More on Arrays

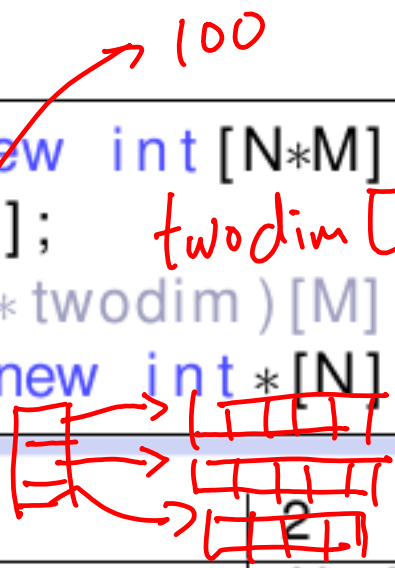
Hsuan-Tien Lin

Dept. of CSIE, NTU

March 7, 2013

Comparison of Three Implementations

```
1 int* twodim = new int[N*M];
2 int twodim[N][M];
3   //also, int (*twodim)[M] = new int[N][M];
4 int** twodim = new int*[N]; // and ...
```

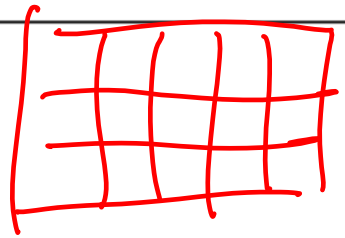


	1	2	3
space	$N * M$ integers	$N * M$ int.	$N * M$ int. + N pointers
type	`int*`	`int*[M]`	`int**`
construct	constant	constant	prop. to N
get	arithmetic+dereference	arith.+deref.	deref.+deref.

method 2 for static allocating (constant M); method 1 or 3 for dynamic allocating (your choice)

A Tale between Two Programs

```
1  int rowsum(){
2    int i, j;
3    int res = 0;
4    for(i=0;i<MAXROW;i++)
5      for(j=0;j<MAXCOL;j++)
6        res += array[i][j];
7  }
```



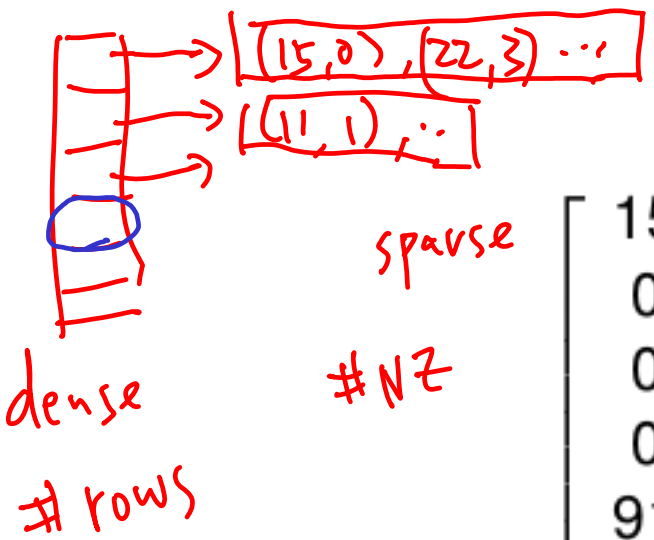
43

4

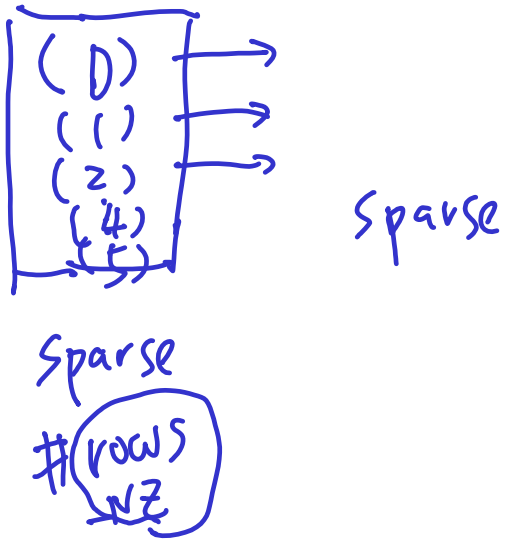
```
1  int colsum(){
2    int i, j;
3    int res = 0;
4    for(j=0;j<MAXCOL;j++)
5      for(i=0;i<MAXROW;i++)
6        res += array[i][j];
7  }
```

2

Sparse Matrix



$$\begin{bmatrix} 15 & 0 & 0 & 22 & 0 & -15 \\ 0 & 11 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 91 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 28 & 0 & 0 & 0 \end{bmatrix}$$



Specialty

a rectangular 2-D array that contains many common elements (0) that we may not want to repeatedly store

Data Structures for Sparse Matrix

- dense implementation: as 2D dense arrays
- array of array implementation:
  - "(dense 1D) of (sparse 1D)"
  - "(sparse 1D) of (sparse 1D)" ✓
- ordered triples implementation: see next page

Ordered Triples Implementation

$$\begin{bmatrix} 15 & 0 & 0 & 22 & 0 & -15 \\ 0 & 11 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 91 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 28 & 0 & 0 & 0 \end{bmatrix}$$

Ordered(-by-row-then-by-col) Triples

6	6	8
row	*col*	*value*
0	0	15
0	3	22
0	5	-15
1	1	11
⋮		

- space? (#NZ * 3 + □)
- getting rapidly?

simple exercise: compare to unordered triple implementation