms
Homework 8

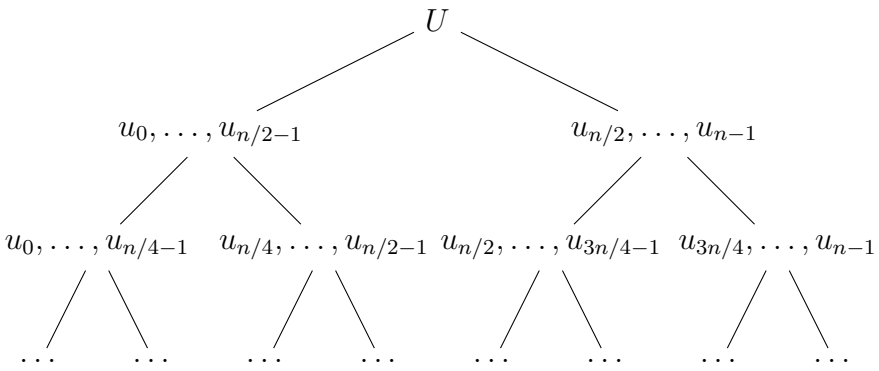
Taoyue Xia, 518370910087

2021/12/06

Ex. 1 — Fast multi-point evaluation and interpolation

Part I — Fast multi-point evaluation

1. The binary tree is as follows,



2. Firstly, when $i = 0$, we can have

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

Then for general cases,

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

With the above two deduction, we have proved that,

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

3. $M_{0,j}$ can be seen as the n leaves $m_0, \ldots, m_{n-1}$, and $M_{i,j}$ is the j_{th} elements at depth i. Also, For $M_{i+1,j}$, it has two children $M_{i,2j}$ and $M_{i,2j+1}$. Therefore, as the depth increases by 1, the number of nodes will shrink to half of the lower level nodes. So at the final depth k, the n-degree polynomial $M_{k,0}$ would exist.

4. a) The algorithm is shown below,

Algorithm 1: Subproduct Tree Build

Input : $n = 2^k$ a power of 2, $U = \{u_0, \ldots, u_{n-1}\}$
Output: $M_{k,0}$

```
for j = 0 to n − 1 do
    M_{0,j} ← x − u_j;
end
for i = 1 to k do
    for j = 0 to 2^{k−i} − 1 do
        M_{i,j} ← M_{i−1,2j} M_{i−1,2j+1};
    end
end
return M_{k,0};
```

b) The algorithm is shown below,

Algorithm 2: Recursive Subproduct Tree Downward

Input : $n = 2^k$, P a polynomial of degree less than n, set $\{u_0, \ldots, u_{n-1}\}$, $M_{i,j}$
Output: $P(u_0), \ldots, P(u_{n-1})$

```
Function RecurseDown(arr, M_{i,j}), P:
    if i == 0 then
        arr[2j] ← P;
        return;
    end
    RecurseDown(arr, M_{i−1,2j}, P mod M_{i−1,2j});
    RecurseDown(arr, M_{i−1,2j}, P mod M_{i−1,2j+1});
end
arr ← an empty array;
RecurseDown(arr, M_{k,0}, P);
return arr;
/* arr contains P_{u_0}, ..., P(u_{n−1})                    */
```

5. a) When $k = 0$, then the polynomial M has degree 1, then P can only be degree 0, which can be returned directly.

   Suppose that for $k = m$ ($m \in \mathbb{N}^*$), it returns $P(u_0), \ldots, P(u_{2^m-1})$. Then for $k = m+1$, at level m, all the $M_{1,j}$ has degree 2, so it is possible for P at that level to have degree 1, which does not give an answer. Therefore, we need to go down another level to $m + 1$, so that we can have the values $P(u_0, \ldots, P(u_{2^{m+1}-1}))$. Proof done.

   b) From the above algorithm, we can write the time needed in every level,

$$T(n) = 2T(n/2) + \mathcal{O}(M(n))$$

   Therefore, according to the Master's Theory, the overall time complexity is $\mathcal{O}(M(n) \log n)$.

Part II — Fast interpolation

1. Knowing the points (x_i, y_i), the Lagrange Interpolation method can be expressed as:

$$\ell_i(x) = \prod_{\substack{0 \le m \le k \\ m \ne i}} \frac{x - x_m}{x_i - x_m} = \frac{(x - x_0) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_k)}{(x_i - x_0) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_k)}$$

$$L(x) = \sum_{i=0}^{k} y_i \ell_i(x)$$

   For our question, having $u_0, \ldots, u_{n-1}$ as x_i and $P(u_0), \ldots, P(u_{n-1})$ as y_i, and $m = \prod_{i=0}^{n-1}(X - u_i)$, we can have

$$P = \sum_{i=0}^{n-1}(y_i \cdot \prod_{j \ne i} \frac{1}{u_i - u_j} \cdot \frac{m}{X - u_i})$$

   Moreover, for each $u_i - u_j$ and $i \ne j$, we can find a $v_{i,j}$ in R such that $(u_i - u_j)v_{i,j} = e$, the unit element.

2. Knowing that $m = \prod_{i=0}^{n-1}(x - u_i)$, denote $p_i = \prod_{j=0}^{i-1}(x - u_j)$ we can deduce that,

$$m' = (\prod_{i=0}^{n-1}(x - u_i))' = p_{n-1} + (x - u_{n-1})p'_{n-1}$$

$$p'_{n-1} = p_{n-2} + (x - u_{n-2})p'_{n-2}$$

   Then we can iterate this progress until it reaches p_1, and by expansion, the expression is

$$m' = \frac{m}{x - u_{n-1}} + \frac{m}{x - u_{n-2}} + \cdots + \frac{m}{x - u_0} = \sum_{i=0}^{n-1} \frac{m}{x - u_i}$$

   According to this equation, we can have

$$m'(u_i) = \prod_{j \ne i}(u_i - u_j) = \frac{1}{s_i}$$

   Since all the terms containing $x - u_i$ is 0. Proof done.

3. The algorithm is as follows,

Algorithm 3: Interpolation for P

Input : $n = 2^k$, $u_0, \ldots, u_{n-1}$, $M_{i,j}$, $P_{u_0}, \ldots, P(u_{n-1})$
Output: P

```
Function Interpolation(P(u_i), M_{i,j}):
    if i = 0 then
        return P(u_j);
    end
    l ← Interpolation(P(u_i), M_{i-1,2j});
    r ← Interpolation(P(u_i), M_{i-1,2j+1});
    result ← l · M_{i-1,2j} + r · M_{i-1,2j+1};
    return result;
end
```

4. Correctness and complexity

   a) Same induction as in Part I 5(a).

   b) For the deduction we have discussed in question 2, the time complexity of computing m' can be expressed as

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

   After calculating m', we just need to insert u_i and take the reciprocal, we can get s_i. Therefore, we can prove that computing s_i in question 2 amounts to $\mathcal{O}(M(n)\log n)$ operations in R.

   c) The time complexity for construct the subproduct tree is $\mathcal{O}(M(n)\log n)$, and the time complexity of calculation for s_i is also $\mathcal{O}(M(n)\log n)$. After combining these two, we can perform the interpolation in linear time. Therefore, the interpolation problem can be solved in $\mathcal{O}(M(n)\log n)$ ring operations.

5. The possibility depends on the memory space needed for storing $M_{i,j}$.

Ex. 2 — Critical thinking

1. If $xy = e$, then $y = x^{-1}$, so $xy = yx = e$.

   If $xy \neq e$, replace y with $x^{-1}y$ (since $\forall x, y \in G$, $x^{-1}y$ can form a bijection $G \to G$). Then we can have

   $$\begin{aligned}(xx^{-1}y)^2 &= (x^{-1}yx)^2 \\ y^2 &= (x^{-1}yx)(x^{-1}yx) \\ y^2 &= x^{-1}y^2x \\ xy^2 &= yx^2\end{aligned}$$

   Since it is mentioned in the stem that $\forall x \neq e,\ x^2 \neq e$, we can reduce y^2 and x^2 to y and x, while keeping generality. Therefore, we can have $xy = yx$, which proves that G is abelian.

2. - Since S_2 says that he cannot infer the two numbers, it means that the two numbers are not both prime, because only when the number can be factorized as two prime numbers can S_2 figure them out.
   - Then S_1 says that he knew S_2 cannot know the two numbers, it means that his $x + y$ are not any of the sum of two primes. Then all the even numbers can be ignored, according to the Goldbach's conjecture, all the even numbers smaller than 4×10^18 can be represented as the sum of two primes. Also, the sum of 2 and other odd primes can be ignored too. After the extraction, we only have the numbers $x + y = 11, 17, 23, 27, 29, 35, 37, 41, 47, 53$ remaining.
   - Then according to what S_1 has said, S_2 would know that $x + y$ must fall in these numbers, and the product xy can only refer to one $x + y$.
   - Finally, we can know that the sum is 17 and $x = 4, y = 13$ or $x = 13, y = 4$.

Ex. 3 — Beyond ve477

Swype was a virtual keyboard for touchscreen smartphones and tablets, where the user enters words by sliding a finger or stylus from the first letter of a word to its last letter, lifting only between words.

It could be implemented in the following steps:

- Firstly, we need to trace the path of user's finger, when it detects an obvious angle, the letter at the angle could be recorded.
- Then it comes to the priority of the display sequence of words, from common to rare.
- After that, we can record users' priority and habits to offer personalized word association.
- Finally, it comes to the important function, the error correction. When it detects some strange word, which may be caused by mistyping, it can correct it probably using AI or other algorithms, and hint for the right one.