Strutture Dati

Lezione 4
Le strutture dati
Array, Matrice

Oggi parleremo di …

- Tipo di dati astratto *Array*
  - specifica
  - rappresentazione
  - esempio
- Tipo di dati astratto *Matrice_Sparsa*
  - specifica
  - rappresentazione
  - operazioni

Il tipo di dati astratto *Array*

- Un **array** è un insieme di coppie <*indice*,*valore*> tali che ad ogni indice che è definito sia associato un determinato valore

Il tipo di dati astratto *Array*

```
Struttura Array
   oggetti: un insieme di coppie <indice, valore> dove per ogni valore
        dell'indice c'è un valore dell'insieme elemento. Indice è un
        insieme ordinato e finito ad una o più dimensioni, ad esempio
        {0,…,n-1} per una dimensione,{(0,0),(0,1),(0,2),(1,0),(1,1),
        (1,2),(2,0),(2,1),(2,2)} per due dimensioni

   funzioni: per ogni A ∈ Array, i ∈ indice, x ∈ elemento, j ∈ Integer

   Array Create(j, list)      ::=   ritorna un array di j dimensioni
                                    dove list è una serie di j valori,
                                    il cui i-esimo elemento è la
                                    grandezza della i-esima
                                    dimensione. Gli elementi sono
                                    indefiniti
   Elemento Retrieve(A,i)     ::=   if (i ∈ indice) return l'elemento
                                    associato al valore dell'indice i
                                    nell'array A,  else return errore
   Array Store(A,i,x)         ::=   if(i ∈ indice) return un array
                                    identico all'array A ad eccezione
                                    della nuova coppia <i,x> ora
                                    inserita, else return errore
end Array
```

Esempio: Il crivello di Eratostene

- Stampare tutti i numeri primi minori di 1000
- Utilizziamo un array A di valori booleani (0,1):
  - A[i] = 1 se i è primo
  - A[i] = 0 altrimenti
- Si pongono a 0 tutti gli elementi di A multipli di i, con i compreso tra 2 e N/2
- Si stampano tutti gli interi risultati primi

Esempio: Il crivello di Eratostene

```
void genera(int A[], int N)
{
  int i,j;

  for(i=1; i<=N; i++)   A[i]=1;
  for(i=2; i<= N/2; i++)
     for(j=2; j<=N/i; j++)
          A[i*j]=0;
  for(i=1; i<=N; i++)
     if(A[i])  printf("%4d", i);
}
```

Osservazioni…

- Ogni elemento dell'array è indirizzabile in tempo costante
- E' necessario conoscere apriori la dimensione dell'array
- Gli array hanno una corrispondenza diretta con la memoria centrale
  - per accedere ad una parola di memoria basta specificarne l'indirizzo
- L'array viene utilizzato per rappresentare valori in forma di tabella, organizzati in righe e colonne
- L'array monodimensionale corrisponde al vettore matematico e l'array bidimensionale alla matrice

7

Il tipo di dati astratto *Matrice_Sparsa*

- Una **matrice** viene rappresentata come un array bidimensionale definito come a[Max_Righe, Max_Colonne]

$$\begin{bmatrix} 2 & 0 & -1 & 5 \\ 1 & 4 & 5 & 12 \\ 10 & 2 & 14 & 0 \end{bmatrix}$$

- Nelle **matrici sparse** in cui molti valori sono nulli, si ha un notevole spreco di spazio
- E' necessaria una rappresentazione alternativa che memorizzi solo gli elementi non nulli

$$\begin{bmatrix} 2 & 0 & 0 & 1 & 0 \\ 0 & 1 & 6 & 0 & 0 \\ 4 & 0 & 0 & 1 & 7 \\ 0 & 0 & 9 & 0 & 0 \\ 1 & 0 & 5 & 8 & 0 \end{bmatrix}$$

8

Il tipo di dati astratto *Matrice_Sparsa*

```
Struttura Matrice_Sparsa
  oggetti: un insieme di triple <riga, colonna, valore> dove riga e
        colonna sono interi che formano una combinazione unica, e
        valore è un valore dell'insieme elemento
  funzioni: per ogni a,b ∈ Matrice_Sparsa, x ∈ elemento, i,j,
        max_righe, max_col ∈ indice

  Matrice_Sparsa Create(max_righe, max_col)   ::=    ritorna una
                Matrice_Sparsa che può contenere fino a max_righe x
                max_col elementi
  Matrice_Sparsa Trasponi(a)   ::=    ritorna la matrice ottenuta
                scambiando i valori delle righe e delle colonne
                di ogni tripla
  Matrice_Sparsa Somma(a,b)    ::=    if le dimensioni di a e b sono
                le stesse, ritorna la matrice ottenuta sommando gli
                elementi corrispondenti, cioè quelli che hanno gli
                stessi valori riga e colonna, else ritorna errore
  Matrice_Sparsa Prodotto(a,b) ::= if numero di colonne di a è uguale
                al numero di righe di b, ritorna la matrice d ottenuta
                moltiplicando a per b secondo la formula d[i][j]=
                Σa[i][k]*b[k][j], dove d[i][j] è l'elemento di riga i e
                colonnna j di d, else ritorna errore
end Matrice_Sparsa
```

Rappresentazione della matrice

- Ogni elemento viene rappresentato dalla tripla <riga, colonna, valore>
- Si usa un array di triple
  - indici delle righe in ordine crescente
  - indici delle colonne in ordine crescente

```
Matrice_Sparsa Create(max_righe, max _col)   ::=

#define MAX_TERMINI 101   /* num max elementi+1 */
typedef struct {
              int riga;
              int col;
              int valore;
              } termine;
termine a[MAX_TERMINI];
```

- a[0].riga contiene il numero delle righe
- a[0].col contiene il numero delle colonne
- a[0].valore il numero di elementi non nulli

10

Esempio

$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 2 & 0 & 0 & 1 & 0 \\ 0 & 1 & 6 & 0 & 0 \\ 4 & 0 & 0 & 1 & 7 \\ 0 & 0 & 9 & 0 & 0 \\ 1 & 0 & 5 & 8 & 0 \end{bmatrix} \end{array}$$

	riga	col	valore
a[0]	5	5	11
a[1]	0	0	2
a[2]	0	3	1
a[3]	1	1	1
a[4]	1	2	6
a[5]	2	0	4
a[6]	2	3	1
a[7]	2	4	7
a[8]	3	2	9
a[9]	4	0	1
a[10]	4	2	5
a[11]	4	3	8

11

Trasposizione di una matrice

- Bisogna scambiare le righe con le colonne
- L'elemento a[i][j] diventa l'elemento b[j][i] nella trasposta

```
per ogni riga i
   prendi l'elemento <i,j,valore> e memorizzalo
   come elemento <j,i,valore> nella matrice
   trasposta;
```

- Dove porre l'elemento <j,i,valore> nella trasposta?
- Per mantenere l'ordine è necessario spostare gli elementi

12

Trasposizione di una matrice

- Per determinare la posizione degli elementi nella matrice trasposta, possiamo utilizzare gli indici delle colonne

```
per tutti gli elementi nella colonna j
   poni l'elemento <i,j,valore> nella posizione
   <j,i,valore> nella matrice trasposta;
```

- Di conseguenza, poiché nella matrice originale le righe sono organizzate in ordine crescente, anche le colonne all'interno di ogni riga della matrice trasposta saranno ordinate in modo crescente

Trasposizione di una matrice

	riga	col	valore		riga	col	valore	
a[0]	5	5	11	b[0]	5	5	11	
a[1]	0	0	2	b[1]	0	0	2	i=0
a[2]	0	3	1	b[2]	0	2	4	
a[3]	1	1	1	b[3]	0	4	1	
a[4]	1	2	6	b[4]	1	1	1	i=1
a[5]	2	0	4	b[5]	2	1	6	i=2
a[6]	2	3	1	b[6]	2	3	9	
a[7]	2	4	7	b[7]	2	4	5	
a[8]	3	2	9	b[8]	3	0	1	i=3
a[9]	4	0	1	b[9]	3	2	1	
a[10]	4	2	5	b[10]	3	4	8	
a[11]	4	3	8	b[11]	4	2	7	i=4

Trasposizione di una matrice

```
void trasponi(termine a[], termine b[])
{
 int n, i, j, bcorrente;
 n = a[0].valore;            /* num elementi */
 b[0].riga = a[0].col;       /* riga di b = colonna di a */
 b[0].col = a[0].riga;       /* colonna di b = riga di a */
 b[0].valore = a[0].valore;
 if(n>0) {
       bcorrente = 1;
       for(i=0; i<a[0].col; i++)      /* trasponi per colonne */
          for(j=1; j<=n; j++)    /* trova elementi colonna corrente */
               if(a[j].col == i) {
                     b[bcorrente].riga = a[j].col;
                     b[bcorrente].col = a[j].riga;
                     b[bcorrente].valore = a[j].valore;
                     bcorrente++;
                     }
       }
}
```

Analisi dell'algoritmo **`trasponi`**

```
void trasponi(termine a[], termine b[])
{
 int n, i, j, bcorrente;
 n = a[0].valore;
 b[0].riga = a[0].col;
 b[0].col = a[0].riga;
 b[0].valore = a[0].valore;
 if(n>0) {
   bcorrente = 1;
   for(i=0; i<a[0].col; i++)
     for(j=1; j<=n; j++)
       if(a[j].col == i) {
         b[bcorrente].riga = a[j].col;
         b[bcorrente].col = a[j].riga;
         b[bcorrente].valore = a[j].valore;
         bcorrente++;
       }
   }
}
```

$= O(elementi)$ $= O(elementi \times col)$

$O(elementi \times colonne) = O(righe \times colonne^2)$

$O(elementi + colonne)$?

$O(righe \times colonne)$?

Trasposizione rapida

- Si determina il numero di elementi in ogni colonna della matrice originale
  - ci dà il numero di elementi in ogni riga della matrice trasposta
- Si determina la posizione di partenza di ogni riga nella matrice trasposta
- Si spostano gli elementi della matrice originale, uno alla volta, nelle loro esatte posizioni nella matrice trasposta

Trasposizione rapida

- Sia **`termini_riga`** il vettore che indica il numero di elementi di ogni riga
- Sia **`pos_iniziale`** il vettore che indica la posizione iniziale di ogni riga
- Il punto di partenza per la riga i (i>1) della matrice trasposta è **`pos_iniziale`**[i-1]+**`termini_riga`** [i-1]

Trasposizione rapida

	riga	col	valore
a[0]	5	5	11
a[1]	0	0	2
a[2]	0	3	1
a[3]	1	1	1
a[4]	1	2	6
a[5]	2	0	4
a[6]	2	3	1
a[7]	2	4	7
a[8]	3	2	9
a[9]	4	0	1
a[10]	4	2	5
a[11]	4	3	8

	[0]	[1]	[2]	[3]	[4]
termini_riga	3	1	3	3	1
pos_iniziale	1	4	5	8	11

19

Trasposizione rapida

	riga	col	valore
a[0]	5	5	11
a[1]	0	0	2
a[2]	0	3	1
a[3]	1	1	1
a[4]	1	2	6
a[5]	2	0	4
a[6]	2	3	1
a[7]	2	4	7
a[8]	3	2	9
a[9]	4	0	1
a[10]	4	2	5
a[11]	4	3	8

	riga	col	valore
b[0]	5	5	11
b[1]	0	0	2
b[2]	0	2	4
b[3]	0	4	1
b[4]	1	1	1
b[5]	2	1	6
b[6]	2	3	9
b[7]	2	4	5
b[8]	3	0	1
b[9]	3	2	1
b[10]	3	4	8
b[11]	4	2	7

	[0]	[1]	[2]	[3]	[4]
termini_riga	3	1	3	3	1
pos_iniziale	1 /2 /3 /4	4 /5	5 /6 /7 /8	8 /9 /10 /11	11 /12

20

Trasposizione rapida

```
void trasp_rapida(termine a[], termine b[])
{
 int termini_riga[MAX_COL], pos_iniziale[MAX_COL];
 int i,j, num_col = a[0].col, num_termini=a[0].valore;
 b[0].riga = num_col;   b[0].col = a[0].riga;
 b[0].valore = num_termini;

 if(num_termini>0) {
       for(i=0; i<num_col; i++)   termini_riga[i]=0;
       for(i=1; i<=num_termini; i++)   termini_riga[a[i].col]++;
       pos_iniziale[0]=1;
       for(i=1; i<num_col; i++)
             pos_iniziale[i] = pos_iniziale[i-1]+termini_riga[i-1];
       for(i=1; i<=num_termini; i++) {
             j=pos_iniziale[a[i].col]++;
             b[j].riga = a[i].col;
             b[j].col = a[i].riga;
             b[j].valore = a[i].valore;
            }
      }
}
```

Analisi di `trasp_rapida`

```
void trasp_rapida(termine a[], termine b[])
{
 int termini_riga[MAX_COL], pos_iniziale[MAX_COL];
 int i,j, num_col = a[0].col, num_termini=a[0].valore;
 b[0].riga = num_col ;   b[0].col = a[0].riga;
 b[0].valore = num_termini;

 if(num_termini>0) {
       for(i=0; i<num_col; i++)
             termini_riga[i]=0;                          } = O(colonne)
       for(i=1; i<=num_termini; i++)
             termini_riga[a[i].col]++;                   } = O(elementi)
       pos_iniziale[0]=1;
       for(i=1; i<num_col; i++)
             pos_iniziale[i] = pos_iniziale[i-1]+
                    termini_riga[i-1];                   } = O(colonne)
       for(i=1; i<=num_termini; i++) {
             j=pos_iniziale[a[i].col]++;
             b[j].riga = a[i].col;
             b[j].col = a[i].riga;
             b[j].valore = a[i].valore;                  } = O(elementi)
            }
       }
}
```

$O(elementi+colonne) = O(righe \times colonne)$