Data Structure and Algorithms

Chapter 6 Sparse matrix and General List

Dr. Zhiqiang Liu

School of Software and Microelectronics, Northwest Polytechnical University

Table of Contents

Next Section

6.1 Special matrix

1) Symmetric matrix:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 5 & 6 & 7 \\ 3 & 6 & 8 & 9 \\ 4 & 7 & 9 & 10 \end{bmatrix}$$

if $a_{ij} = a_{ji}\ 1 \le i, j \le n$

We only need to take the low triangle and diagonal elements into account. As a result, the space for these elements is $n(n+1)/2$. Suppose we use a 1D array $s[0(n+1)/2]$ to save these values, the corresponding address k in 1D array for the element a_{ij} is:

6.1 Special matrix

Symmetric matrix

0-th row

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ a_{n-1,1} & a_{n-1,2} & a_{n-1,3} & \cdots & a_{n-1,n} \\ a_{n,1} & a_{n,2} & a_{n,3} & \cdots & a_{n,n} \end{bmatrix}$$

$i \rightarrow$

0-th column

$$k = \begin{cases} \dfrac{i(i-1)}{2} + j & \text{if } i \geq j \\ \dfrac{j(j-1)}{2} + i & \text{if } i < j \end{cases}$$

j

k

0

a_{11}	a_{21}	a_{22}	a_{31}	a_{32}	a_{33}	...	...	...	$a_{n,1}$	$a_{n,2}$	$a_{n,3}$	...	...	$a_{n,n-1}$	$a_{n,n}$

6.1 Special matrix

2) Triangle matrix:

$$i \longrightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 3 & 0 & 0 & 0 & 0 \\ 4 & 5 & 6 & 0 & 0 & 0 \\ 7 & 8 & 9 & 10 & 0 & 0 \\ 11 & 12 & 13 & 14 & 15 & 0 \\ 16 & 17 & 18 & 19 & 20 & 21 \end{bmatrix}$$

j

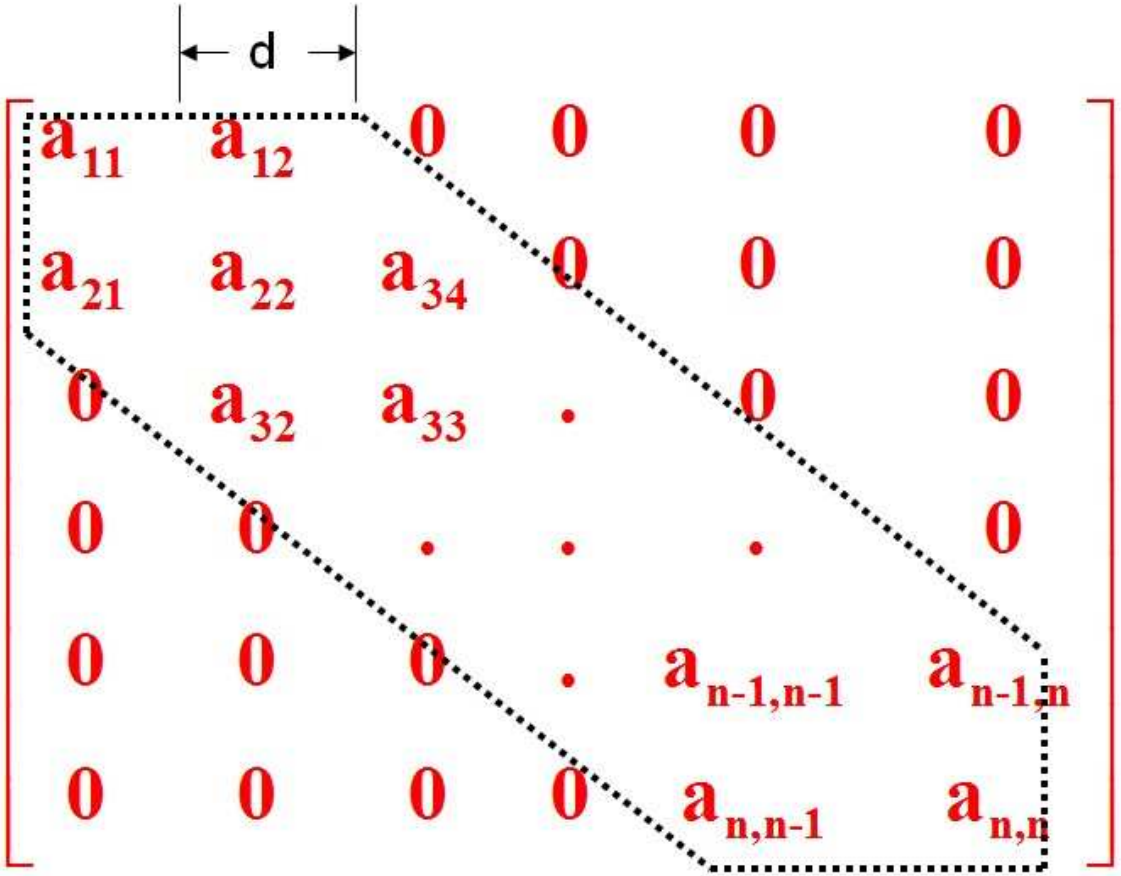
$$k = \begin{cases} \dfrac{i(i-1)}{2} + j & \text{if} \quad i >= j \\ Special\ value & \text{if} \quad i < j \end{cases}$$

0

1	2	3	4	5	6	...	...	...	16	17	18	...	...	20	21

6.1 Special matrix

3) Diagonal matrix:



Next Section

6.2 Sparse Matrix and ADT

$$A_{6\times7} = \begin{pmatrix} 0 & 0 & 0 & 22 & 0 & 0 & 15 \\ 0 & 11 & 0 & 0 & 0 & 17 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 39 & 0 \\ 91 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 28 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Row: m=6; Column: n=7;

The number of non-zero items: t=8

$$\text{Sparse factor } \delta = \frac{t}{m \times n}$$

6.2 Sparse Matrix and ADT

Storage of Sparse Matrix (1)

Sequential list

- Triple list
  - Dimensional sizes of matrix
  - Number of non-zero values
  - Triple
    - Row, Col, Value
- Requirement
  - Triple should be stored in sorted mode.

6.2 Sparse Matrix and ADT

Sequential list for Sparse Matrix

```
1  // ADT of SparseMatrix
2  template <class Type>
3  class SparseMatrix {
4        int Rows, Cols, Terms;
5        Trituple<Type> smArray[MaxTerms];
6    public:
7        SparseMatrix ( int MaxRow, int Maxcol );
8        SparseMatrix<Type> Transpose ( );
9        SparseMatrix<Type>
10                Add ( SparseMatrix<Type> b );
11       SparseMatrix<Type>
12           Multiply ( SparseMatrix<Type> b );
13 }
```

6.2 Sparse Matrix and ADT


```
1  // Class for Trituple
2  template<class Type> class SparseMatrix<Type>;
3
4  template<class Type> class Trituple{
5  friend class SparseMatrix <Type>
6  private:
7      int row, col;
8      Type value;
9  }
```

6.2 Sparse Matrix and ADT

$$A_{6\times7} = \begin{pmatrix} 0 & 0 & 0 & 22 & 0 & 0 & 15 \\ 0 & 11 & 0 & 0 & 0 & 17 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 39 & 0 \\ 91 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 28 & 0 & 0 & 0 & 0 \end{pmatrix}$$

A_{6x7}

	(row)	(col)	(value)
[0]	0	3	22
[1]	0	6	15
[2]	1	1	11
[3]	1	5	17
[4]	2	3	−6
[5]	3	5	39
[6]	4	0	91
[7]	5	2	28

6.2 Sparse Matrix and ADT

$$B_{7\times6} = \begin{pmatrix} 0 & 0 & 0 & 0 & 91 & 0 \\ 0 & 11 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 28 \\ 22 & 0 & -6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 17 & 0 & 39 & 0 & 0 \\ 15 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

B_{7x6}

	(row)	(col)	(value)
[0]	0	4	91
[1]	1	1	11
[2]	2	5	28
[3]	3	0	22
[4]	3	2	−6
[5]	5	1	17
[6]	5	3	39
[7]	6	0	16

6.2 Sparse Matrix and ADT

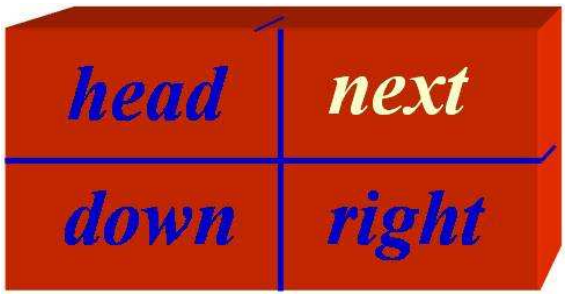
Storage of Sparse Matrix (2)

Cross linked list (Orthogonal list)

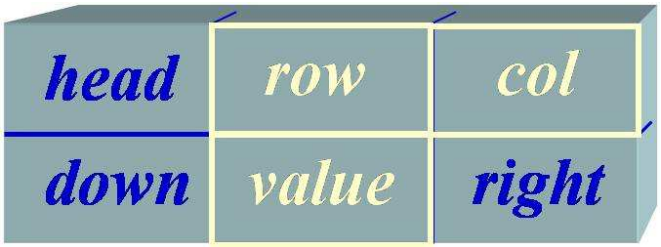
- Head node
  - next, down, right
- Element node
  - row, col, value, right, down

6.2 Sparse Matrix and ADT

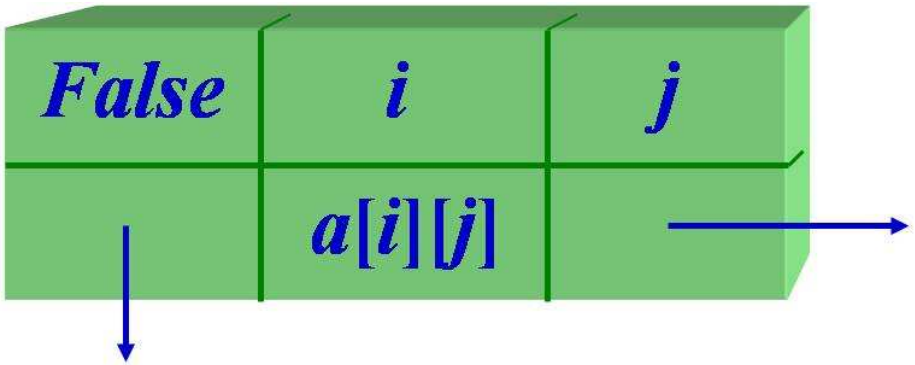
Cross linked list for Sparse Matrix



(a) Head node for each row



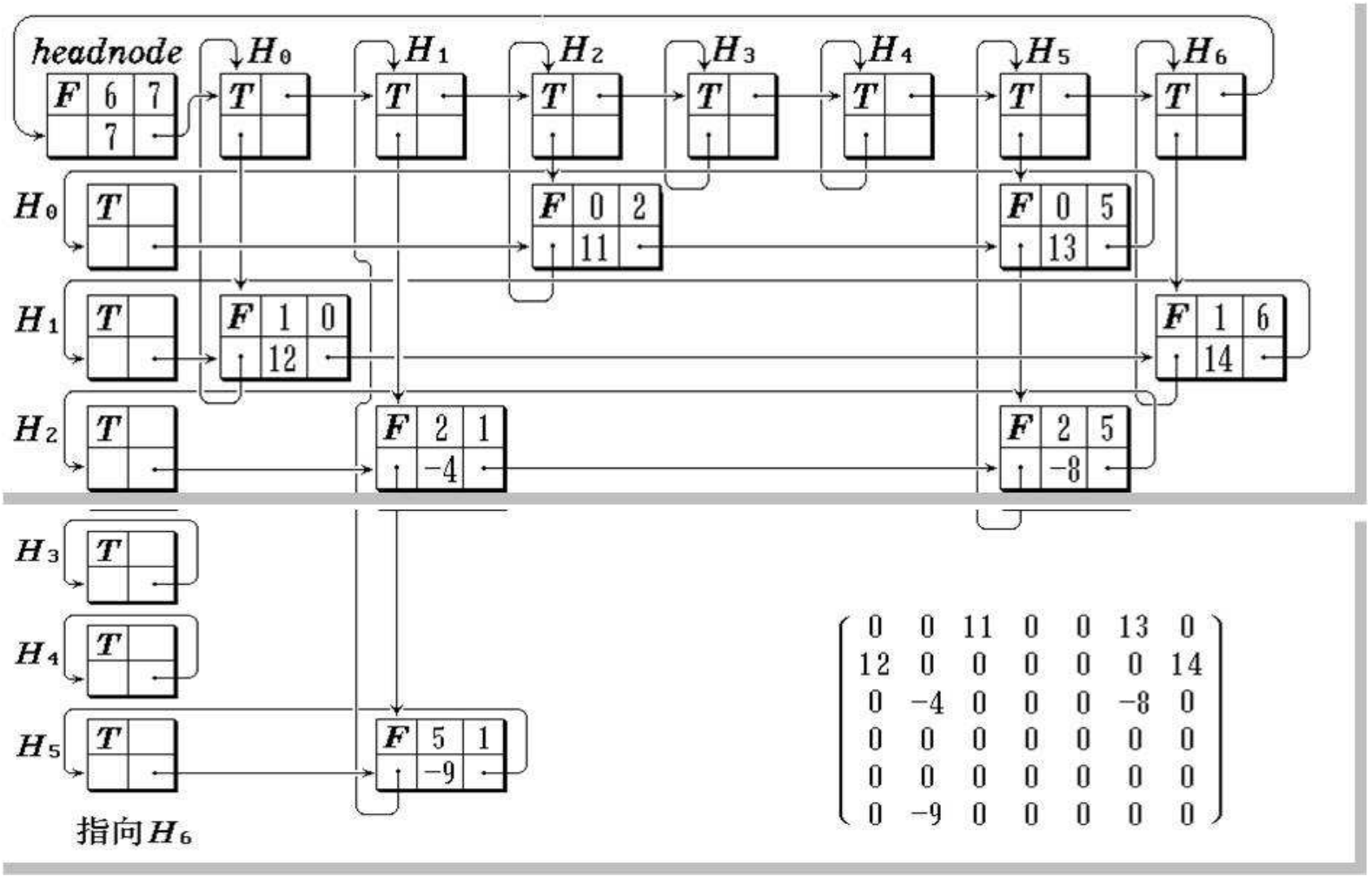
(b) Element node



(c) Element node with (i,j,a[i][j])

6.2 Sparse Matrix and ADT

Example



```
1  // Cross Linked List Class
2  enum Boolean{False, True};
3  struct Triple{int row, col; float value;};
4
5  class Matrix;
6
7  class MatrixNode{
8  friend class Matrix;
9  friend istream & operator >> (istream &, Matrix &);
10 private:
11      MatrixNode *down, *right;
12      Boolean head;
13      Union{
14          Triple triple;
15          MatrixNode *next;
16      }
17      MatrixNode(Boolean, Triple*);
18 }
19
20 typedef MatrixNode *MatrixNodePtr;
21
```

```
22 class Matrix{
23 friend istream & operator >> (istream &, Matrix &);
24 public:
25      ~Matrix ( );
26 private:
27      MatrixNode *headnode;
28 };
29
30 MatrixNode::MatrixNode(Boolean b, Triple *t)
31 {
32      head = b;
33      if(b)
34      {
35          right = next = this;
36      }
37      else
38          triple = *t;
39 }
```

```
1  // Initialization of Sparse Matrix using Orthogonal list
2  // Input: (6,6,7) (0,2,11) (0,5,13) (1,0,12) ... ...
3  istream & operator >> (istream &is, Matrix &matrix){
4       Triple s;
5       int p;
6       is >> s.row >> s.col >> s.value;
7       if(s.row > s.col)
8           p = s.row;
9       else
10          p = s.col;
11      matrix.headnode = new MatrixNode(False, &s);
12      if(!p ){
13         matrix.headnode->right = matrix.headnode;
14         return is;
15     }
16     MatrixNodePtr *H = new MatrixNodePtr(p);
17     for(int i=0; i<p; i++)
18         H[i]=new MatrixNode(True, 0);
19     int CurrentRow=0;
20     MatrixNode *last = H[0];
21     for(i=0; i<s.value; i++){
```

```
22          Triple t;
23          is>>t.row>>t.col>>t.value;
24          if(t.row > CurrentRow){
25              last->right = H[CurrentRow];
26              CurrentRow = t.row;
27              last = H[CurrentRow];
28          }
29          last = last->right = new MatrixNode(False, &t);
30          H[t.col]->next = H[t.col]->next->down = last;
31      }
32      last->right = H[CurrentRow];
33      for(i=0; i<s.col; i++)
34          H[i]->next->down = H[i];
35      for(i=0; i<p-1; i++)
36          H[i]->next = H[i+1];
37      H[p-1]->next = matrix.headnode;
38      matrix.headnode->right = H[0];
39      delete[]H;
40      return is;
41  }
```

Next Section

6.3 Transposition and Multiplication of Sparse Matrix

- Transpose a spare matrix
  - Generic method
  - Fast method
- Principle
  - A transpose to B, it means $b_{ij}=a_{ji}$, for i,j

6.3 Transposition and Multiplication of Sparse Matrix

A_{6x7}

	(row)	(col)	(value)
[0]	0	3	22
[1]	0	6	15
[2]	1	1	11
[3]	1	5	17
[4]	2	3	−6
[5]	3	5	39
[6]	4	0	91
[7]	5	2	28

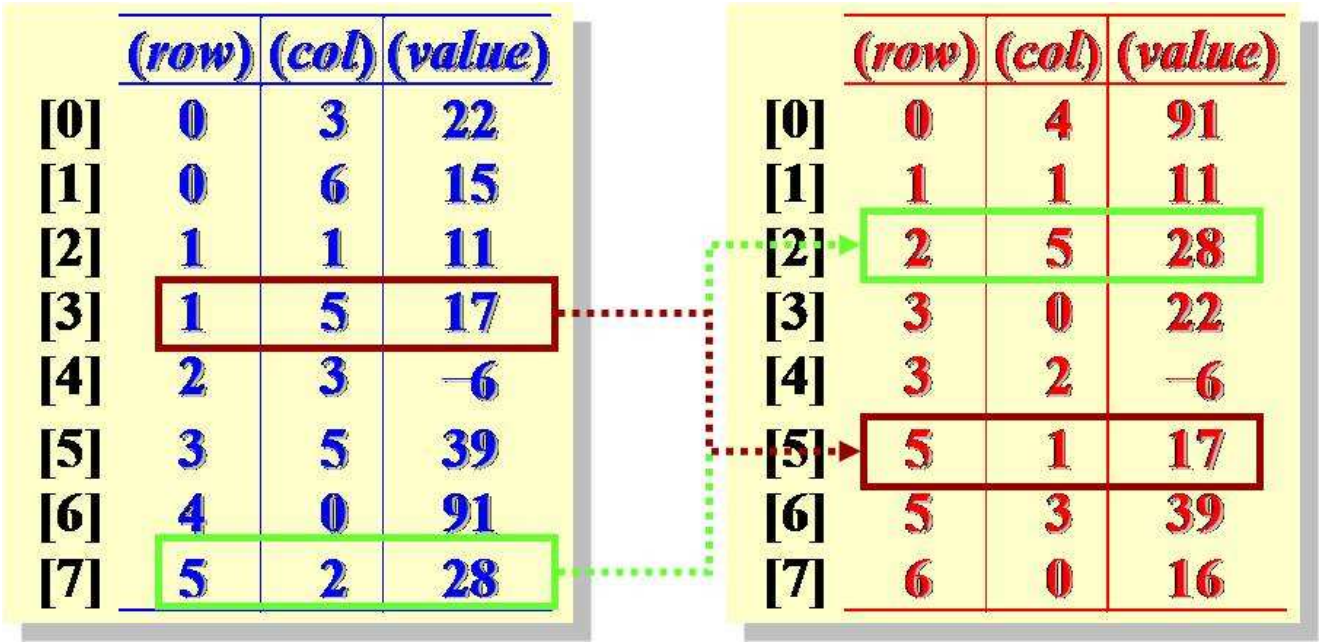
B_{7x6}

	(row)	(col)	(value)
[0]			
[1]			
[2]			
[3]			
[4]			
[5]			
[6]			
[7]			

```
1  // Generic Transposition algorithm
2  template <class Type> SparseMatrix<Type>
3  SparseMatrix<Type>::Transpose( )
4  {
5      SparseMatrix<Type> b(Cols,Rows);
6      b.Rows = Cols; b.Cols = Rows;
7      b.Terms = Terms;
8      if(Terms > 0)
9      {   int CurrentB = 0;
10         for(int k=0; k<Cols; k++)
11             for(int i=0; i<Terms; i++)
12               if( smArray[i].col==k){
13                   b.smArray[CurrentB].row=k;
14                   b.smArray[CurrentB].col=
15                        smArray[i].row;
16                   b.smArray[CurrentB].value=
17                        smArray[i].value;
18                   CurrentB++;
19               }
20     }  return b;
21 }
```

Problem and thinking...

- The complexity is O (Cols* Terms)
- Why?
- How to improve the algorithm?
- Strategy



	(row)	(col)	(value)
[0]	0	3	22
[1]	0	6	15
[2]	1	1	11
[3]	1	5	17
[4]	2	3	−6
[5]	3	5	39
[6]	4	0	91
[7]	5	2	28

	(row)	(col)	(value)
[0]	0	4	91
[1]	1	1	11
[2]	2	5	28
[3]	3	0	22
[4]	3	2	−6
[5]	5	1	17
[6]	5	3	39
[7]	6	0	16

6.3 Transposition and Multiplication of Sparse Matrix

$$A_{6\times7} = \begin{pmatrix} 0 & 0 & 0 & 22 & 0 & 0 & 15 \\ 0 & 11 & 0 & 0 & 0 & 17 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 39 & 0 \\ 91 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 28 & 0 & 0 & 0 & 0 \end{pmatrix}$$

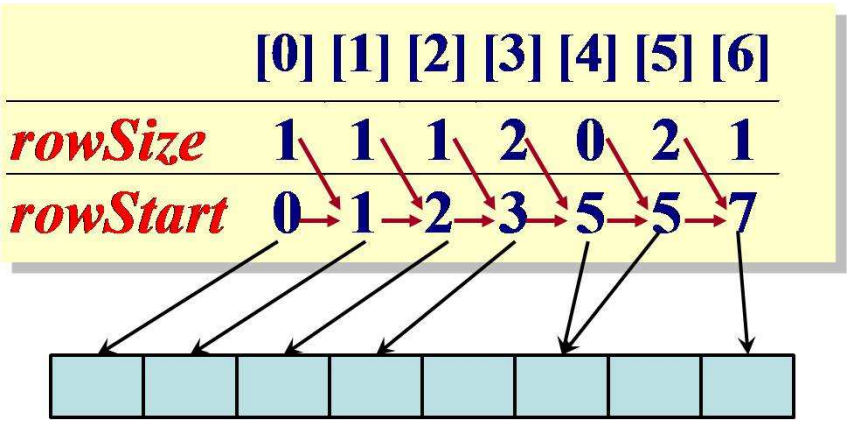
	(row)	*(col)*	*(value)*
[0]	0	3	22
[1]	0	6	15
[2]	1	1	11
[3]	1	5	17
[4]	2	3	–6
[5]	3	5	39
[6]	4	0	91
[7]	5	2	28

	[0]	[1]	[2]	[3]	[4]	[5]	[6]
rowSize	1	1	1	2	0	2	1
rowStart	0	1	2	3	5	5	7

Number of non-zero element in each column

Position of the first non-zero element in each column

6.3 Transposition and Multiplication of Sparse Matrix



	(row)	(col)	(value)
[0]	0	4	91
[1]	1	1	11
[2]	2	5	28
[3]	3	0	22
[4]	3	2	-6
[5]	5	1	17
[6]	5	3	39
[7]	6	0	16

```
1  for(int i=0;i<Cols;i++) rowSize[i]=0;
2   for(i=0;i<Terms;i++)
3         rowSize[smArray[i].col]++;
4   rowStart[0]=0;
5   for(i=1;i<Cols;i++)
6         rowStart[i]=rowStart[i-1]+rowSize[i-1];
```

```
1  // Fast Transposition algorithm
2  template <class Type>
3  SparseMatrix<Type>
4  SparseMatrix<Type>::FastTranspos( ){
5      int *rowSize = new int[Cols];
6      int *rowStart = new int[Cols];
7      SparseMatrix<Type> b(Cols,Rows);
8
9      b.Rows = Cols;
10     b.Cols = Rows;
11     b.Terms = Terms;
12     if(Terms > 0){
13         for(int i=0;i<Cols;i++)
14             rowSize[i]=0;
15
16         for(i=0;i<Terms;i++)
17             rowSize[smArray[i].col]++;
18
19         rowStart[0]=0;
20         for(i=1; i<Cols; i++)
21             rowStart[i]=rowStart[i-1]+rowSize[i-1];
```

```
22
23          for(i=0; i<Terms; i++)
24          {
25              int j=rowStart[smArray[i].col];
26              b.smArray[j].row=smArray[i].col;
27              b.smArray[j].col=smArray[i].row;
28              b.smArray[j].value=smArray[i].value;
29              rowStart[smArray[i].col]++;
30          }
31      }
32      delete []rowSize;
33      delete []rowStart;
34      return b;
35  }
```

Multiplication of Sparse Matrix

Theory:

$$C_{m\times l} = A_{m\times n} \times B_{n\times l}$$

$$C[i][j] = \sum_{k=0}^{n-1} A[i][k] * B[k][j]$$

$$i = 0,\ldots,m-1; j = 0,\ldots,l-1.$$

$$A = \begin{pmatrix} 10 & 0 & 5 & 7 \\ 2 & 1 & 0 & 0 \\ 3 & 0 & 4 & 0 \end{pmatrix} \quad B = \begin{pmatrix} 2 & 0 \\ 4 & 8 \\ 0 & 14 \\ 3 & 5 \end{pmatrix} \quad C = \begin{pmatrix} 41 & 105 \\ 8 & 8 \\ 6 & 56 \end{pmatrix}$$

6.3 Transposition and Multiplication of Sparse Matrix

Matrix multiplication

$$A_{m\times n}=\begin{pmatrix} a_1 \\ a_2 \\ \\ a_m \end{pmatrix} \qquad B_{n\times l}=(\beta_1,\beta_2,\ldots\beta_l)$$

$$C_{m\times l}=A_{m\times n}\times B_{n\times l}=\begin{pmatrix} \alpha_1 B \\ \alpha_2 B \\ \vdots \\ \alpha_m B \end{pmatrix}=\begin{pmatrix} \alpha_1\beta_1 & \alpha_1\beta_2 & \cdots & \alpha_1\beta_l \\ \alpha_2\beta_1 & \alpha_2\beta_2 & \cdots & \alpha_2\beta_l \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_m\beta_1 & \alpha_m\beta_2 & \cdots & \alpha_m\beta_l \end{pmatrix}$$

Next Section

6.4 General List and ADT

- Definition
  - The general list is a special list

$$LS = (\alpha_0, \alpha_1, \alpha_2, ..., \alpha_{n-1})$$

    LS is the name of the general List, α_i is an element, whose type is atom or general list.
  - n is the length of LS
- Concepts
  - Head: the first element
  - Tail: the sub general list derived from the other elements except the first one.

Characteristics of General List

- Ordered
- Length
  - Limited
- Depth
  - Hierarchical
- Recursive
- Shareable

Examples:

A = ( )

B = (6, 2)

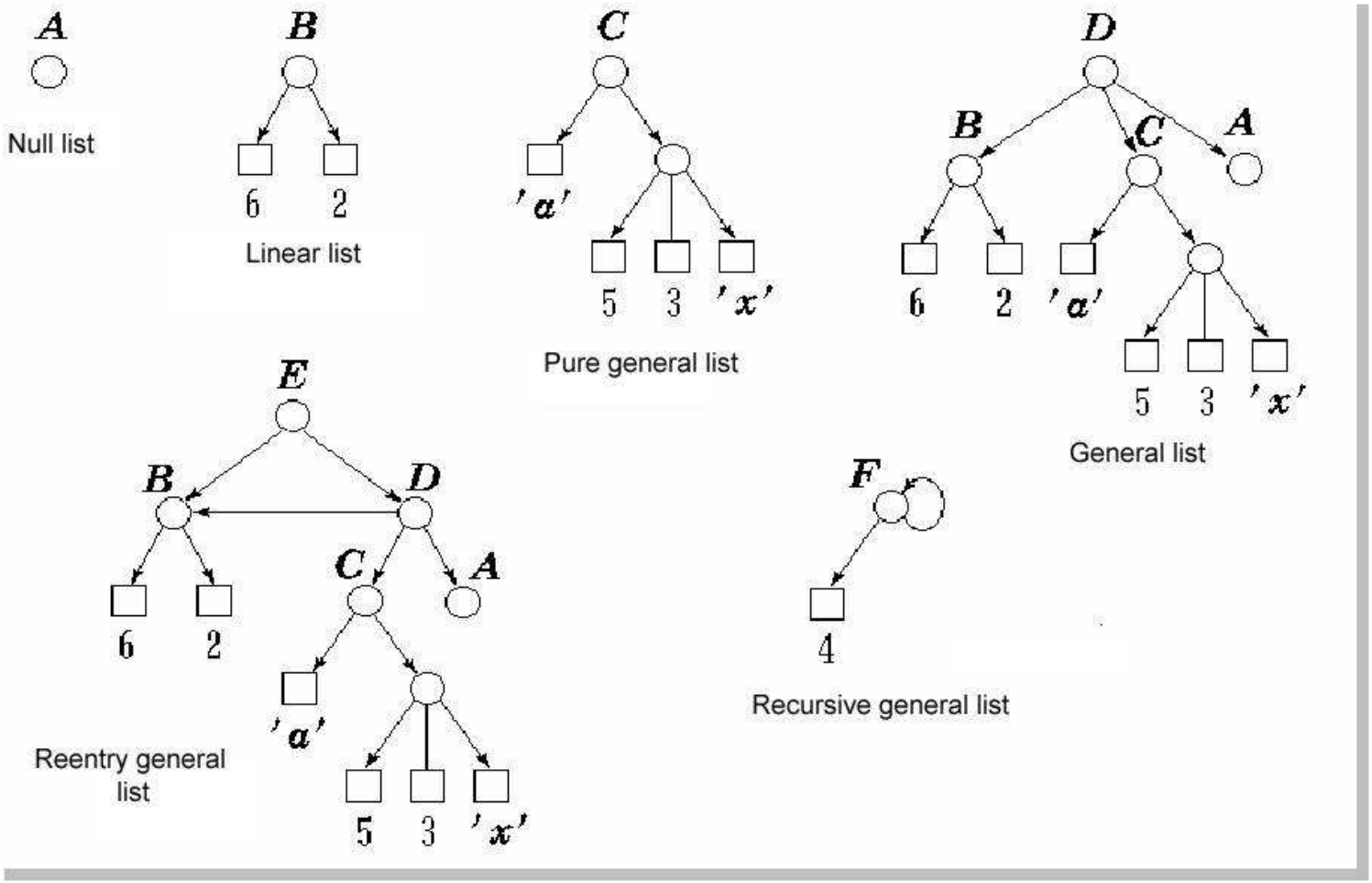
C = (‘a’, (5, 3, ‘x’))

D = (B, C, A)

E = (B, D)

F = (4, F)

Examples

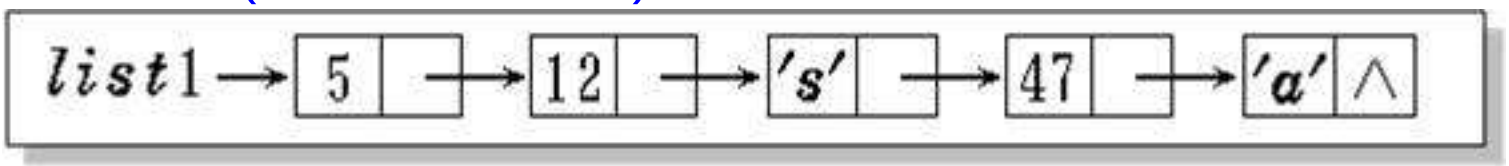

A
Null list
B
6
2
Linear list
C
'a'
5
3
'x'
Pure general list
D
B
C
A
6
2
'a'
5
3
'x'
General list
E
B
D
C
A
6
2
'a'
5
3
'x'
Reentry general list
F
4
Recursive general list

6.4 General List and ADT

Representation of General List

- Simple general list
  - Only contain atom elements

list1=(5, 12, ‘s’, 47, ‘a’)



Head(list1)=5

Tail(list1)=(12, ‘s’, 47, ‘a’)

Head(Tail(list1))=12

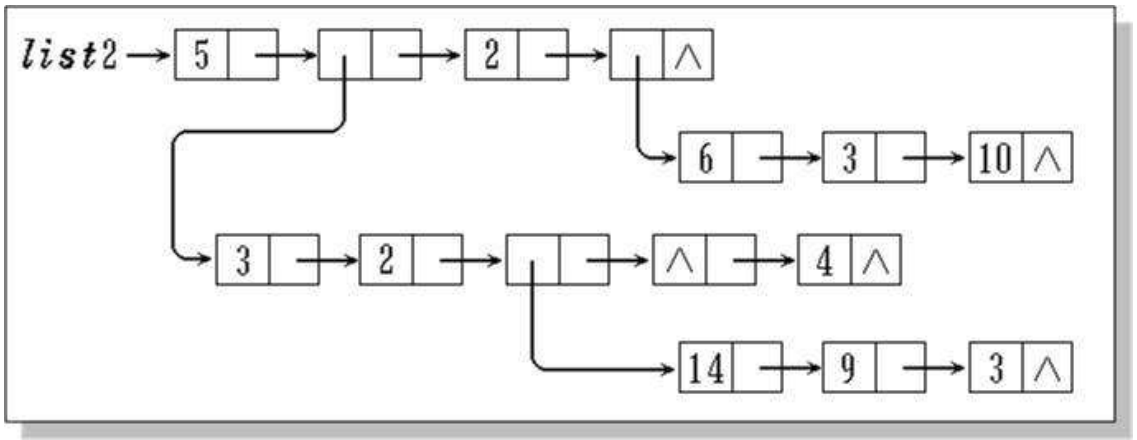
Tail(Tail(list1))=(‘s’, 47, ‘a’)

6.4 General List and ADT

Representation of General List

- Complex general list

list2=(5, (3, 2, (14, 9, 3), (), 4), 2, (6, 3, 10))



Head(list2)=5

Tail(list2)=((3, 2, (14, 9, 3), (), 4), 2, (6, 3, 10))

Head(Tail(list2))=(3, 2, (14, 9, 3), (), 4)

Head(Head(Tail(list2)))=3

6.4 General List and ADT

Node specification for General List

- Flag: utype
  - 0: head node
  - 1: atom type node (integer)
  - 2: atom type node (character)
  - 3: node of sub general list
- Value: value
  - utype=0
    Citation of general list
  - Integer or character
  - Pointer to the head node of the sub general list

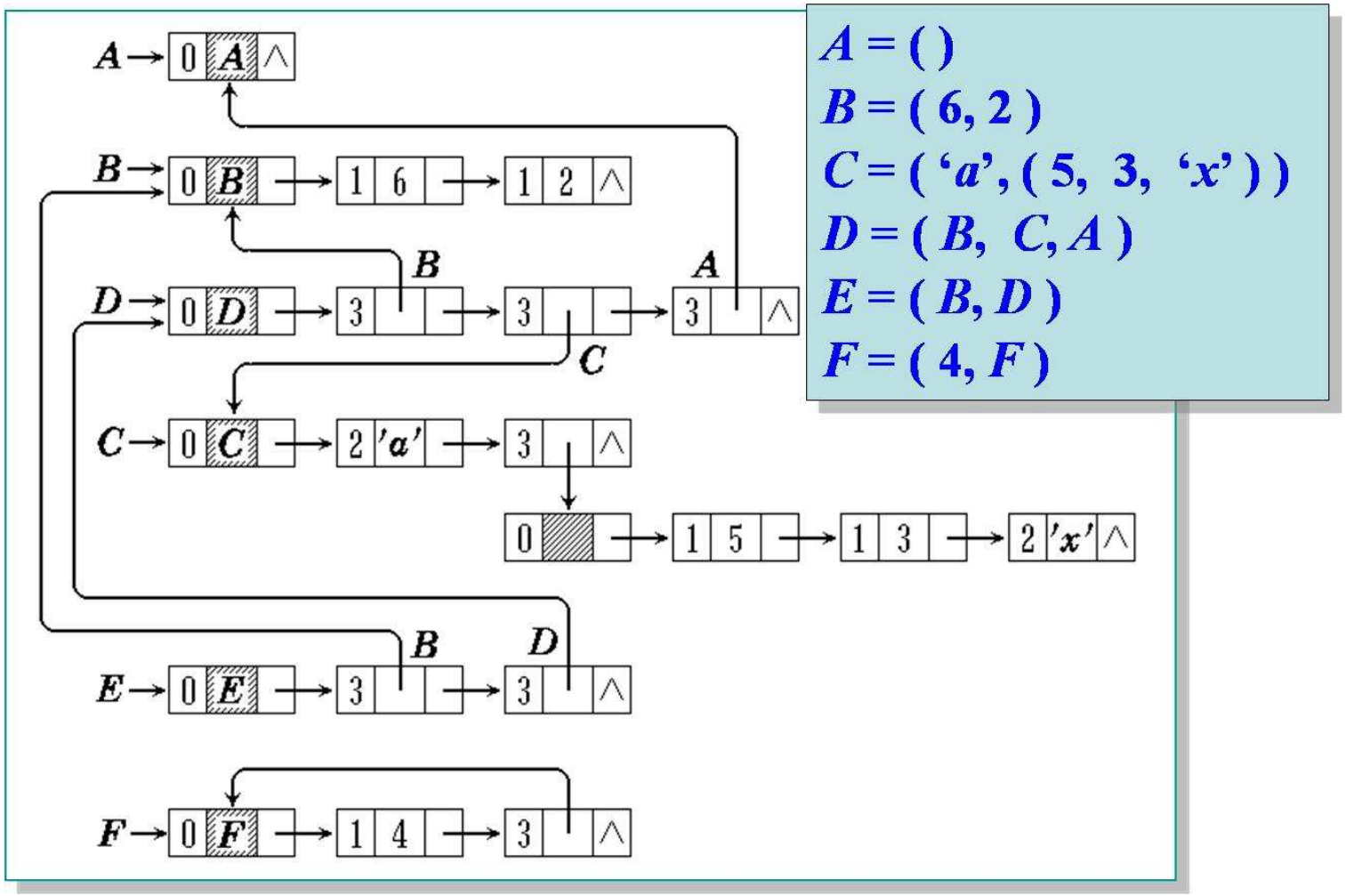
6.4 General List and ADT

Node specification for General List

- Tail pointer: tlink
  - utype=0
    Point to the first element node
  - utype !=0
    Point to the next element node at the same level

6.4 General List and ADT

Linked list (with head node) for general list



General List Class

```
1  // Node Class
2  #define HEAD  0
3  #define INTGR 1
4  #define CH  2
5  #define LST 3
6  class GenList;
7  class GenListNode{
8  friend class Genlist;
9  private:
10     int utype;
11     GenListNode *tlink;
12     union{
13         int ref;              //utype = 0, head node
14         int intgrinfo;        //utype = 1, integer
15         char charinfo;        //utype = 2, character
16         GenListNode *hlink; //utype = 3, sub list node
17     } value;
18 public:
19     GenListNode&Info(GenListNode *elem);
20     int nodetype(GenListNode *elem)
```

```
21      {
22          return elem->utype;
23      }
24      void setInfo(GenListNode *elem, GenListNode *x);
25  };
```

```
1  //General list Class
2  class GenList {
3  private:
4      GenListNode *first;
5      // Copy a Non-Recursive GL
6      GenListNode *Copy(GenListNode *ls);
7      int depth(GenListNode *ls);
8      int equal(GenListNode *s, GenListNode *t);
9      void Remove(GenListNode *ls);
10 public:
11     Genlist( );
12     ~GenList( );
13     GenListNode& Head( );
14     GenListNode& Tail( );
15     GenListNode *First( );
16     GenListNode *Next(GenListNode *elem);
17     void Push(GenListNode & x);
18     // Return a new GL with head x and tail list
19     GenList &Addon(GenList& list, GenListNode& x);
20     // Set x as head
21     void setHead(GenListNode& x);
```

```
22      void setNext(GenListNode *elem1,
23          GenListNode *elem2);
24      // Set list as tail
25      void setTail(GenList& list);
26      void Copy(const GenList& l);
27      int depth( );
28      int Createlist(GenListNode* ls, char* s);
29  }
```

Next Section

Implementation and Application of General List

```
1  //Get the value of node
2  GenListNode& GenListNode::
3  Info (GenListNode* elem ){
4       GenListNode * pitem = new GenListNode;
5       pitem->utype = elem->utype;
6       pitem->value = elem->value;
7       return * pitem;
8  }
9  //Set the value of node
10 void GenListNode::setInfo(GenListNode* elem,
11      GenListNode& x){
12     elem->utype = x->utype;
13     elem->value = x->value;
14 }
15 //Constructor
16 Genlist::GenList( ){
17     GenListNode *first = new GenListNode;
18     first->utype = 0; first->ref = 1;
19     first->tlink = NULL;
20 }
```

```
21
22 //Get the head of general list
23 GenListNode&GenList::Head( ){
24     if(first->tlink == NULL ){
25      cout << “Invalid operation to the head ”! << endl;
26      exit (1);
27     }
28     else{
29        GenListNode *temp = new GenListNode;
30        temp->utype = frist->tlink->utype;
31        temp->value = frist->tlink->value;
32        return *temp;
33     }
34 }
35
36 //Get the tail of the general list
37 GenListNode&GenList::Tail( ){
38     if(first->tlink == NULL){
39        cout << “Invalid operation to the Tail”! << endl;
40        exit (1);
41     }
42     else{
```

```
43          GenListNode *temp = new GenListNode;
44          temp->first->tlink = first->tlink->tlink;
45          return *temp;
46      }
47  }
48
49  //Add the node x at the front of general list
50  void GenList::Push (GenListNode& x){
51      if(first->tlink == NULL) first->tlink=x;
52      else{
53          x->tlink=first->tlink;first->tlink=x;
54      }
55  }
56
57  //Return a new list head=x, tail=list
58  GenList&GenList::Addon(GenList &list, GenListNode &x){
59      GenList *newlist = new GenList;
60      newlist->first= Copy(list.first );
61      x->tlink = newlist->first->tlink;
62      newlist->first->tlink = x;
63      return *newlist;
64  }
```

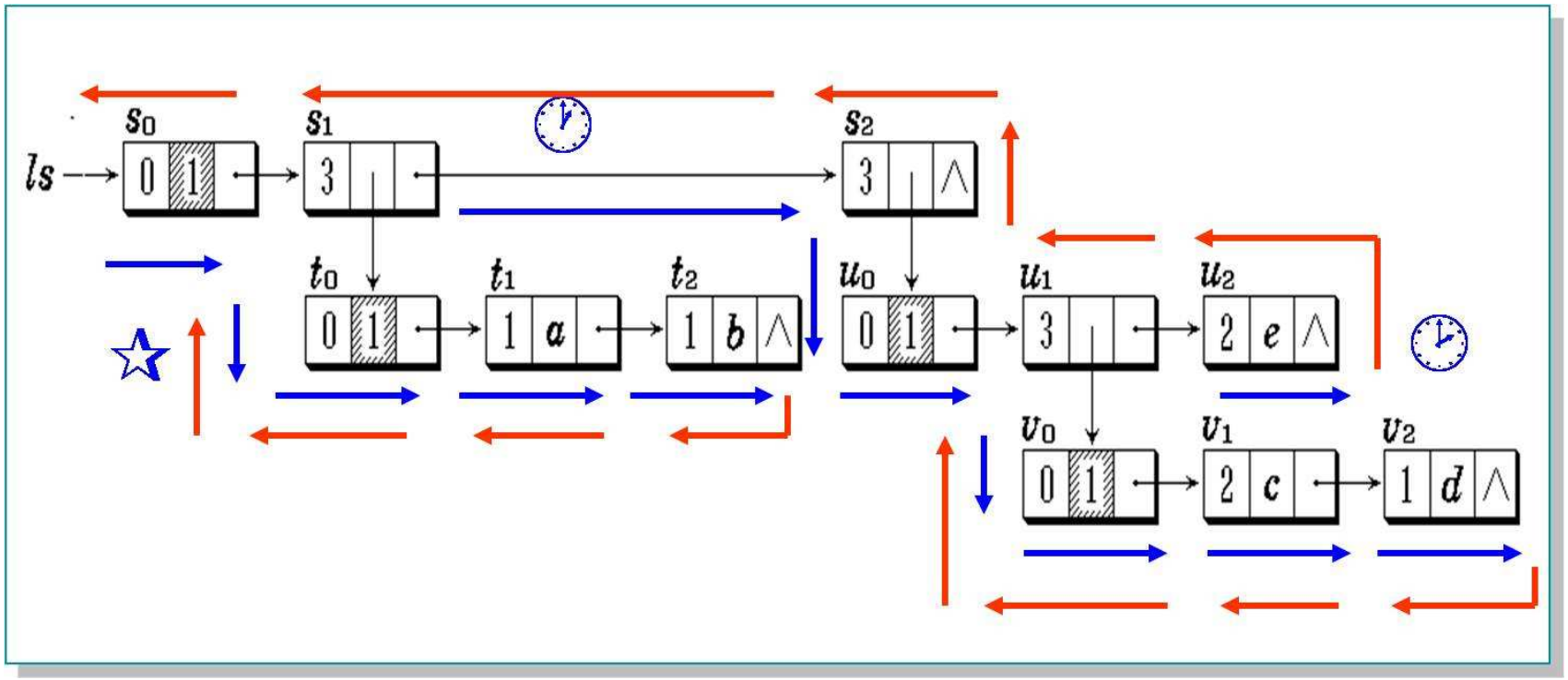
Recursive algorithm for general list

```
1  // Duplication: Non-Recursive GL and Non-Share
2  void GenList::Copy(const GenList &l ){
3      first = Copy(l.first);
4  }
5
6  GenListNode* GenList::Copy(GenListNode *ls)
7  {
8    GenListNode *q = NULL;
9    if (ls !=NULL)
10   {
11      q = new GenListNode;
12      q->utype = ls->utype;
13      switch( ls->utype ){
14        case HEAD:
15          q->value.ref = ls->value.ref;
16          break;
17        case INTGR:
18          q->value.intgrinfo = ls->value.intgrinfo;
19          break;
20        case CH:
```

```
21            q->value.charinfo = ls->value.charinfo;
22            break;
23          case LST:
24            q->value.hlink = Copy (ls->value.hlink);
25            break;
26        }
27        q->tlink = Copy(ls->tlink);
28    }
29    return q;
30  }
```

6.5 Implementation and Application of General List

Procedure of duplication



recursive call

return

6.5 Implementation and Application of General List

Depth of General List

$$Depth(LS) = \begin{cases} 1 & \text{if LS is NULL} \\ 0 & \text{if LS is an atom} \\ l + \max\limits_{0 \le i \le n-1} & otherwise, n \ge 1 \end{cases}$$

Example:

E (B (a, b), D (B (a, b), C (u, (x, y, z)), A ( )))

$Depth(E) = 1 + Max\{Depth(B), Depth(D)\}$

Depth (B) = 1+Max { Depth (a), Depth (b) } = 1

Depth (D) = 1+Max { Depth (B), Depth (C), Depth (A)}

Depth (C) = 1+Max { Depth (u), Depth ((x, y, z)) }

6.5 Implementation and Application of General List

Depth (A) = 1

Depth (u) = 0

Depth ((x, y, z)) = 1+Max Depth (x),

Depth (y), Depth (z) = 1

Depth (C) = 1+Max Depth (u), Depth ((x, y, z))

= 1+Max 0, 1 = 2

Depth (D) = 1+Max Depth (B), Depth (C), Depth (A)

= 1+Max 1, 2, 1 = 3

Depth (E) = 1+Max Depth (B), Depth (D)

= 1+Max 1, 3 = 4

E (B (a, b), D (B (a, b), C (u, (x, y, z)), A ( )))

```
1  int GenList::depth(GenListNode *ls){
2      if(ls->tlink == NULL)
3          return 1;
4      GenListNode *temp = ls->tlink;
5      int m = 0;
6      while(temp != NULL){
7          if(temp->utype == LST){
8              int n = depth(temp->value.hlink );
9              if(m < n)
10                 m = n;
11         }
12         temp = temp->tlink;
13     }
14     return m+1;
15 }
16 int GenList::depth( ){
17     return depth(first);
18 }
```

Representation of Multinomial

$$P(x,y,z) = x^{10}y^3z^2 + 2x^8y^3z^2 + 3x^8y^2z^2 + x^4y^4z^1 + 6x^3y^4z^1 + 2yz$$

- Multi-item Sequential list

coef1	expx1	expy1	expz1
coef2	expx2	expy2	expz2
coef3	expx3	expy3	expz3
.....			
coefn	expxn	expyn	expzn

- Single linked list

coef	expx	expy	expz	link

Variable separation

$$
\begin{aligned}
P(x,y,z) &= x^{10}y^3z^2 + 2x^8y^3z^2 + 3x^8y^2z^2 + x^4y^4z^1 + 6x^3y^4z^1 + 2yz \\
&= (x^{10}y^3 + 2x^8y^3 + 3x^8y^2)z^2 + (x^4y^4 + 6x^3y^4 + 2y)z \\
&= Az^2 + Bz \\
A(x,y) &= x^{10}y^3 + 2x^8y^3 + 3x^8y^2 \\
B(x,y) &= x^4y^4 + 6x^3y^4 + 2y \\
A(x,y) &= (x^{10} + 2x^8)y^3 + 3x^8y^2 = Cy^3 + Dy^2 \\
B(x,y) &= (x^4 + 6x^3)y^4 + 2y = Ey^4 + Fy \\
C(x) &= x^{10} + 2x^8 \quad D(x) = 3x^8 \quad E(x) = x^4 + 6x^3 \quad F(x) = 2
\end{aligned}
$$

$$P(x,y,z) = Az^2 + Bz \Rightarrow Pz(A-2, B-1)$$

$$A = Cy^3 + Dy^2 \Rightarrow Ay(C-3, D-2)$$

$$B = Ey^4 + Fy \Rightarrow By(E-4, F-1)$$

$$C = x^{10} + 2x^8 \Rightarrow Cx(1-10, 2-8)$$

$$D = 3x^8 \Rightarrow Dx(3-8)$$

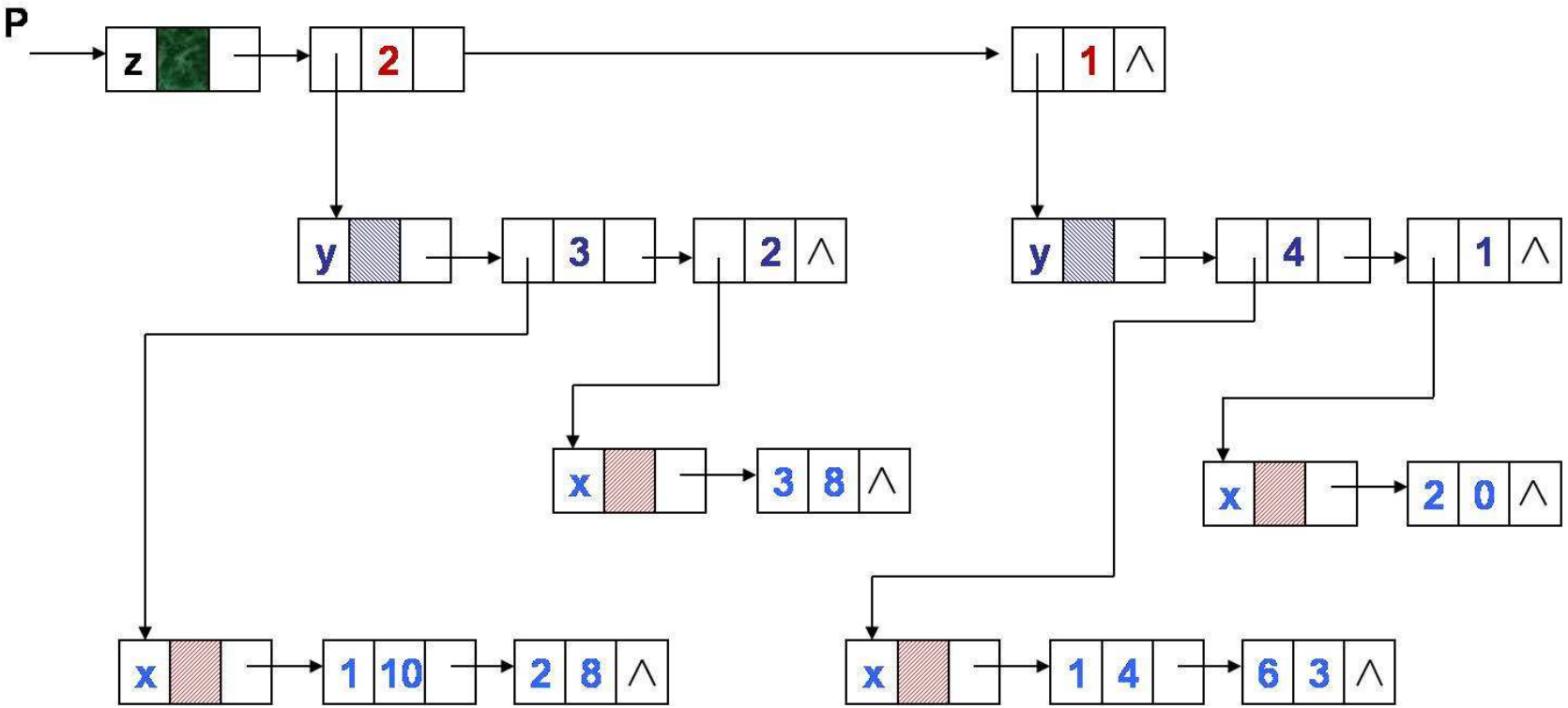
$$E = x^4 + 6x^3 \Rightarrow Ex(1-4, 6-3)$$

$$F = 2 \Rightarrow Fx(2-0)$$

Pz(Ay(Cx(1-10),2-8)-3,Dx(3-8)-2)-2,
By(Ex(1-4,6-3)-4,Fx(2-0)-1)-1)

```
1  //Node declaration
2
3  enum Triple{var,ptr,num};
4  //var-head of list;ptr-head of sub-list;num-atom
5  class PolyNode {
6        PolyNode *tlink;
7        int exp;
8        //var-head of list; ptr-head of sub-list;num-atom
9        Triple tag;
10
11       union{
12         char vrble;
13         PolyNode *hlink;
14         double coef;
15     };
16 };
```

General List for Multinomial



Next Section

6.6 Summary, Homework and QUIZ

6.6 Summary, Homework and QUIZ

1、广义表（x，（a，b，c））的表尾是什么?
2、画出如下广义表的存储结构图:
（（），（（（））），（（（））））），求它的深度和长度。
3、用扩展线性表表示具有共享结构的广义表
（（（b，c），d），（a），（（a），（（b，c），d）），e，（））
的存储结构图。

(Homework title: General List)
4、Determine whether two general list are equal with Recursive algorithm
5、 Operations of Multinomial： Addition, Subtraction, Multiplication