Studi Kasus: Polinom Representasi Berkait

IF2110/IF2111 – Algoritma dan Struktur Data Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung

Deskripsi Persoalan

Sebuah polinom berderajat n didefinisikan sebagai fungsi P(x) berikut:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \dots + a_2 x^2 + a_1 x^1 + a_0$$

Perhatikan bahwa untuk mempermudah pemrosesan, definisi P(x) tersebut berbeda dengan definisi polinom pada matematik, yang biasanya diberikan sebagai berikut:

$$P(x) = a_0 x^n + a_1 x^{n-1} + a_2 x^{n-2} + \dots + a_{n-2} x^2 + a_{n-1} x^1 + a_n$$

Contoh Polinom:

$$P1(x) = 4x^{5} + 2x^{4} + 7x^{2} + 10$$

$$P2(x) = 23x^{100} + 9x^{9} + 2x^{7} + 4x^{5} + 9x^{9} + 2x^{4} + 3x^{2} + 1$$

$$P3(x) = 10$$

$$P4(x) = 3x^{2} + 2x + 8$$

$$P5(x) = x^{1000}$$

Proses Dasar Polinom (1)

- 1. Membentuk sebuah polinom *P* dari pasangan harga yang dibaca dari masukan, data yang dibaca adalah pasangan harga:
 - (*) \(Degree: integer, Coefficient: integer \)
 - $(1) \langle -999, 0 \rangle$
- 2. Menuliskan sebuah polinom P terurut mulai dari suku terbesar sampai terkecil
- 3. Menjumlahkan dua buah polinom P1 dan P2 dan menyimpan hasilnya pada P3,
 - $P3 \neq P1$ dan $P3 \neq P2$; pada akhir proses P3 = P1 + P2.

Proses Dasar Polinom (2)

- 4. Mengurangi dua buah polinom P1 dan P2 dan menyimpan hasilnya pada P3, $P3 \neq P1$ dan $P3 \neq P2$, pada akhir proses P3 = P1 P2.
- 5. Membuat turunan dari sebuah polinom P dan menyimpan hasilnya pada P', $P' \neq P1$, pada akhir proses P' adalah turunan P1.

Pemilihan Struktur Data

Secara lojik sebuah suku polinom direpresentasi oleh sepasang harga integer:

```
〈 Degree: integer, Coefficient: integer 〉
```

P(x) adalah kumpulan pasangan harga tersebut yang diidentifikasi oleh namanya dan merupakan sebuah list linier dengan elemen Suku, yaitu pasangan harga (Degree: integer, Coefficient: integer), dengan harga Degree yang maksimum sebagai derajat polinom.

P(x) dapat direpresentasi secara **KONTIGU** atau **BERKAIT**.

Representasi Berkait (1)

- Hanya suku yang muncul saja yang disimpan datanya.
- Degree setiap suku harus disimpan secara eksplisit.
- Operasi akan lebih efisien jika suku yang muncul diurut mulai dari derajat tertinggi sampai derajat terendah

Maka:

- terjadi penghematan memori kalau suku-suku [0..N] banyak yang tidak muncul. Kalau banyak yang muncul?
- polinom kosong menjadi sangat "natural"

Representasi Berkait (2)

Membentuk sebuah polinom

Membentuk sebuah polinom dari pasangan harga yang dibaca dari alat masukan:

Setiap kali memasukkan data, yang dimasukkan adalah pasangan

(*) \(\text{Degree: integer, Coefficient: integer } \)

Akhir pemasukan adalah pasangan harga yang diketikkan bernilai (-999, 0)

Proses pembentukan polinom adalah proses sekuensial untuk membaca dari masukan dan melakukan penyisipan dalam list Suku polinom yang selalu terurut menurun menurut Degree.

Membentuk sebuah polinom

Contoh: Jika dibaca P1:

$$\langle 1,4 \rangle$$
, $\langle 2,5 \rangle$, $\langle 5,7 \rangle$, $\langle 8,9 \rangle$, $\langle 3,4 \rangle$, $\langle -999,0 \rangle$

maka list polinom yang dibentuk adalah polinom berderajat 8 dengan urutan penyisipan:

```
\( 1,4 \)
\( 2,5 \), \( 1,4 \)
\( 5,7 \), \( 2,5 \), \( 1,4 \)
\( 8,9 \), \( 5,7 \), \( 2,5 \), \( 1,4 \)
\( 8,9 \), \( 5,7 \), \( 3,4 \), \( 2,5 \), \( 1,4 \)
```

Polinom "kosong", yaitu polinom (P) = nil

Menuliskan sebuah polinom

Prosesnya menggunakan skema traversal dasar terhadap sebuah list polinom.

Harga suku yang dituliskan otomatis hanya yang koefisiennya tidak nol.

Contoh:

```
I P(I)
8 9
5 7
3 4
2 5
1 4
```

Menjumlahkan dua buah polinom (1)

Menjumlahkan dua buah polinom P1 dan P2 dan menyimpan hasilnya pada P3

Prosesnya adalah "merging" dua buah list linier yang terurut dan setiap suku P1 dan P2

Analisis kasus:

- derajat P1 sama dengan derajat P2, jumlahkan dan sisipkan pada P3 (jika hasil penjumlahan tidak 0), maju ke suku P1 dan P2 yang berikutnya
- derajat P1 > derajat P2: sisipkan suku P1 pada P3, maju ke suku P1 yang berikutnya
- derajat P1 < derajat P2: sisipkan suku P2 pada P3, maju ke suku P2 yang berikutnya

Menjumlahkan dua buah polinom (2)

```
Contoh, jika diberikan: P1: \langle 9,4 \rangle, \langle 7,4 \rangle, \langle 5,5 \rangle, \langle 4,-9 \rangle, \langle 3,2 \rangle, \langle 2,1 \rangle, \qquad \langle 0,10 \rangle P2: \langle 9,2 \rangle, \langle 7,3 \rangle, \qquad \langle 4,1 \rangle, \qquad \langle 1,2 \rangle Maka urutan pembentukan P3 adalah:  \langle 9,6 \rangle \qquad \text{InsertLast } (P3,P1+P2), \text{Next}(P1), \text{Next}(P2)   \langle 7,7 \rangle \qquad \text{InsertLast } (P3,P1+P2), \text{Next}(P1), \text{Next}(P2)   \langle 5,5 \rangle \qquad \text{InsertLast } (P3,P1), \text{Next}(P1)   \langle 4,-8 \rangle \qquad \text{InsertLast } (P3,P1+P2), \text{Next}(P1), \text{Next}(P2)   \langle 3,2 \rangle \qquad \text{InsertLast } (P3,P1), \text{Next}(P1)   \langle 2,1 \rangle \qquad \text{InsertLast } (P3,P1), \text{Next}(P1)   \langle 1,2 \rangle \qquad \text{InsertLast } (P3,P2), \text{Next}(P2)   \langle 0,10 \rangle \qquad \text{InsertLast } (P3,P1), \text{Next}(P1)  Keadaan akhir P3: \langle 9,6 \rangle, \langle 7,7 \rangle, \langle 5,5 \rangle, \langle 4,-8 \rangle, \langle 3,2 \rangle, \langle 2,1 \rangle, \langle 1,2 \rangle, \langle 0,10 \rangle
```

Mengurangi dua buah polinom

Sama dengan menjumlahkan, hanya operasi penjumlahan diganti operasi pengurangan. Contoh, jika diberikan:

Membuat turunan polinom

Prosesnya adalah proses sekuensial, traversal list P, untuk setiap suku ke-i, i>0: yaitu a_ix^i pada polinom P, dihitung $i * a_i$ dan disisipkan P1 sebagai suku ke-(i-1).

Seperti pada proses penjumlahan, list P1 dibentuk dengan penyisipan elemen terakhir, dan P1 diinisialisasi sebagai list kosong.

Karena penyisipan selalu pada akhir list, maka alamat elemen terakhir list P1 selama proses berlangsung layak untuk disimpan.

Contoh: Jika diberikan:

P:
$$\langle 9,4 \rangle$$
, $\langle 7,4 \rangle$, $\langle 5,5 \rangle$, $\langle 4,-9 \rangle$, $\langle 3,2 \rangle$, $\langle 2,1 \rangle$, $\langle 0,10 \rangle$

maka P1 akan mempunyai harga: $\langle 8,36 \rangle$, $\langle 6,28 \rangle$, $\langle 4,25 \rangle$, $\langle 3,-36 \rangle$, $\langle 2,6 \rangle$, $\langle 1,2 \rangle$

Representasi lojik	Representasi fisik	Representasi fisik
berkait	berkait dengan Pointer	berkait dengan Tabel
KAMUS UMUM	KAMUS UMUM	KAMUS UMUM
type Address:	<pre>type Address: pointer to Suku</pre>	<pre>constant NMAX: integer=100</pre>
type Suku:	type Suku:	<pre>type Address: integer[0NMAX]</pre>
< degree: <u>integer</u> ,	< degree: <u>integer</u> ,	type Suku:
coef: <u>integer</u> ,	coef: <u>integer</u>	< degree: <u>integer</u> ,
next: Address >	next: Address >	coef: <u>integer</u> ,
<u>type</u> Polinom: Address	<pre>type Polinom: Address</pre>	next: Address >
p : Polinom	p : Polinom	<u>type</u> Polinom: Address
pt: Address	pt: Address	arrSuku: <u>array</u> [0NMAX] <u>of</u> Suku
		firstAvail: Address
		p : Polinom
		pt: Address
AKSES:	AKSES:	AKSES:
FIRST(p)	р	p
NEXT(pt)	pt1.next	arrSuku[pt].next
DEGREE(pt)	pt1.degree	arrSuku[pt].degree
COEF(pt)	pt1.coef	arrSuku[pt].coef
PRIMITIF ALOKASI/DEALOKASI:	PRIMITIF ALOKASI/DEALOKASI:	PRIMITIF ALOKASI/DEALOKASI:
{ tergantung rep. fisik }	{ sistem, e.g., malloc/free }	{ Harus direalisasi }
	pt ← newSuku()	initialize(arrSuku)
	deallocSuku(pt)	pt ← newSuku()
		deallocSuku(pt)

```
Program POLINOMIAL1
{ Representasi BERKAIT, dengan notasi LOJIK }
KAMUS
{ Struktur data untuk representasi polinom }
   type Address: ... { type terdefinisi }
   type Suku: < degree: integer,
                  coef: integer,
                  next: Address >
   type Polinom: Address
   constant NIL: Address = ... { untuk address tidak terdefinisi }
   p1, p2: Polinom { operan }
   p3: Polinom { hasil }
{ Untuk interaksi: }
                      { mengakhiri proses }
   finish: boolean
   pilihan: integer [0..5] { nomor tawaran }
{ Primitif operasi polinom untuk operasi internal }
    function newSuku () → Address { Alokasi sebuah suku }
    procedure deallocSuku (input/output pt: Address) { Dealokasi sebuah suku }
    procedure CreatePolinom (output p: polinom) { Membuat polinom kosong P }
    procedure insertLast (input pt: Address, input/output p: Polinom,
                         input/output last: Address) { Insert pt sesudah
                         elemen terakhir p dengan address elemen terakhir
                          = last }
```

```
{ Primitif operasi polinom yang ditawarkan ke pengguna }
    procedure populatePol (output p1: polinom) { Mengisi polinom p1 }
    procedure displayPol (input p: polinom) { Menulis polinom p }
    procedure addPol (input p1, p2: polinom, output p3: polinom)
    { Menjumlahkan p1 + p2 dan menyimpan hasilnya di p3, p3 ≠ p1 dan p3 ≠ p2 }
    procedure subPol (input p1, p2: polinom, output p3: polinom)
    { Mengurangkan p1 - p2 dan menyimpan hasilnya di p3, p3 ≠ p1 dan p3 ≠ p2 }
    procedure derivPol (input p: polinom, output p1: polinom)
    { Membuat turunan p dan menyimpan hasilnya di p1, p1 ≠ p }
ALGORITMA
{ Sama dengan untuk representasi kontigu }
```