Matemática

Polinômios

Prof. Edson Alves
Faculdade UnB Gama

Definição de polinômios

Seja A o conjuntos dos coeficientes e V o conjuntos das variáveis. Um polinômio é definido por meio de elementos de A e de um subconjunto de variáveis de V através de operações ariméticas de adição, subtração, multiplicação e divisão.

Polinômios univariados

Um polinômio univariado, isto é, definido em uma única variável x, pode ser escrito como

$$a_0 + a_1 x + a_2 x^2 + \ldots + a_N x^N$$

Em forma de somatório,

$$\sum_{i=0}^{N} a_i x^i$$

O maior expoente da variável x na expressão que define o polinômio (N nas notações acima, se $a_N \neq 0$) é denominado **grau** do polinômio.

Polinômios em C/C++

Polinômios podem ser representados em C/C++ através de *arrays* ou de vetores:

```
using polynomial = vector<int>;
```

Observe que um polinômio de grau N tem $N+1$ coeficientes:

```
int degree(const polynomial& p) { return p.size() - 1; }
```

Em geral, os coeficientes são armazenados do termo constante ao termo que determina o grau do polinômio.

```
polynomial p { 6, -5, 1 };          // p(x) = x^2 - 5x + 6
```

Função polinomial

- Todo polinômio está associado a uma função $p(x)$ dada por

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_N x^N$$

- Para computar o valor y_0 associado ao valor x_0 por $p(x)$, basta substituir todas as ocorrências de x no polinômio por x_0, isto é,

$$y_0 = a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_N x_0^N$$

- O cálculo de y_0 tem complexidade $O(N^2)$, se cada termo x^i for computado por meio de somas repetidas

Algoritmo de Horner

- O algoritmo de Horner computa $y_0 = p(x_0)$ com complexidade $O(N)$
- Este algoritmo é baseado na regra de Horner:

$$a_0 + a_1 x + a_2 x^2 + \ldots + a_N x^N = a_0 + x(a_1 + x(a_2 + \ldots + x(a_N) + \ldots))$$

- O algoritmo inicia fazendo $y_0 \leftarrow 0$ e $i \leftarrow N$
- Para cada i ele atualiza o valor de y por meio de duas operações:
  1. $y_0 \leftarrow y_0 \times x_0$
  2. $y_0 \leftarrow y_0 + a_i$
- Em seguida, o valor de i é decrementado

Implementação do algoritmo de Horner

```
int evaluate(const polynomial& p, int x)
{
    int y = 0, N = degree(p);

    for (int i = N; i >= 0; --i)
    {
        y *= x;
        y += p[i];
    }

    return y;
}
```

Zeros de polinômios

- Se $p(x_0) = 0$, dizemos que x_0 é um **zero** (ou raiz) de $p(x)$
- O Teorema Fundamental da Álgebra afirma que um polinômio de grau N tem N raízes complexas (não necessariamente distintas)
- O polinômio $p(x) = a_0 + a_1 x$, com $a_1 \neq 0$, tem um único zero, a saber:

$$x = -\frac{a_0}{a_1}$$

Zeros de polinômios

- O polinômio $p(x) = a_0 + a_1x + a_2x^2$, com $a_2 \neq 0$, tem duas raízes complexas, que podem ser obtidas por meio da fórmula de Bhaskara:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- Há expressões semelhantes para polinômios de grau 3 (Formula de Cardano/Tartaglia) e 4 (Fórmula de Ferrari), porém não são de fácil memorização
- Para polinômios de grau 5 ou maior, Galois mostrou, usando a Teoria dos Grupos, que não há expressão racional para computar suas raízes

Adição de polinômios

A adição de dois polinômios $p(x) = a_0 + a_1 x + \ldots, a_N x^N$ e $q(x) = b_0 + b_1 x + \ldots + b_M x^M$ resulta em um polinômio

$$r(x) = p(x) + q(x) = (a_0 + b_0) + (a_1 + b_1)x + \ldots + (a_R + b_R)x^R,$$

onde $R \leq \max\{N, M\}$, $a_i = 0$, se $i > N$ e $b_j = 0$, se $j > M$.

Implementação da adição de polinômios

```
polynomial operator+(const polynomial& p, const polynomial& q)
{
    int N = degree(p), M = degree(q);
    polynomial r(max(N, M) + 1, 0);

    for (int i = 0; i <= N; ++i)
        r[i] += p[i];

    for (int i = 0; i <= M; ++i)
        r[i] += q[i];

    while (not r.empty() and r.back() == 0)
        r.pop_back();

    if (r.empty())
        r.pop_back(0);

    return r;
}
```

Multiplicação de polinômios

- A multiplicação de polinômios é feita por meio da aplicação da distributividade
- Se $\text{grau}(p(x)) = N$, $\text{grau}(q(x)) = N$ e $r(x) = p(x)q(x)$, então $\text{grau}(r(x)) = NM$
- Se $p(x)$ em coeficientes $a_0, a_1, \ldots, a_N$ e $q(x)$ tem coeficientes $b_1, b_2, \ldots, b_M$, então os coeficientes c_i de $r(x)$ são dados por

$$c_i = \sum_{k=0}^{i} a_k b_{i-k}$$

Implementação da multiplicação de polinômios

```
polynomial operator*(const polynomial& p, const polynomial& q)
{
    int N = degree(p), M = degree(q);
    polynomial r(N*M + 1, 0);

    for (int i = 0; i <= N; ++i)
        for (int j = 0; j <= M; ++j)
            r[i + j] = p[i]*q[j];

    while (not r.empty() and r.back() == 0)
        r.pop_back();

    if (r.empty())
        r.pop_back(0);

    return r;
}
```

Divisão de polinômios

- A divisão de polinômios é idêntica à divisão de Euclides nos inteiros
- Dados dois polinômios $a(x)$ e $b(x)$, onde $b(x) \neq 0$, existem dois polinômios $q(x)$ e $r(x)$ tais que

$$a(x) = b(x)q(x) + r(x),$$

tais que $0 \leq \text{grau}(r(x)) < \text{grau}(b(x))$

Divisão de polinômios e zeros

- Observe que, se x_0 é uma raiz de $p(x)$, então $(x - x_0)$ divide $p(x)$
- De fato, pela divisão de polinômios existem $q(x)$ e $r(x)$ tais que

$$p(x) = (x - x_0)q(x) + r(x)$$

- Assim, como x_0 é raiz, vale que

$$0 = p(x_0) = (x_0 - x_0)q(x_0) + r(x_0) = r(x_0)$$

Fatoração de polinômios

- Pelo Teorema Fundamental da Álgebra, um polinômio $p(x)$ de grau N tem N raízes complexas
- Para cada raiz x_i, o polinômio $(x - x_i)$ divide $p(x)$
- Portanto, $p(x)$ pode ser escrito na forma

$$p(x) = a(x - x_1)(x - x_2)\ldots(x - x_N)$$

- Esta fatoração remete à ideia de decomposição de um inteiro em fatores primos

Polinômios irredutíveis

- Considere que os coeficientes de um polinômio $p(x)$ e os valores que a variável x pode assumir pertencem a um conjunto A
- Se $i(x)$ não pode ser fatorado em A, isto é, não existem dois polinômios com coeficientes em A de grau maior do que zero tais que $i(x) = a(x)b(x)$, ele é denominado polinômio **irredutível** em A
- Não ter zeros em A é necessário, mas não suficiente, para que $p(x)$ seja irredutível
- Por exemplo, $p(x) = (x^2 + 1)(x^2 + 1)$ é redutível mas não tem zeros nos inteiros

Relações de Girard

- As relações de Girard são consequentes da igualdade entre a representação de um polinômio e sua fatoração, isto é

$$c_0 + c_1 x + c_2 x^2 + \ldots + a_N x^N = a(x - x_1)(x - x_2)\ldots(x - x_N)$$

- As duas relações mais importantes são:

$$x_1 x_2 \ldots x_N = \frac{c_0}{a}$$

e

$$x_1 + x_2 + \ldots + x_N = \frac{c_{N-1}}{a}$$

Polinômios nos inteiros

- Se $p(x)$ é um polinômio definido no conjunto dos números inteiros, suas raízes tem relações de divisibilidade com o produto de seus coeficientes, de acordo com as relações de Girard
- Se x_0 é uma raiz inteira de $p(x)$, então x_0 é um divisor de c_0/a
- Assim, o conjunto de candidatos a raiz inteira de $p(x)$ tem tamanho $O(\sqrt{c_0})$
- Uma vez encontrada uma raiz inteira x_0 de $p(x)$, ele pode ser dividido por $(x - x_0)$ para obter um novo polinômio $q(x)$ de grau $N - 1$
- Como cada raiz de $q(x)$ será também raiz de $p(x)$, o processo pode ser repetido para determinar todas as raízes inteiras de $p(x)$

Problemas

- Codeforces
  1. 20B - Equation
- OJ
  1. 498 - Polly the Polynomial
  2. 10268 - 498-bits
  3. 10302 - Summation of Polynomials
  4. 10586 - Polynomial Remains

Referências

1. Wikipédia. Horner's method. Acesso em 10/02/2021.
2. Wikipédia. Newton's Identities. Acesso em 11/02/2021.
3. Wikipédia. Polynomial. Acesso em 10/02/2021.
4. WolframMathWorld. Cubic Formula. Acesso em 10/02/2021.
5. WolframMathWorld. Polynomial. Acesso em 10/02/2021.
6. WolframMathWorld. Quartic Equation. Acesso em 10/02/2021.