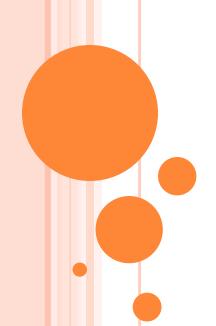




Jyh-Shing Roger Jang (張智星)
CSIE Dept, National Taiwan University





Array Representation of Polynomials

- A polynomial of order n
 - $a_n x^n + a_{n-1} x^{n-1} + ... + a_2 x^2 + a_1 x + a_0$
- Straightforward representation
 - Unique exponent arranged in increasing order
 - For instance: $7 x^5 + 4 x^3 + 2 x^2 + 3$

Representation:

p.degree = 5

p.coef = 302407

- Characteristics
 - Easy to implement addition & subtraction
 - Waste of space to represent a sparse polynomial x^{1000+x+1}
 - Complexity of addition: O(max(m, n))

Order of a(x).

Order of b(x).

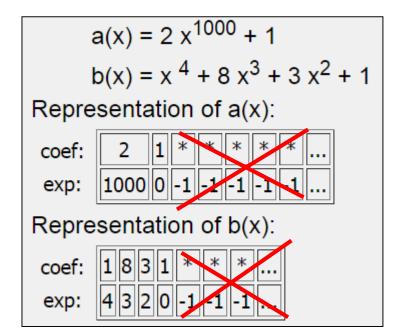


Sparse Array Representation of Polynomials

A polynomial of order n

•
$$a_n x^n + a_{n-1} x^{n-1} + ... + a_2 x^2 + a_1 x + a_0$$

- Representation
 - Keep non-zero terms only
 - For instance, see the right figure.
- How to add two polynomials
 - Traverse each polynomials
 - Add terms of the same exponent
- Characteristics
 - Takes only necessary memory
 - Complexity of addition & subtraction: O(m+n)



No. of terms in a(x). No. of terms in b(x).



Sparse Matrices

Matrix representation

Dense:

	·				
15	0	0	22	0	-15
0	11	3	0	0	0
0	0	0	-6	0	0
0	0	0	0	0	0
91	0	0	0	0	0
0	0	28	0	0	0

Sparse:



- Sparse format
 - The first row stores the numbers of rows, columns, and non-zero elements, respectively.
 - Elements are sorted by row first, by column second.



Complexity of Operations on Dense Matrices

Operations

c=transpose → O(mn)

```
for (i=0; i<m; i++)
for (j=0; j<n; j++)
c[j][i]=a[i][j];
```

• $c=add(a, b) \rightarrow O(mn)$

• c=multiply(a, b) \rightarrow O(pqr) for (i=0; i<p; i++)



Algorithm 1 for Transposing a Sparse Matrix

Algorithm 1

Pseudo code

```
for each entry {
   take term (i, j, value) and store it as (j, i, value)
}
```

For example

```
\circ (0, 0, 15) \rightarrow (0, 0, 15)
```

$$\circ$$
 (0, 3, 22) \rightarrow (3, 0, 22)

$$\circ$$
 (0, 5, -15) \rightarrow (5, 0, -15)

$$\circ$$
 (1, 1, 11) \rightarrow (1, 1, 11)

o ...

#NZ = No. of non-zero elements = No. of entries

Complexity: O(#NZ)+O(#NZ*log(#NZ))+O(#NZ*log(#NZ)) →
 O(#NZ*log(#NZ))



Algorithm 2 for Transposing a Sparse Matrix

Algorithm 2

- Idea: Find all terms in column 0 and store them in row 0; find all terms in column 1 and store them in row 1, and so on.
- Pseudo code

for (j=0; j<colNum; j++)
for all term in column j
place the entry (i, j, value) in the next position of the output

• Example:

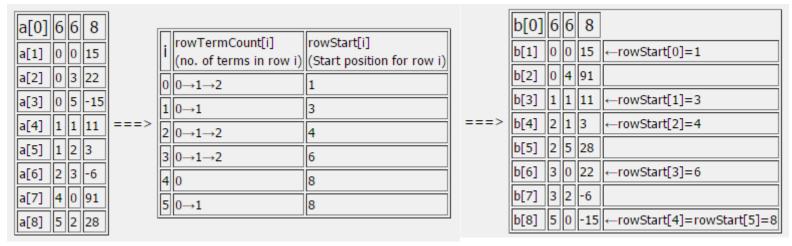
Complexity: O(#col*#NZ)



Algorithm 3 for Transposing a Sparse Matrix

Algorithm 3

- A fast algorithm that scans the term list only twice, as follows
 - Find number of terms in a row and then find the starting position of each row.
 - Fill the output matrix.
- Example



Complexity: O(#row + #NZ)