Conversão R2 $\rightarrow$ R1

Do ponto de vista da nossa disciplina de algoritmos, talvez a gente já pudesse parar por aqui.

Quer dizer, a solução que vimos para o problema da conversão R1 $\rightarrow$ R2 nos dá o exemplo mais interessante de divisão e conquista que nós vamos ver no curso.

Mas, do ponto de vista do problema da multiplicação de polinômios, é preciso seguir adiante e mostrar como se resolve a conversão R2 $\rightarrow$ R1.

(Nós vamos ver que essa solução é tão interessante quanto a anterior ...)

Nós vimos na Introdução que a conversão R2 $\rightarrow$ R1 consiste basicamente na solução de um sistema de equação lineares.

$$\begin{aligned} 1 &= a \cdot 1^2 + b \cdot 1 + c \\ 2 &= a \cdot 3^2 + b \cdot 3 + c \\ 3 &= a \cdot 4^2 + b \cdot 4 + c \end{aligned}$$

Isto é, cada ponto da coleção U_m nos dá uma equação, onde a equação é obtida substituindo a variável x no polinômio pelo valor ponto, e o outro lado da equação é o valor do polinômio naquele ponto.

Abaixo nós temos o sistema de equação no caso geral (em notação matricial):

$$\begin{bmatrix} 1 & p_0 & p_0^2 & \dots & p_0^{n-1} \\ 1 & p_1 & p_1^2 & \dots & p_1^{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & p_{n-1} & p_{n-1}^2 & \dots & p_{n-1}^{n-1} \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ \dots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} A(p_0) \\ A(p_1) \\ \dots \\ A(p_{n-1}) \end{bmatrix}$$

Note o formato peculiar dessa matriz.

Um dia[b], em que não tinha nada de mais importante para fazer, Alexandre Vandermonde ficou curioso a respeito dessa matriz e resolveu estudá-la.

Ele descobriu que a sua inversa também tem um formato peculiar.

No nosso caso particular (onde as entradas da matriz são pontos da coleção U_m) a inversa é a seguinte:

$$\frac{1}{n} \cdot \begin{bmatrix} 1 & p_0^{-1} & p_0^{-2} & \dots & p_0^{-(n-1)} \\ 1 & p_1^{-1} & p_1^{-2} & \dots & p_1^{-(n-1)} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & p_{n-1}^{-1} & p_{n-1}^{-2} & \dots & p_{n-1}^{-(n-1)} \end{bmatrix}$$

Multiplicando essa inversa nos dois lados da equação acima (e trocando a ordem dos lados), nós obtemos

$$\frac{1}{n} \cdot \begin{bmatrix} 1 & p_0^{-1} & p_0^{-2} & \dots & p_0^{-(n-1)} \\ 1 & p_1^{-1} & p_1^{-2} & \dots & p_1^{-(n-1)} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & p_{n-1}^{-1} & p_{n-1}^{-2} & \dots & p_{n-1}^{-(n-1)} \end{bmatrix} \cdot \begin{bmatrix} A(p_0) \\ A(p_1) \\ \dots \\ A(p_{n-1}) \end{bmatrix} = \begin{bmatrix} a_0 \\ a_1 \\ \dots \\ a_{n-1} \end{bmatrix}$$

Ou seja, para obter os coeficientes $a_0, a_1, ldots, a_n$, basta fazer uma multiplicação de matriz por vetor.

Infelizmente, essa multiplicação leva tempo $\Theta(n^2)$ ...

Mas, isso não é necessário!

Note que essa multiplicação corresponde a calcular o valor do polinômio com coeficientes $A(p_0)$, $A(p_1)$, ..., $A(p_{n-1})$ sobre a coleção de pontos $U'_m = \{p_0^{-1}, p_1^{-1}, \dots, p_{n-1}^{-1}\}$.

A coleção de pontos U'_m possui as mesmas propriedades da coleção U_m.

Logo, a conversão R2 $\rightarrow$ R1 pode ser realizada da mesma forma que a conversão R1 $\rightarrow$ R2.

E isso leva tempo $\Theta(n \log n)$.

Juntando todas as etapas do processo, nós tempos um algoritmo para a multiplicação de polinômios de grau n que executa em tempo

$$T(n) = \Theta(n \log n) + \Theta(n) + \Theta(n \log n) = \Theta(n \log n)$$

[b] Provavelmente mais de um dia ...