

# Lecture 12 – Object Oriented Design Example: SpMV

Prof. Brendan Kochunas

NERS/ENGR 570 - Methods and Practice of Scientific Computing (F20)



COLLEGE OF ENGINEERING  
NUCLEAR ENGINEERING & RADIOLOGICAL SCIENCES  
UNIVERSITY OF MICHIGAN

# Outline

- Overview of HW 2 (to be assigned after lecture)
- Review of Lab 06
- Class Hierarchy Development
- Matrix State Machine
- Mediator Design Pattern

# Sparse Matrix Storage Formats: COOrdinate Storage

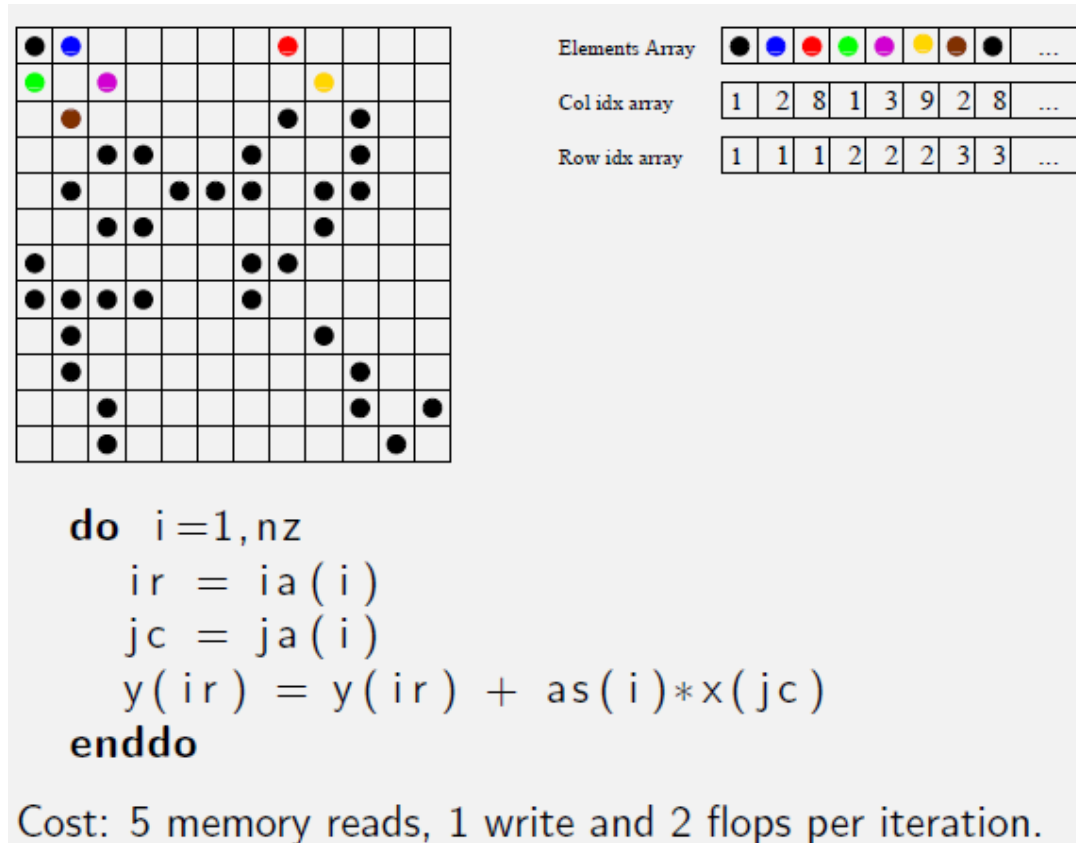


Diagram illustrating COOrdinate Storage format for a sparse matrix. The matrix is represented by a grid where non-zero elements are marked by colored dots. The corresponding arrays are shown to the right:

- Elements Array: [Black, Blue, Red, Green, Purple, Yellow, Brown, Black, ...]
- Col idx array: [1, 2, 8, 1, 3, 9, 2, 8, ...]
- Row idx array: [1, 1, 1, 2, 2, 2, 3, 3, ...]

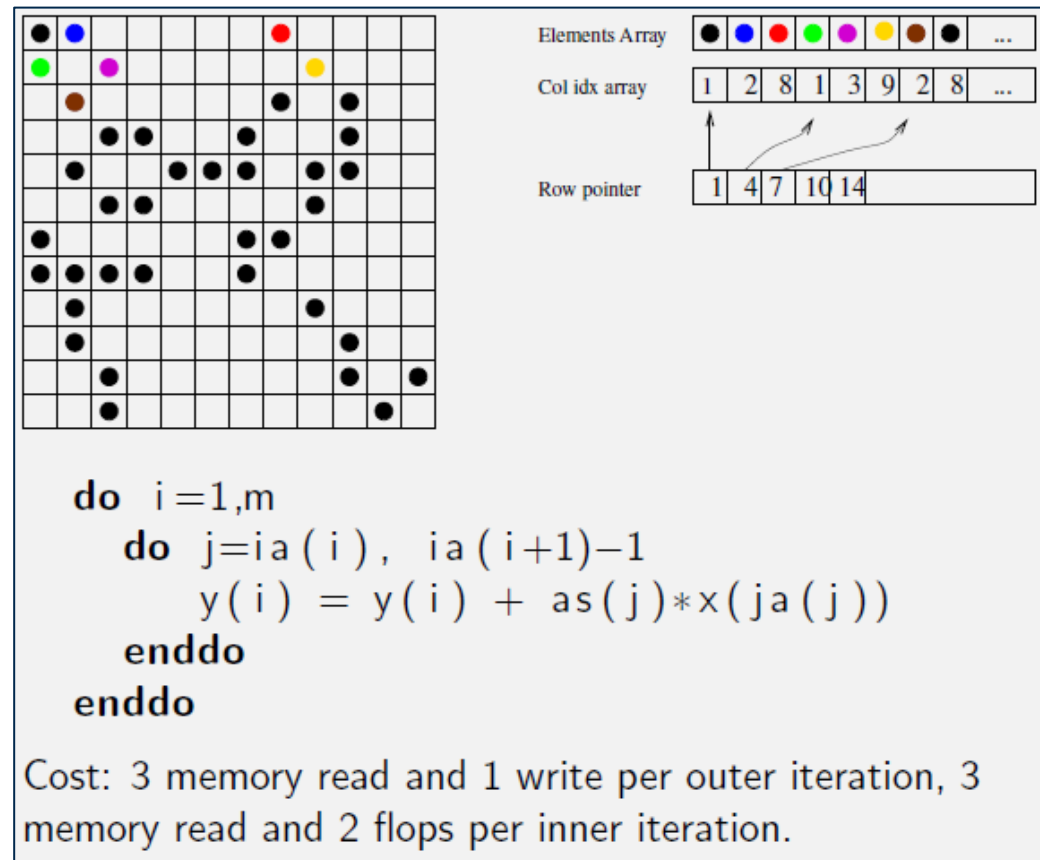
```

do i=1,nz
  ir = ia(i)
  jc = ja(i)
  y(ir) = y(ir) + as(i)*x(jc)
enddo
  
```

Cost: 5 memory reads, 1 write and 2 flops per iteration.

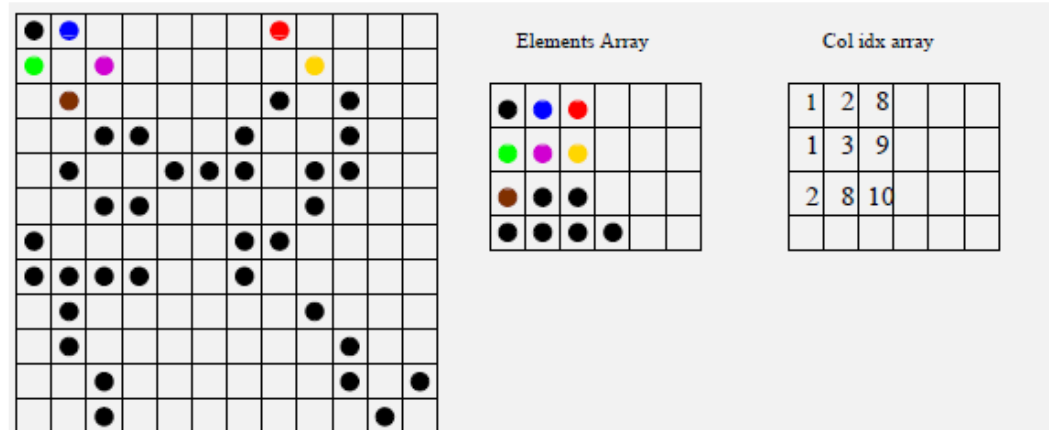
# Sparse Matrix Storage Formats:

## Compressed Sparse Row (CSR) Storage



# Sparse Matrix Storage Formats:

## ELLPACK Storage



The diagram illustrates the ELLPACK storage format. It shows a 10x10 sparse matrix with non-zero elements (black dots) and a specific pattern of colored dots (blue, green, brown, red, yellow, magenta) representing a subset of elements. To the right, the 'Elements Array' is shown as a 4x6 grid with colored dots, and the 'Col idx array' is shown as a 4x6 grid with numerical values.

Elements Array

●	●	●			
●	●	●			
●	●	●			
●	●	●			

Col idx array

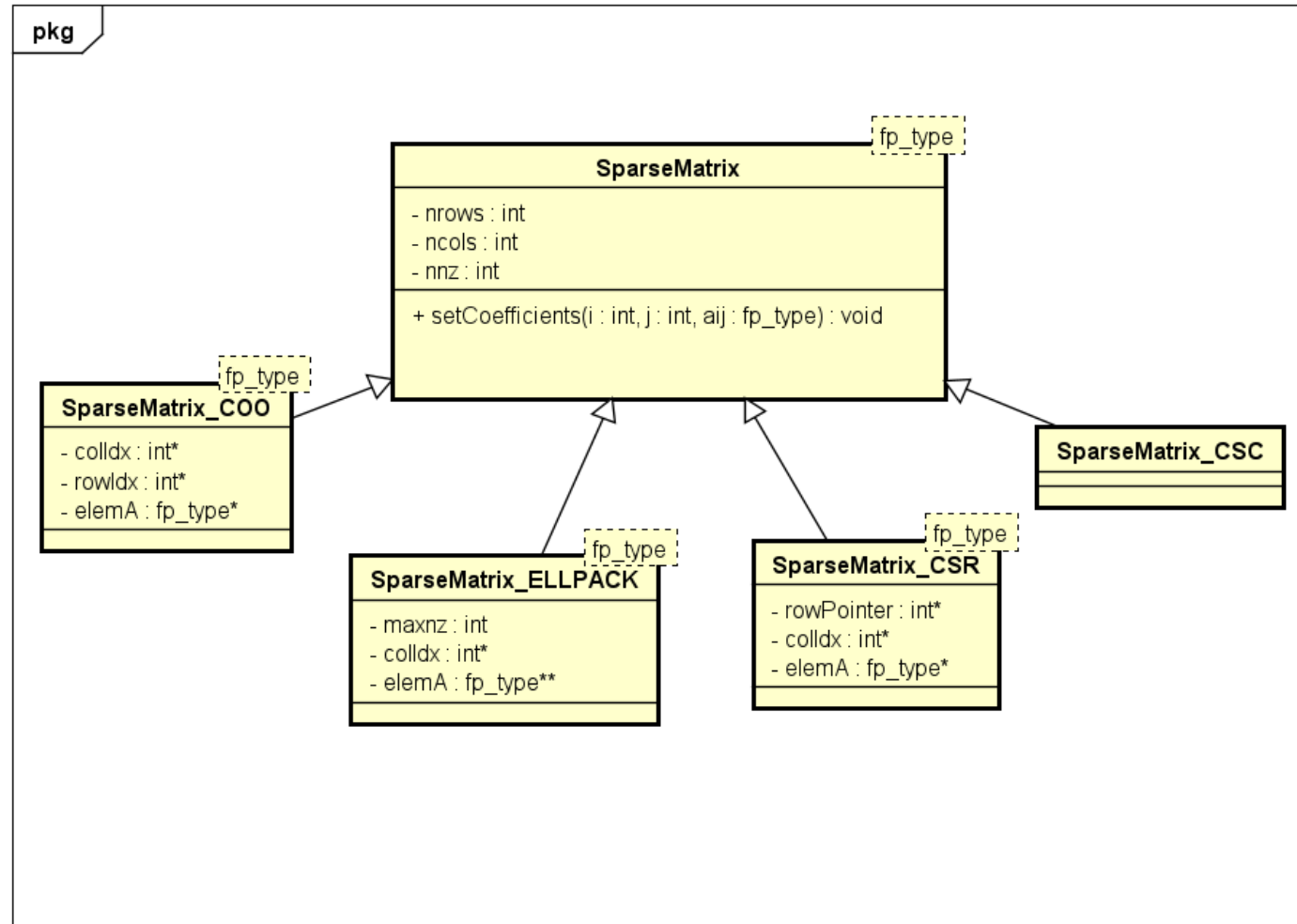
1	2	8			
1	3	9			
2	8	10			

```

do i=1,m
  do j=1, maxnz
    y(i) = y(i) + as(i,j)*x(ja(i,j))
  enddo
enddo
  
```

Cost: 1 memory read and 1 write per outer iteration, 3 memory read and 2 flops per inner iteration (also, regular access pattern).

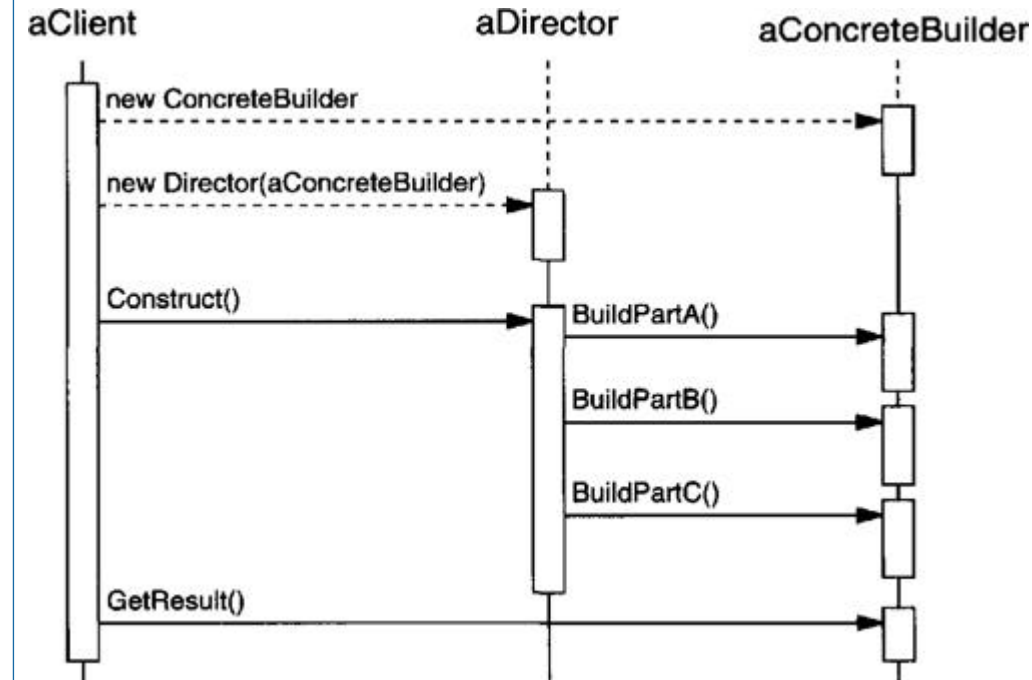
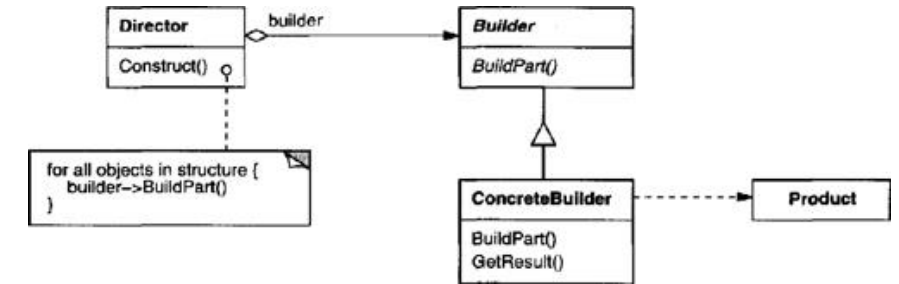
# Class Hierarchy



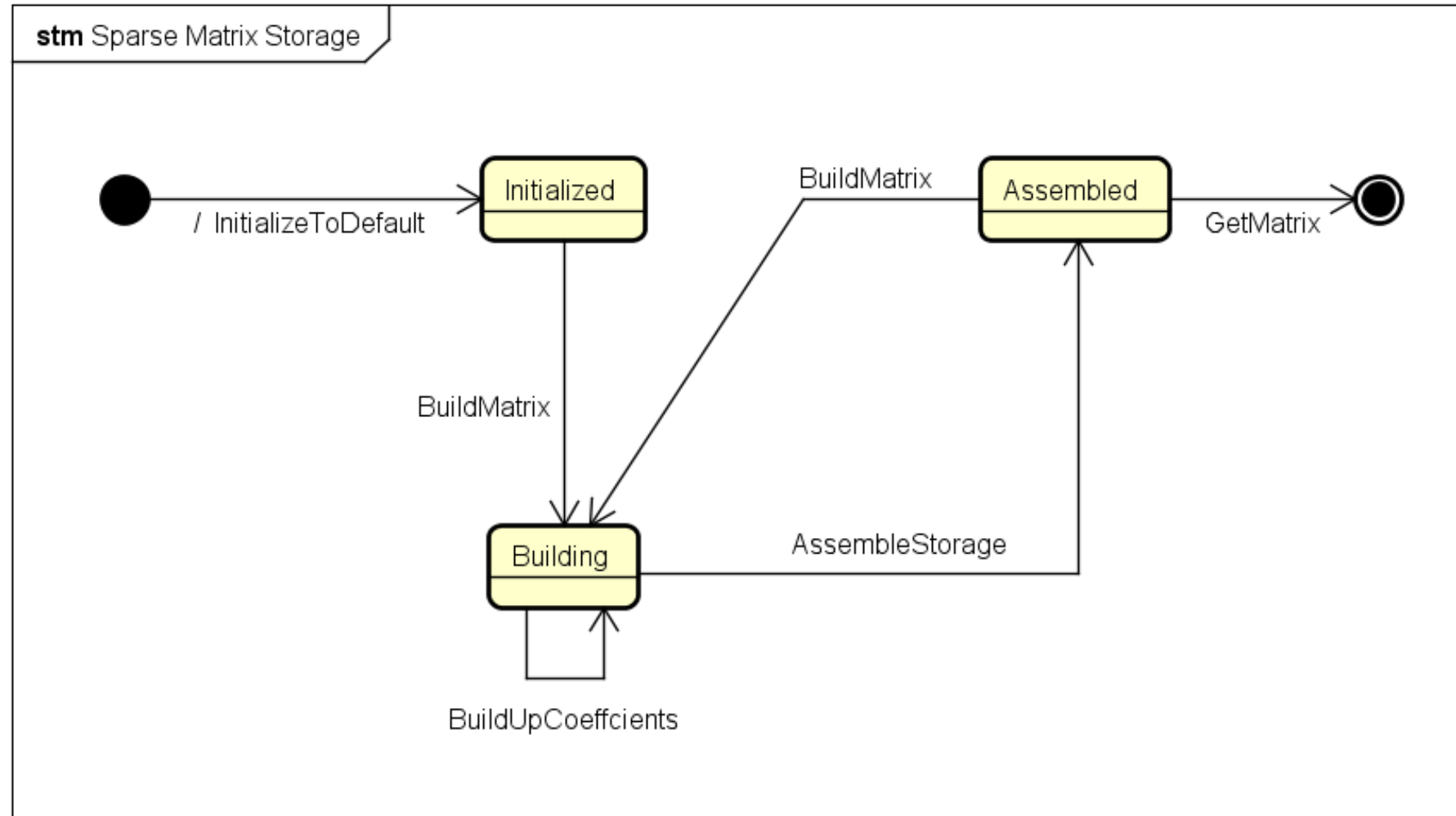
# Creational Pattern (Builder)

- `InitializeToDefault(n)`
  - Create some internal storage and default values
  - e.g. could initialize to identity
- `BuildMatrix()` – Changes state
  - allows values to be set
- `BuildUpCoefficients(i, j, aij)`
  - Assign coefficients to matrix
  - Perhaps overload to allow other formats
    - e.g. COO - `ia(:), ja(:), aa(:)`
  - Store all internally as COO format
- `AssembleStorage()` – Changes state
  - Converts internal representation of data to format suitable for solvers
- `GetMatrix()` – would return the matrix object

## Builder Design Pattern



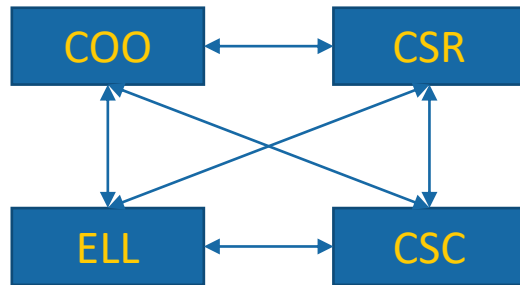
# Behavioral Pattern (State)



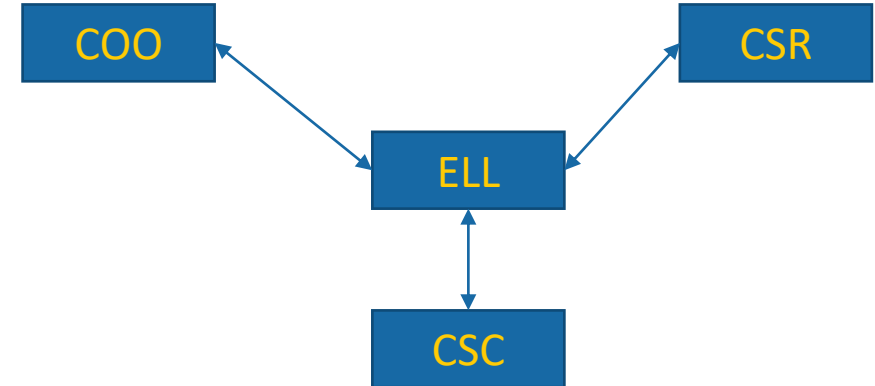


# Behavioral Pattern (Mediator)

- Support  $N$  matrix formats
  - That is  $N^2$  different types of conversions
  - Don't implement them all!



- Use Mediator!
  - Move from fully connected graph to “star” graph



# Notes on Mediator

