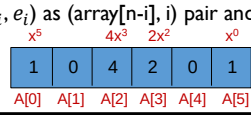
2.3 Polynomial

2.3

Polynomial

- $p(x) = a_0x^{e_0} + a_1x^{e_1} + \cdots + a_nx^{e_n} = \sum a_ix^{e_i}$
- Each $a_ix^{e_i}$ is called a **term** with coefficient a_i
  - The **degree** of p(x) is the largest exponent from among the non-zero terms.
  - Ex. $p(x) = x^5 + 4x^3 + 2x^2 + 1$
    Has 4 terms with coefficients 1, 4 ,2 and 1.
    The degree of p(x) is 5
- Array representation
  - Store (a_i, e_i) as (array[n-i], i) pair and n is the degree



Polynomial Operation

- If $a(x) = \sum a_ix^i$ and $b(x) = \sum b_ix^i$
- Polynomial addition
  - $a(x) + b(x) = \sum (a_i + b_i)x^i$

  Ex. $a(x) = x^5 + 4x^3 + 2x^2 + 1$ (degree = 5)
  $b(x) = 3x^6 + 4x^3 + x$ (degree = 6)
  $a(x) + b(x) = 3x^6 + x^5 + 8x^3 + 2x^2 + x + 1$ (degree = 6)
- Polynomial multiplication
  - $a(x) \cdot b(x) = \sum (a_ix^i \cdot \sum (b_jx^j))$

Polynomial :ADT

```
class Polynomial {
public:
   // Construct p(x) = 0
   Polynomial(void);
   // Destructor
   ~Polynomial(void);
   // Return the sum of *this and poly
   Polynomial Add(Polynomial poly);
   // Return multiplication of *this and poly
   Polynomial Mult(Polynomial poly);
   // Return the evaluation result
   float Eval(float x );
private:
   // Array representation
    …
};
```

We will ignore destructor in the codes hereafter. It is programmer's responsibility to treat her memory well ☺

9

Polynomial: 1st Representation

```
// in class Polynomial
public: // for convenience…
   // degree ≤ MaxDegree
   int degree;
   // coefficient array
   float coef[MaxDegree+1];
```

```
Usage:
   Polynomial a;
    a.degree = n;
    a.coef[i] = a_{n-i}
```

- Coefficients are stored in order of decreasing exponents
- Advantages:
  - Simple algorithm of operations
- Disadvantages:
  - Waste memory in a sparse polynomial

10

Polynomial: 2nd Representation

```
class Term {
 friend Polynomial;
 float coef;
 int exp;
};
```

```
// in class Polynomial
private:
  // array of nonzero terms
  Term* termArray;
  int capacity; // size of termArray
  int terms; // number of nonzero terms
```

- Store only nonzero terms.
- Each nonzero term holds an exponent and its corresponding coefficient.
- If polynomial is sparse, 2nd representation is better. If polynomial is full, 2nd one has double size of 1st.

11

2.3.2

Polynomial Addition: Code

```
Polynomial Polynomial::Add(Polynomial b)
{ // Return sum of polynomial *this and b
  Polynomial c;
  int aPos = 0, bPos = 0;
  while((aPos < terms) && (bPos < b.terms))
    if(termArray[aPos].exp == b.termArray[bPos].exp){
        float t = termArray[aPos].coef + b.termArray[bPos].coef;
        If(t) c.NewTerm(t, termArray[aPos].exp);
        aPos++; bPos++;
    }
    else if(termArray[aPos].exp < b.termArray[bPos].exp){
        c.NewTerm(b.termArray[bPos].coef, b.termArray[bPos].exp);
        bPos++;
    }
    else{
        c.NewTerm(termArray[aPos].coef,termArray[aPos].exp);
        aPos++;
    }
  // add in remaining terms of *this
  for(; aPos < terms; aPos++)
    c.NewTerm(termArray[aPos].coef, termArray[aPos].exp);
  // add in remaining terms of b
  for(; bPos < b.terms; bPos++)
    c.NewTerm(b.termArray[bPos].coef, b.termArray[bPos].exp);
  return c;
}
```

Example

$a(x) = x^5 + 9x^4 + 7x^3 + 2x$

$b(x) = x^6 + 3x^5 + 6x + 3$

$c(x) = x^6 + (1+3)x^5 + 9x^4 + 7x^3 + (2+6)x + 3$
$= x^6 + 4x^5 + 9x^4 + 7x^3 + 8x + 3$

13

Time Complexity of Analysis

- Inside the while loop: every statement takes O(1) time
- How many times the "while loop" is executed in the worst case ?
  - Let a(x) have m terms, and b(x) have n terms.
  - In each iteration, we access next element in a(x) or b(x), or both.
  - Worst case: m + n.
    e.g. It happens when
    $A(x) = 7x^5 + x^3 + x$; $B(x) = x^6 + 2x^4 + 6x^2 + 3$
    Access remaining terms in A(x): O(m)
    Access remaining terms in B(x): O(n)
- Hence, total run time = O(m + n)

14