Estruturas de Dados

Cristina Gomes Fernandes

Representação de polinômios

$$p(x) = -4 + 2x - 6x^2 + 3x^4$$

0	1	2	3	4
-4	2	-6	0	3

Representação de polinômios

$p(x) = -4 + 2x - 6x^2 + 3x^4$

0	1	2	3	4
-4	2	-6	0	3

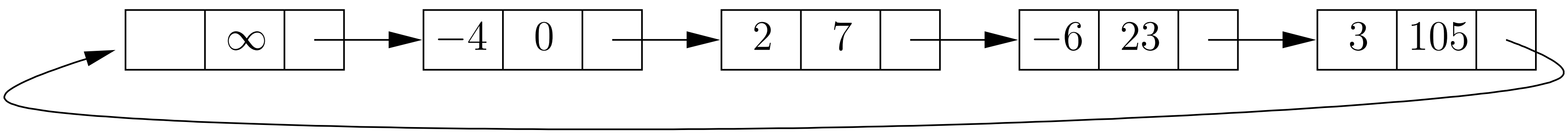
Soma de um polinômio de grau n com um de grau m consome tempo $\Theta(n+m)$.

Produto consome tempo $\Theta(mn)$.
(Há um pequeno cuidado que você deve tomar na implementação para garantir esse consumo.)

Polinômios esparsos

Representação com lista ligada circular com cabeça: cada célula tem os campos $coef$, $expo$ e $prox$.

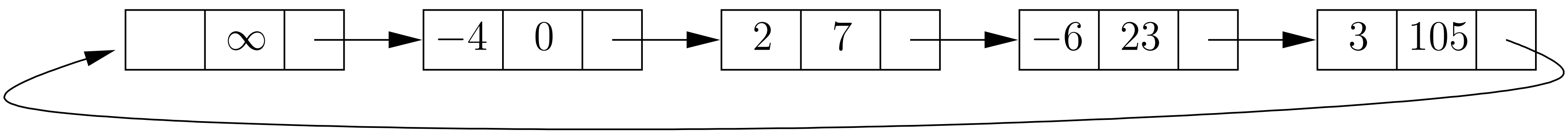
$$p(x) = -4 + 2x^7 - 6x^{23} + 3x^{105}$$



Polinômios esparsos

Representação com lista ligada circular com cabeça:
cada célula tem os campos coef, expo e prox.

$$p(x) = -4 + 2x^7 - 6x^{23} + 3x^{105}$$

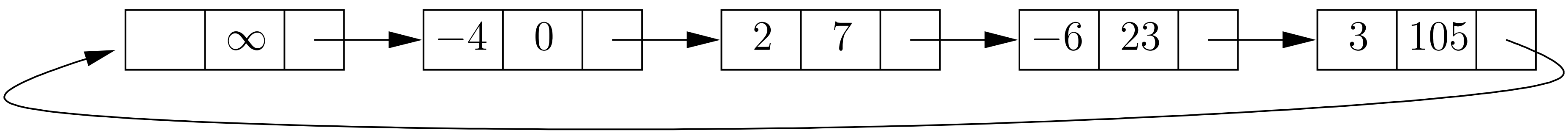


A cabeça tem ∞ no seu campo expo.

Polinômios esparsos

Representação com lista ligada circular com cabeça:
cada célula tem os campos $coef$, $expo$ e $prox$.

$$p(x) = -4 + 2x^7 - 6x^{23} + 3x^{105}$$



A cabeça tem ∞ no seu campo $expo$.

Os elementos estão ordenados pelo expoente.

Apenas monômios com coeficiente não-nulo
devem estar presentes na lista.

Soma de polinômios esparsos

SOMA (p, q)

$s \leftarrow$ NULO$()$ ▷ polinômio nulo
app $\leftarrow prox(p)$
apq $\leftarrow prox(q)$
enquanto $expo($*apq*$) \neq \infty$ **ou** $expo($*apq*$) \neq \infty$ **faça**
  se $expo($*app*$) < expo($*apq*$)$
    então $coef(s) \leftarrow coef($*app*$)$
      $expo(s) \leftarrow expo($*app*$)$
      app $\leftarrow prox($*app*$)$
    senão se $expo($*app*$) > expo($*apq*$)$
      então $coef(s) \leftarrow coef($*apq*$)$
        $expo(s) \leftarrow expo($*apq*$)$
        apq $\leftarrow prox($*apq*$)$
      senão $\cdots$ ▷ $expo($*app*$) = expo($*apq*$)$

Soma de polinômios esparsos

```
SOMA (p, q)
 ...
 enquanto apq ≠ NIL ou apq ≠ NIL faça
 ...
       senão ▷ expo(app) = expo(apq)
          se coef(app) + coef(apq) ≠ 0
             então coef(s) ← coef(app) + coef(apq)
                   expo(s) ← expo(app)
          app ← prox(app)
          apq ← prox(apq)
    se expo(s) ≠ ∞
       então t ← NOVACÉLULA(0, ∞)
             prox(t) ← prox(s)
             prox(s) ← t
             s ← t
 devolva s
```

Soma de polinômios esparsos

NULO ()

$p \leftarrow$ NOVACÉLULA$(0, \infty)$

$prox(p) \leftarrow p$

devolva p

Soma de polinômios esparsos

NULO ()

1 $p \leftarrow$ NOVACÉLULA$(0, \infty)$

2 $prox(p) \leftarrow p$

3 **devolva** p

Consumo de tempo:

O algoritmo SOMA consome tempo $\Theta(n+m)$, onde n e m são o número de monômios não-nulos em p e q respectivamente.

Produto de polinômios esparsos

```
PRODUTO (p, q)
r ← NULO()      ▷ polinômio nulo
app ← prox(p)
enquanto expo(app) ≠ ∞ faça
    apq ← prox(q)
    t ← r
    enquanto expo(apq) ≠ ∞ faça
        enquanto expo(prox(t)) < expo(app) + expo(apq)
            faça t ← prox(t)
        se expo(prox(t)) ≠ expo(app) + expo(apq)
            então s ← NOVACÉLULA(coef(app) * coef(apq),
                                            expo(app) + expo(apq))
                  prox(s) ← prox(t)
                  prox(t) ← s
            senão ···
```

Produto de polinômios esparsos

PRODUTO (p, q)

$\cdots$

enquanto $expo(\text{app}) \neq \infty$ **faça**

  $\cdots$

  enquanto $expo(\text{apq}) \neq \infty$ **faça**

    $\cdots$

      senão se $coef(prox(t)) + coef(\text{app}) * coef(\text{apq}) = 0$

        então $s \leftarrow prox(t)$

          $prox(t) \leftarrow prox(s)$

          LIBERACÉLULA(s)

        senão $t \leftarrow prox(t)$

          $coef(t) \leftarrow coef(t) + coef(\text{app}) * coef(\text{apq})$

    $\text{apq} \leftarrow prox(\text{apq})$

  $\text{app} \leftarrow prox(\text{app})$

devolva r

Consumo de tempo do produto

Sejam n e m o número de monômios com coeficiente não-nulo em p e q respectivamente.

Consumo de tempo do produto

Sejam n e m o número de monômios com coeficiente não-nulo em p e q respectivamente.

O polinômio r construído pode ter mn monômios com coeficiente não-nulo no pior caso.

Consumo de tempo do produto

Sejam n e m o número de monômios com coeficiente não-nulo em p e q respectivamente.

O polinômio r construído pode ter mn monômios com coeficiente não-nulo no pior caso.

Assim sendo, para cada iteração do **enquanto** da linha 3, o número total de execuções da linha 8 é $O(mn)$.

Consumo de tempo do produto

Sejam n e m o número de monômios com coeficiente não-nulo em p e q respectivamente.

O polinômio r construído pode ter mn monômios com coeficiente não-nulo no pior caso.

Assim sendo, para cada iteração do **enquanto** da linha 3, o número total de execuções da linha 8 é $O(mn)$.

Como entramos no **enquanto** da linha 3 n vezes, o consumo de tempo do PRODUTO é $O(mn^2)$ no pior caso.

Consumo de tempo do produto

Sejam n e m o número de monômios com coeficiente não-nulo em p e q respectivamente.

O polinômio r construído pode ter mn monômios com coeficiente não-nulo no pior caso.

Assim sendo, para cada iteração do **enquanto** da linha 3, o número total de execuções da linha 8 é $O(mn)$.

Como entramos no **enquanto** da linha 3 n vezes, o consumo de tempo do PRODUTO é $O(mn^2)$ no pior caso.

Logo, a implementação fica mais eficiente se deixarmos no papel de p sempre o polinômio entre p e q com menos monômios não-nulos.