Sparse Matrices

- sparse … many elements are zero
- dense … few elements are zero

Example Of Sparse Matrices

diagonal
tridiagonal
lower triangular (?)

These are structured sparse matrices.
May be mapped into a 1D array so that a mapping function can be used to locate an element.

Unstructured Sparse Matrices

Airline flight matrix.
- airports are numbered 1 through n
- flight(i,j) = list of nonstop flights from airport i to airport j
- n = 1000 (say)
- n x n array of list references => 4 million bytes
- total number of flights = 20,000 (say)
- need at most 20,000 list references => at most 80,000 bytes

Web page matrix.
- web pages are numbered 1 through n
- web(i,j) = number of links from page i to page j

Web analysis.
- authority page … page that has many links to it
- hub page … links to many authority pages

Web Page Matrix

- n = 2 billion (and growing by 1 million a day)
- n x n array of ints => 16 * 10^18 bytes (16 * 10^9 GB)
- each page links to 10 (say) other pages on average
- on average there are 10 nonzero entries per row
- space needed for nonzero elements is approximately 20 billion x 4 bytes = 80 billion bytes (80 GB)

Representation Of Unstructured Sparse Matrices

Single linear list in row-major order.
- scan the nonzero elements of the sparse matrix in row-major order
- each nonzero element is represented by a triple (row, column, value)
- the list of triples may be an array list or a linked list (chain)
Single Linear List Example

```
0 0 3 0 4
0 0 5 7 0
0 0 0 0 0
0 2 6 0 0
```

List =

```
row      1  1  2  2  4  4
column  3  5  3  4  2  3
value     3  4  5  7  2  6
```

Array Linear List Representation

```
row 1 1 2 2 4 4
column 3 5 3 4 2 3
value 3 4 5 7 2 6
```

Element 0 1 2 3 4 5

```
row 1 1 2 2 4 4
column 3 5 3 4 2 3
value 3 4 5 7 2 6
```

Chain Representation

Node structure.

```
row
col
next
value
```

Single Chain

```
row 1 1 2 2 4 4
column 3 5 3 4 2 3
value 3 4 5 7 2 6
```

Array Of Row Chains

Node structure.

```
next
col
value
```

One Linear List Per Row

```
0 0 3 0 4
0 0 5 7 0
0 0 0 0 0
0 2 6 0 0
```

Row1 = [(3, 3), (5, 4)]
Row2 = [(3, 5), (4, 7)]
Row3 = []
Row4 = [(2, 2), (3, 6)]

Array Of Row Chains

Node structure.
Array Of Row Chains

Orthogonal List Representation
Both row and column lists.
Node structure.

Row Lists

Column Lists

Orthogonal Lists

Variations
May use circular lists instead of chains.
## Approximate Memory Requirements

<table>
<thead>
<tr>
<th>Format</th>
<th>Memory Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>500 x 500 matrix with 1994 nonzero elements</td>
<td>1 million bytes</td>
</tr>
<tr>
<td>2D array</td>
<td>500 x 500 x 4 = 1 million bytes</td>
</tr>
<tr>
<td>Single Array List</td>
<td>3 x 1994 x 4 = 23,928 bytes</td>
</tr>
<tr>
<td>One Chain Per Row</td>
<td>23928 + 500 x 4 = 25,928 bytes</td>
</tr>
</tbody>
</table>

## Runtime Performance

<table>
<thead>
<tr>
<th>Format</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix Transpose</td>
<td>210 ms</td>
</tr>
<tr>
<td>500 x 500 matrix with 1994 nonzero elements</td>
<td></td>
</tr>
<tr>
<td>2D array</td>
<td></td>
</tr>
<tr>
<td>Single Array List</td>
<td>6 ms</td>
</tr>
<tr>
<td>One Chain Per Row</td>
<td>12 ms</td>
</tr>
</tbody>
</table>

## Performance

<table>
<thead>
<tr>
<th>Format</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix Addition</td>
<td>880 ms</td>
</tr>
<tr>
<td>500 x 500 matrices with 1994 and 999 nonzero elements</td>
<td></td>
</tr>
<tr>
<td>2D array</td>
<td></td>
</tr>
<tr>
<td>Single Array List</td>
<td>18 ms</td>
</tr>
<tr>
<td>One Chain Per Row</td>
<td>29 ms</td>
</tr>
</tbody>
</table>