Housekeeping

Lookup the cluster script pinned to #general. Run asap

No class on 3/16. In it's place we will extend the class on 3/14 from 4 – 6:30

Final Project:

1. Proposal Presentations due 3/14
2. Proposal writeup due 3/21

Final Project

- Introductions (Name, members and project/idea) - TODAY
- Present your proposal — 3/14
  - Each team member will present one aspect of the project
  - Describe the project, related work, deliverables, schedule, and intellectual challenges (or what do you expect to gain from it).
  - Will be Q&A style with feedback
  - 15-20mins
- Submit your proposal — 3/21
  - 3-4 pages, single spaced
  - Response to any questions/concerns raised during the presentation
  - Details that could not be covered in the proposal, as well as schedule and deliverables.

Final Project (cont…)

- Project Update – 4/6 (tentative)
  - 5-10 min presentation.
  - Are you on track?
- Final Presentation – 4/25 & 4/27 (tentative)
- Final Report – 5/2

Some final words on Parallelization

Sources for this lecture:

- Jim Demmel/Kathy Yelick (Berkeley - CS 267)
- Mary Hall (Utah – CS4961)
- Software Systems for Numerical Analysis (Emory CS561)

http://www.mathcs.emory.edu/~cheung/Courses/561/

- Our focus has been Dense arrays and loop-based data-parallel computation has been the focus of this class so far
- Real world is rarely that dense and organized

SPARSE MATRICES

Dense Linear Algebra vs. Sparse Linear Algebra

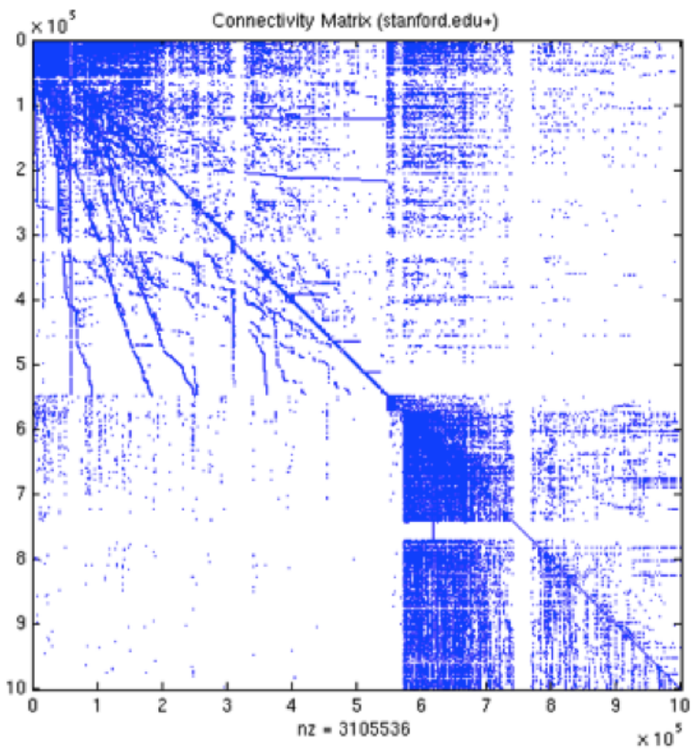
- Matrix vector multiply:

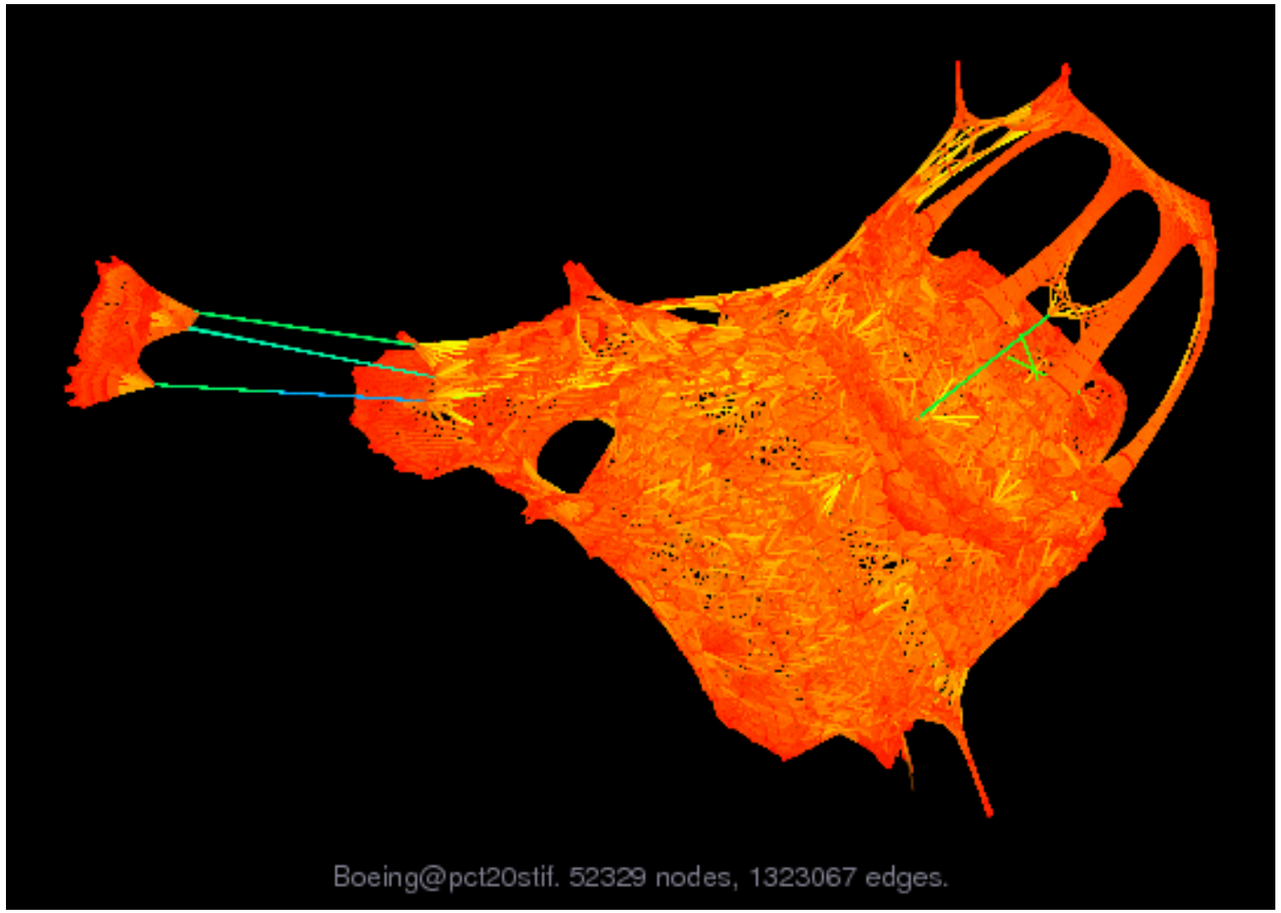
```
for (i=0; i<n; i++)
    for (j=0; j<n; j++)
        a[i] += c[j][i]*b[j];
```

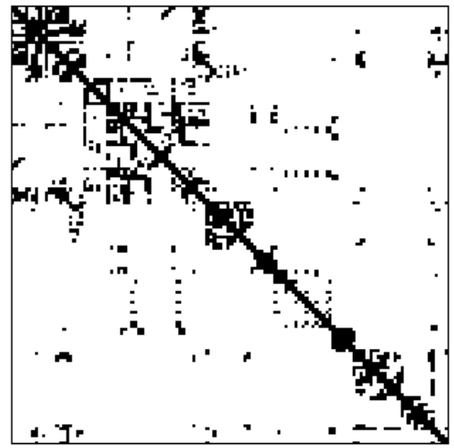
- What if n is very large, and some large percentage (say 90%) of c is zeros?
- Should you represent all those zeros?
- If not, how to represent “c”?

Not a hypothetical situation

- Most matrices arising from real applications are sparse.
- A 1M-by-1M submatrix of the web connectivity graph, constructed from an archive at the Stanford WebBase.







Matrix properties

number of rows	52,329
number of columns	52,329
nonzeros	2,698,463

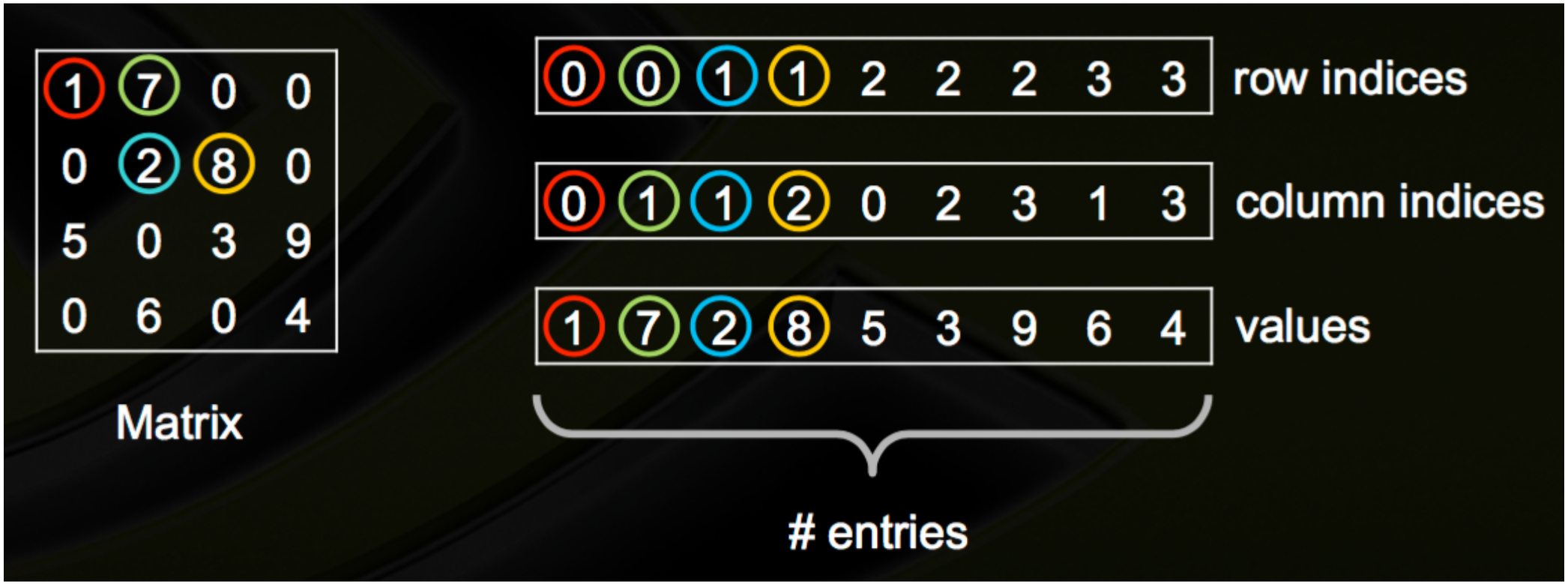
Examples from Tim Davis's Sparse Matrix Collection,
http://www.cise.ufl.edu/research/sparse/matrices/

Boeing CT20 ENGINE BLOCK -- STIFFNESS MATRIX

Sparse Linear Algebra

- Suppose you are applying matrix-vector multiply and the matrix has lots of zero elements
  - Computation cost? Space requirements?
- General sparse matrix representation concepts
  - Primarily only represent the nonzero data values
  - Auxiliary data structures describe placement of nonzeros in “dense matrix"

1. COO → Coordinate-wise Storage Method
2. CSR/CCR → Compressed Sparse Row (or Column) Method



Sparse Matrix Representations and Iterative Solvers – Nathan Bell, nVidia

COO → Coordinate-wise Storage Method

	0	1	2	3	4	5	6	7	
0	[11	12		14					]
1	[	22	23		25				]
2	[31		33	34					]
3	[	42			45	46			]
4	[				55				]
5	[				65	66	67		]
6	[				75		77	78	]
7	[						87	88	]

Representation using Coordinate-wise method:

Index	0	1	2	3	4	5	6	7	8	9	10
Val	11	12	14	22	23	25	31	33	34	42	45
Row	0	0	0	1	1	1	2	2	2	3	3
Col	0	1	3	1	2	4	0	2	3	1	4

Index	11	12	13	14	15	16	17	18	19	20	21
Val	46	55	65	66	67	75	77	78	87	88	-
Row	3	4	5	5	5	6	6	6	7	7	-
Col	5	4	4	5	6	4	6	7	6	7	-

Detailed Example

Val[N]: value of the non-zero elements

Row[N]: row-index of non-zero elements

Col[N]: column -index of non-zero elements

Slides from: Emory – CS561

Detailed Example

```
      0   1   2   3   4   5   6   7
0 [ 11  12      14                  ]
1 [     22  23      25              ]
2 [ 31      33  34                  ]
3 [     42          45  46          ]
4 [                 55              ]
5 [                 65  66  67      ]
6 [                 75      77  78  ]
7 [                         87  88  ]
```

Representation using Coordinate-wise method:

Index	0	1	2	3	4	5	6	7	8	9	10
Val	11	12	14	22	23	25	31	33	34	42	45
Row	0	0	0	1	1	1	2	2	2	3	3
Col	0	1	3	1	2	4	0	2	3	1	4

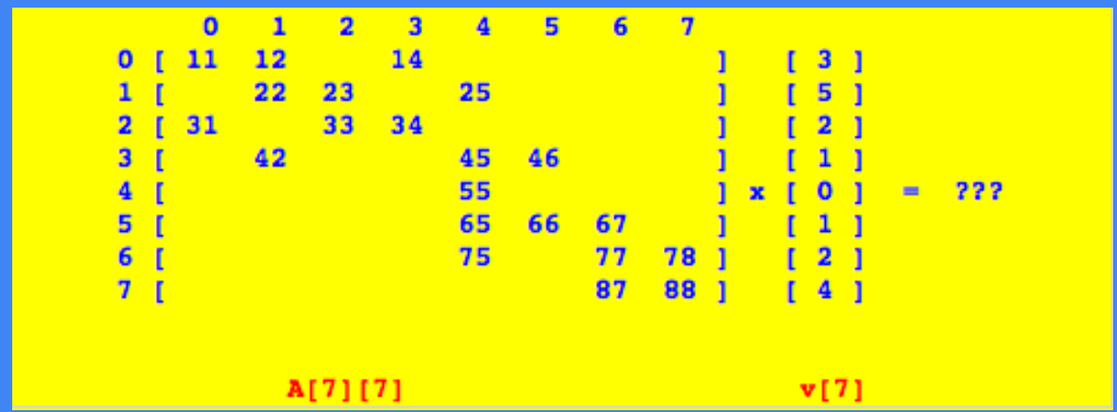
Index	11	12	13	14	15	16	17	18	19	20	21
Val	46	55	65	66	67	75	77	78	87	88	-
Row	3	4	5	5	5	6	6	6	7	7	-
Col	5	4	4	5	6	4	6	7	6	7	-

Slides from: Emory – CS561

Matrix Multiplication

```
for (k = 0; k < N; k = k + 1)
  result[i] = 0;

for (k = 0; k < nnz; k = k + 1)
  result[Row[k]] += Val[k]*d[Col[k]];
```



Slides from: Emory – CS561 (http://goo.gl/HUYKzP)

Suppose d[0..N-1] = (1, 0, 0, 0, 0, 0, 0, 0)

```
[ 11  12      14                    ]   [ 1 ]   [ 11 ]
[     22  23      25                ]   [ 0 ]   [  0 ]
[ 31      33  34                    ]   [ 0 ]   [ 31 ]
[     42          45  46            ] x [ 0 ] = [  0 ]
[                 55                ]   [ 0 ]   [  0 ]
[                 65  66  67        ]   [ 0 ]   [  0 ]
[                 75      77  78    ]   [ 0 ]   [  0 ]
[                         87  88    ]   [ 0 ]   [  0 ]
```

After:

```
for (k = 0; k < N; k = k + 1)
    result[i] = 0;
                        ---> [0, 0, 0, 0, 0, 0, 0, 0]

k = 0;
    result[Row[0]] = result[Row[0]] + Val[0]*d[Col[0]];
    result[0] = result[0] + Val[0]*d[0];
    result[0] = 0 + 11*1;
                        ---> [11, 0, 0, 0, 0, 0, 0, 0]
```

```
k = 1;
    result[Row[1]] = result[Row[1]] + Val[1]*d[Col[1]];
    result[0] = result[0] + Val[1]*d[1];
    result[0] = 0 + 12*0;
                        ---> [11, 0, 0, 0, 0, 0, 0, 0]

k = 2;
    result[Row[2]] = result[Row[2]] + Val[2]*d[Col[2]];
    result[0] = result[0] + Val[2]*d[3];
    result[0] = 0 + 14*0;
                        ---> [11, 0, 0, 0, 0, 0, 0, 0]

...
k = 6;
    result[Row[6]] = result[Row[6]] + Val[6]*d[Col[6]];
    result[2] = result[2] + Val[6]*d[0];
    result[2] = 0 + 31*1;
                        ---> [11, 0, 31, 0, 0, 0, 0, 0]
```


Sample Execution

Slides from: Emory – CS561 (http://goo.gl/HUYKzP)

```
[ 11  12        14                      ]
[     22  23        25                  ]
[ 31        33  34                      ]
[     42              45  46            ]
[                       55              ]
[                       65  66  67      ]
[                       75      77  78  ]
[                               87  88  ]
```

Representation using Compressed Sparse Row method:

Index	0	1	2	3	4	5	6	7	8	9	10
Val	11	12	14	22	23	25	31	33	34	42	45
RowPtr	0	3	6	9	12	13	16	19	21	-	-
Col	0	1	3	1	2	4	0	2	3	1	4

Index	11	12	13	14	15	16	17	18	19	20	21
Val	46	55	65	66	67	75	77	78	87	88	-
RowPtr	-	-	-	-	-	-	-	-	-	-	-
Col	5	4	4	5	6	4	6	7	6	7	-

Compressed Sparse Row

Val[N]: contains the value of the non-zero elements

Col[N]: contains the column-index of the non-zero elements

RowPtr[N]: contains the row-index range of the non-zero elements

Slides from: Emory – CS561 (http://goo.gl/HUYKzP)

```
[ 11  12        14                  ]
[     22  23        25              ]
[ 31        33  34                  ]
[     42              45  46        ]
[                     55            ]
[                     65  66  67    ]
[                     75      77  78]
[                                 87  88]
```

Representation using Compressed Sparse Row method:

Index	0	1	2	3	4	5	6	7	8	9	10
Val	11	12	14	22	23	25	31	33	34	42	45
RowPtr	0	3	6	9	12	13	16	19	21	-	-
Col	0	1	3	1	2	4	0	2	3	1	4

Index	11	12	13	14	15	16	17	18	19	20	21
Val	46	55	65	66	67	75	77	78	87	88	-
RowPtr	-	-	-	-	-	-	-	-	-	-	-
Col	5	4	4	5	6	4	6	7	6	7	-

Compressed Sparse Row

Val[N]: contains the value of the non-zero elements

Col[N]: contains the column-index of the non-zero elements

RowPtr[N]: contains the row-index range of the non-zero elements

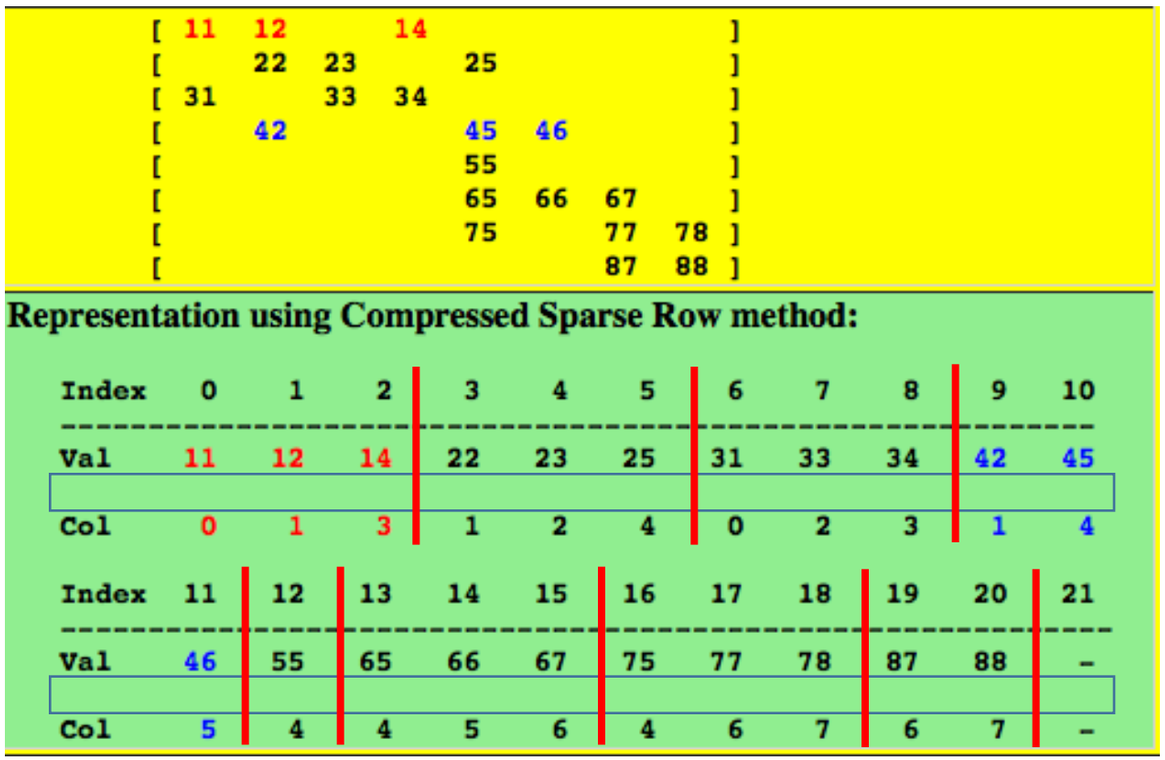
Slides from: Emory – CS561 (http://goo.gl/HUYKzP)

Compressed Sparse Row

Val[N]: contains the value of the non-zero elements

Col[N]: contains the column-index of the non-zero elements

RowPtr[N]: contains the row-index range of the non-zero elements



```
[ 11  12      14                    ]
[     22  23      25                ]
[ 31      33  34                    ]
[     42              45  46        ]
[                     55            ]
[                     65  66  67    ]
[                     75      77  78]
[                             87  88]
```

Representation using Compressed Sparse Row method:

Index	0	1	2	3	4	5	6	7	8	9	10
Val	11	12	14	22	23	25	31	33	34	42	45
Col	0	1	3	1	2	4	0	2	3	1	4

Index	11	12	13	14	15	16	17	18	19	20	21
Val	46	55	65	66	67	75	77	78	87	88	-
Col	5	4	4	5	6	4	6	7	6	7	-

0 1 3 | 1 2 4 | 0 2 3 | 1 4 5 | 4 | 4 5 6 | 4 6 7 | 6 7 |

0 | 3 | 6 | 9 | 12 | 13 | 16 | 19 | 21

Slides from: Emory – CS561 (http://goo.gl/HUYKzP)

Matrix Multiplication

```
for (k = 0; k < N; k = k + 1)
     result[i] = 0;

  for (i = 0; i < N; i = i + 1)
  {
    for (k = RowPtr[i]; k<RowPtr[i+1];k++)
    {
      result[i] += Val[k]*d[Col[k]];
    }
  }
```

Suppose d[0..N-1] = (1, 0, 0, 0, 0, 0, 0, 0)

```
[ 11  12      14                    ]   [ 1 ]   [ 11 ]
[     22  23      25                ]   [ 0 ]   [  0 ]
[ 31      33  34                    ]   [ 0 ]   [ 31 ]
[     42          45  46            ] x [ 0 ] = [  0 ]
[                 55                ]   [ 0 ]   [  0 ]
[                 65  66  67        ]   [ 0 ]   [  0 ]
[                 75      77  78    ]   [ 0 ]   [  0 ]
[                             87  88 ]   [ 0 ]   [  0 ]
```


Slides from: Emory – CS561 (http://goo.gl/HUYKzP)

```
After:

   for (k = 0; k < N; k = k + 1)
       result[i] = 0;
                            ---> [0, 0, 0, 0, 0, 0, 0, 0]

   i = 0;
   k = 0;
       result[0] = result[0] + Val[0]*d[Col[0]];
       result[0] = result[0] + Val[0]*d[0];
       result[0] = 0 + 11*1;
                            ---> [11, 0, 0, 0, 0, 0, 0, 0]

   i = 0;
   k = 1;
       result[0] = result[0] + Val[1]*d[Col[1]];
       result[0] = result[0] + Val[1]*d[1];
       result[0] = 0 + 12*0;
                            ---> [11, 0, 0, 0, 0, 0, 0, 0]
```

```
   i = 0;
   k = 2;
       result[0] = result[0] + Val[2]*d[Col[2]];
       result[0] = result[0] + Val[2]*d[3];
       result[0] = 0 + 14*0;
                            ---> [11, 0, 0, 0, 0, 0, 0, 0]

   ...
   i = 2;
   k = RowPtr[2] = 6;
       result[2] = result[2] + Val[6]*d[Col[6]];
       result[2] = result[2] + Val[6]*d[0];
       result[2] = 0 + 31*1;
                            ---> [11, 0, 31, 0, 0, 0, 0, 0]
```


Sample Execution

Slides from: Emory – CS561 (http://goo.gl/HUYKzP)

HW 3

Individual or Team of 2
Due 3/20

Large scale problems so develop a testing and evaluation strategy.

1. Implement the quicksort on the cluster. Implement for a nD hypercube
2. Implement Google's PageRank algorithm using MPI.
    1. Use data from the Stanford Network Repository
    2. Assume a damping of 0.85
    3. Due 3/20
    4. Submit a brief report w/ your code.