Special thanks to:

# Direct Methods for Sparse Linear Systems:

# MATLAB sparse backslash

Tim Davis

davis@cise.ufl.edu

University of Florida

# Sparse matrices arise in ...

computational fluid dynamics, finite-element methods, statistics, time/frequency domain circuit simulation, dynamic and static modeling of chemical processes, cryptography, magneto-hydrodynamics, electrical power systems, differential equations, quantum mechanics, structural mechanics (buildings, ships, aircraft, human body parts...), heat transfer, MRI reconstructions, vibroacoustics, linear and non-linear optimization, financial portfolios, semiconductor process simulation, economic modeling, oil reservoir modeling, astrophysics, crack propagation, Google page rank, 3D computer vision, cell phone tower placement, tomography, multibody simulation, model reduction, nano-technology, acoustic radiation, density functional theory, quadratic assignment, elastic properties of crystals, natural language processing, DNA electrophoresis, ...

For problems this important,

I can't resist to ask:

Can you solve $Ax=b$ faster?

# Sparse data structures

- compressed sparse column format
- Thus, `A(:,j)` is easy in MATLAB; `A(i,:)` hard

$$A = \begin{bmatrix} 4.5 & 0 & 3.2 & 0 \\ 3.1 & 2.9 & 0 & 0.9 \\ 0 & 1.7 & 3.0 & 0 \\ 3.5 & 0.4 & 0 & 1.0 \end{bmatrix}$$

```
Ap: [0,                 3,                 6,           8,          10]
Ai: [0,   1,   3,   1,   2,   3,   0,   2,   1,   3  ]
Ax: [4.5,3.1,3.5,2.9,1.7,0.4,3.2,3.0,0.9,1.0]
```

# Sparse lower triangular solve, x=L\b

```
x = b
for j = 1:n
    if (x(j) ≠ 0)
        x(j+1:n) = x(j+1:n) - L(j+1:n,j) * x(j)
    end
end
```

# Sparse lower triangular solve, x=L\b

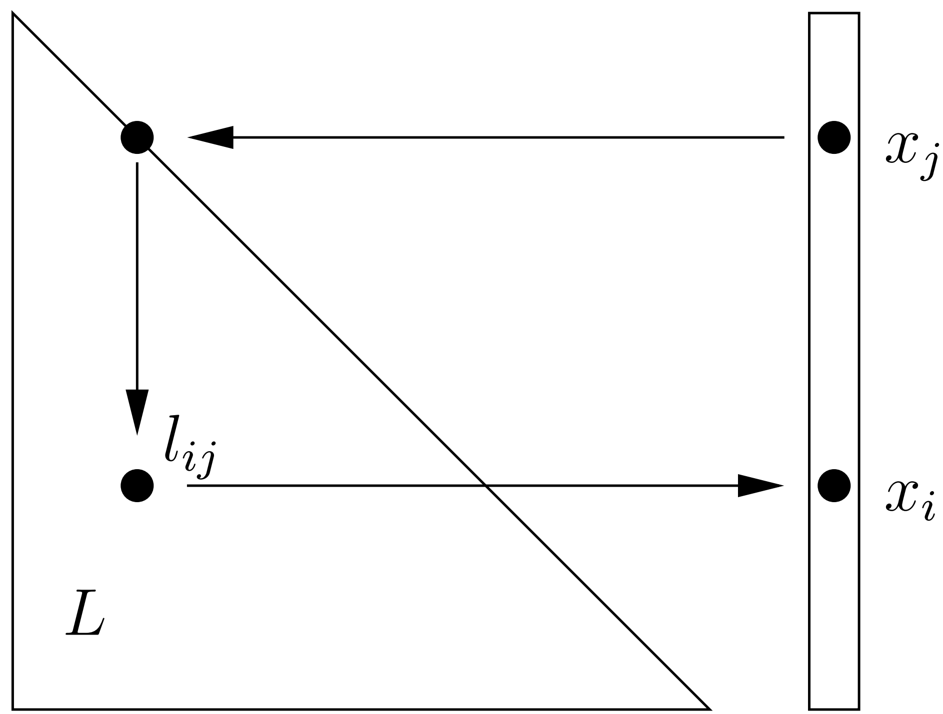
```
x = b
for j = 1:n
    if (x(j) ≠ 0)
        x(j+1:n) = x(j+1:n) - L(j+1:n,j) * x(j)
    end
end
```

- O(n+flops) time too high
- the problem:
  ```
  for j=1:n
    if (x(j) ≠ 0)
  ```
- need pattern of x before computing it

# Sparse lower triangular solve, `x=L\b`

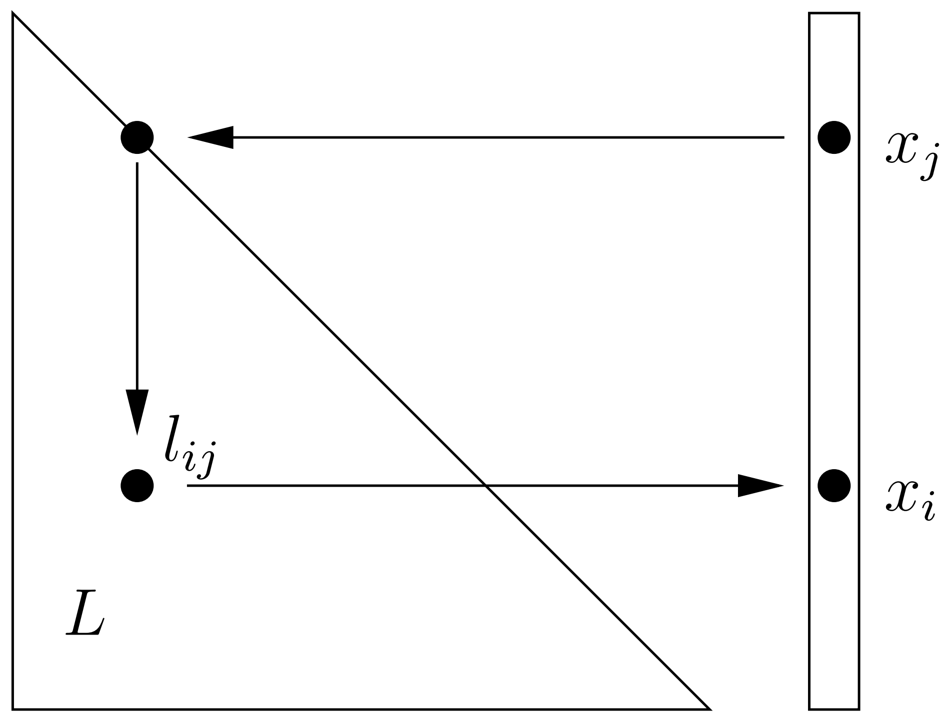
```
x = b
for j = 1:n
    if (x(j) ≠ 0)
        x(j+1:n) = x(j+1:n) - L(j+1:n,j) * x(j)
    end
end
```

- $b_i \neq 0 \Rightarrow x_i \neq 0$



# Sparse lower triangular solve, `x=L\b`

```
x = b
for j = 1:n
    if (x(j) ≠ 0)
        x(j+1:n) = x(j+1:n) - L(j+1:n,j) * x(j)
    end
end
```

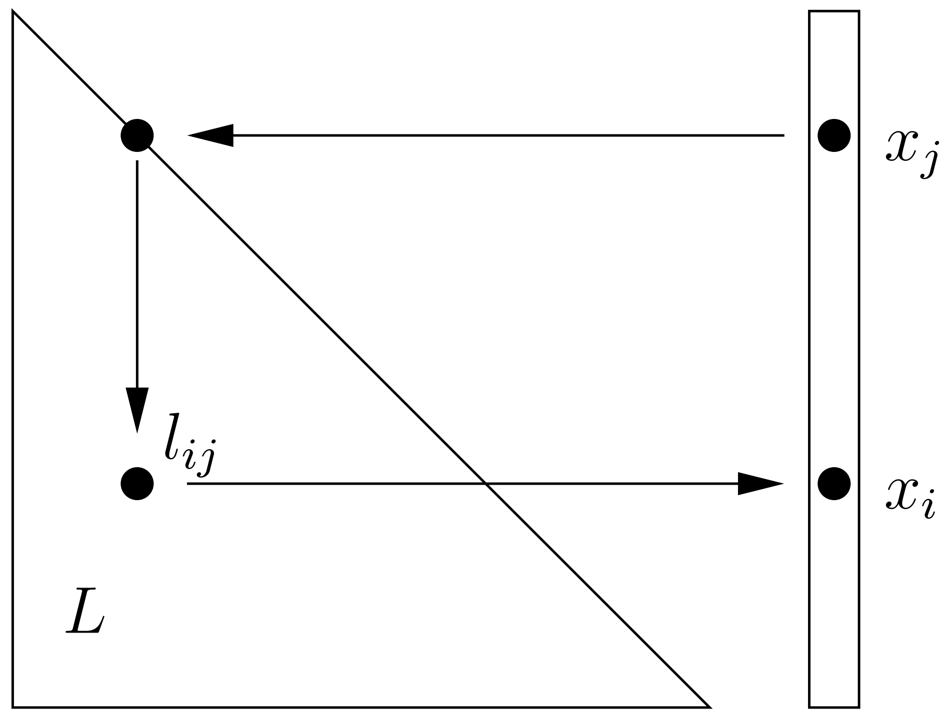



- $b_i \neq 0 \Rightarrow x_i \neq 0$
- $x_j \neq 0 \wedge l_{ij} \neq 0 \Rightarrow x_i \neq 0$

# Sparse lower triangular solve, x=L\b

```
x = b
for j = 1:n
    if (x(j) ≠ 0)
        x(j+1:n) = x(j+1:n) - L(j+1:n,j) * x(j)
    end
end
```

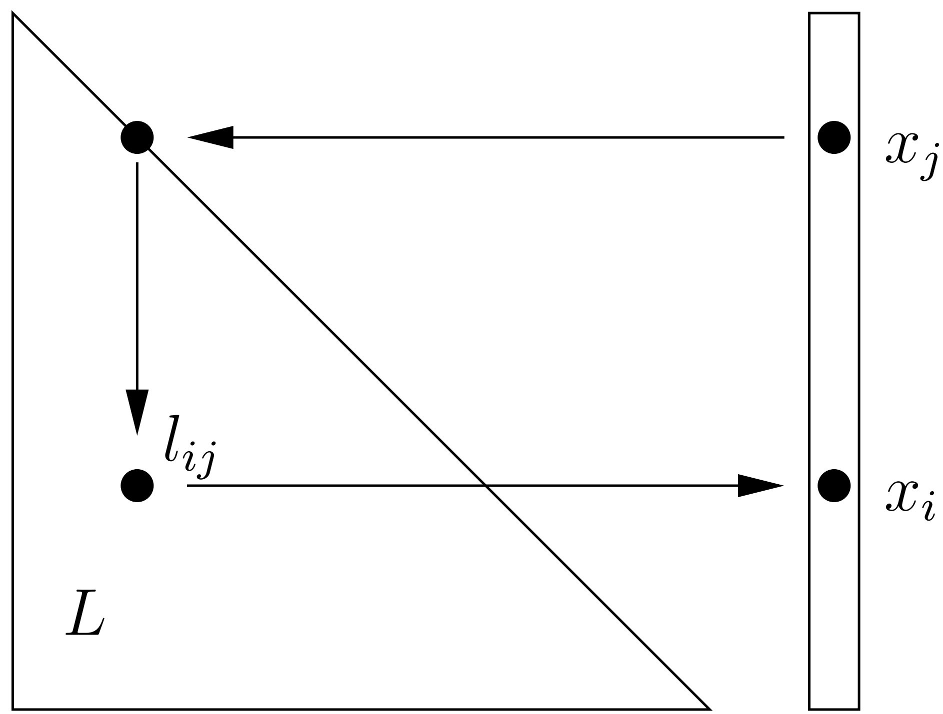
- $b_i \neq 0 \Rightarrow x_i \neq 0$
- $x_j \neq 0 \wedge l_{ij} \neq 0 \Rightarrow x_i \neq 0$
- let $G(L)$ have an edge $j \rightarrow i$ if $l_{ij} \neq 0$



# Sparse lower triangular solve, `x=L\b`

```
x = b
for j = 1:n
    if (x(j) ≠ 0)
        x(j+1:n) = x(j+1:n) - L(j+1:n,j) * x(j)
    end
end
```

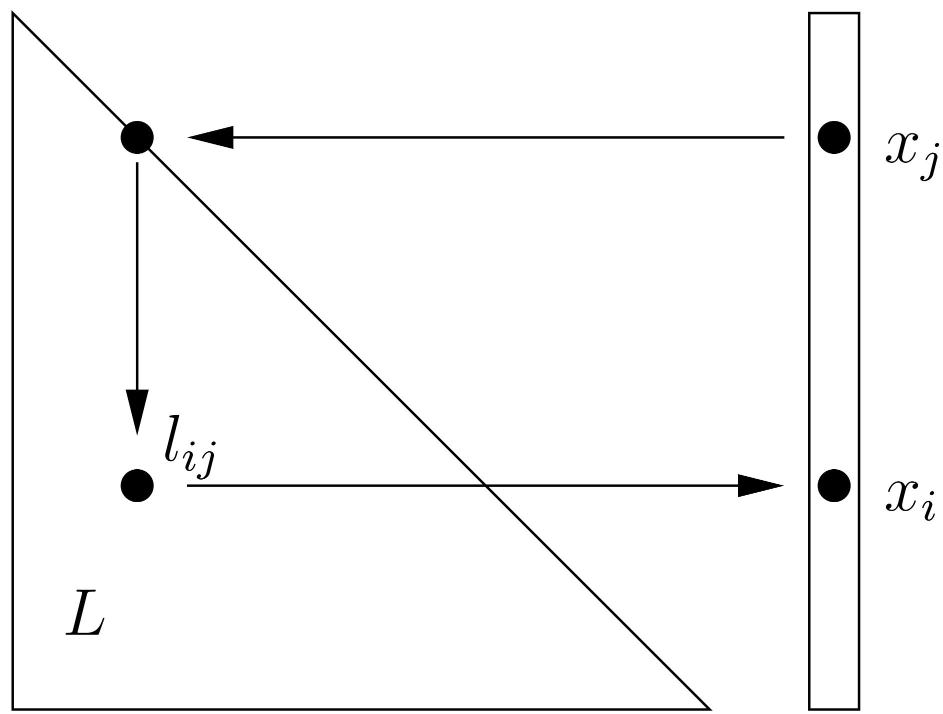
- $b_i \neq 0 \Rightarrow x_i \neq 0$
- $x_j \neq 0 \wedge l_{ij} \neq 0 \Rightarrow x_i \neq 0$
- let $G(L)$ have an edge $j \rightarrow i$ if $l_{ij} \neq 0$
- let $\mathcal{B} = \{i \,|\, b_i \neq 0\}$ and $\mathcal{X} = \{i \,|\, x_i \neq 0\}$



# Sparse lower triangular solve, x=L\b

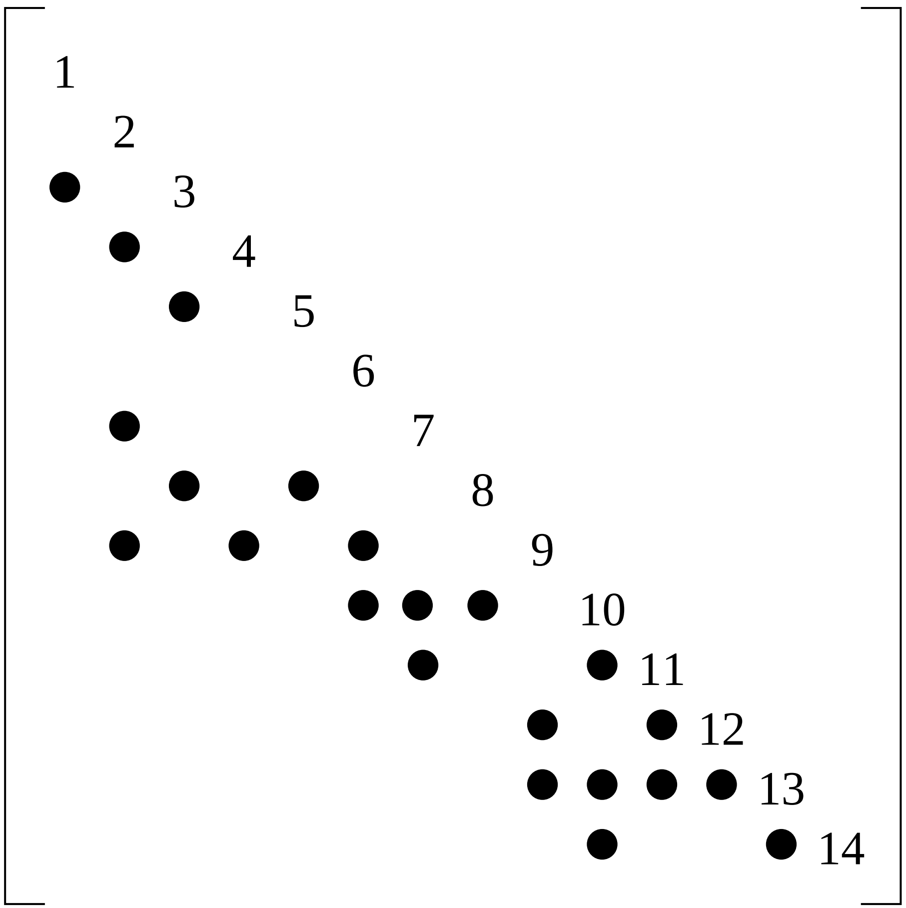
```
x = b
for j = 1:n
    if (x(j) ≠ 0)
        x(j+1:n) = x(j+1:n) - L(j+1:n,j) * x(j)
    end
end
```

- $b_i \neq 0 \Rightarrow x_i \neq 0$
- $x_j \neq 0 \wedge l_{ij} \neq 0 \Rightarrow x_i \neq 0$
- let $G(L)$ have an edge $j \rightarrow i$ if $l_{ij} \neq 0$
- let $\mathcal{B} = \{i \,|\, b_i \neq 0\}$ and $\mathcal{X} = \{i \,|\, x_i \neq 0\}$
- then $\mathcal{X} = \text{Reach}_{G(L)}(\mathcal{B})$

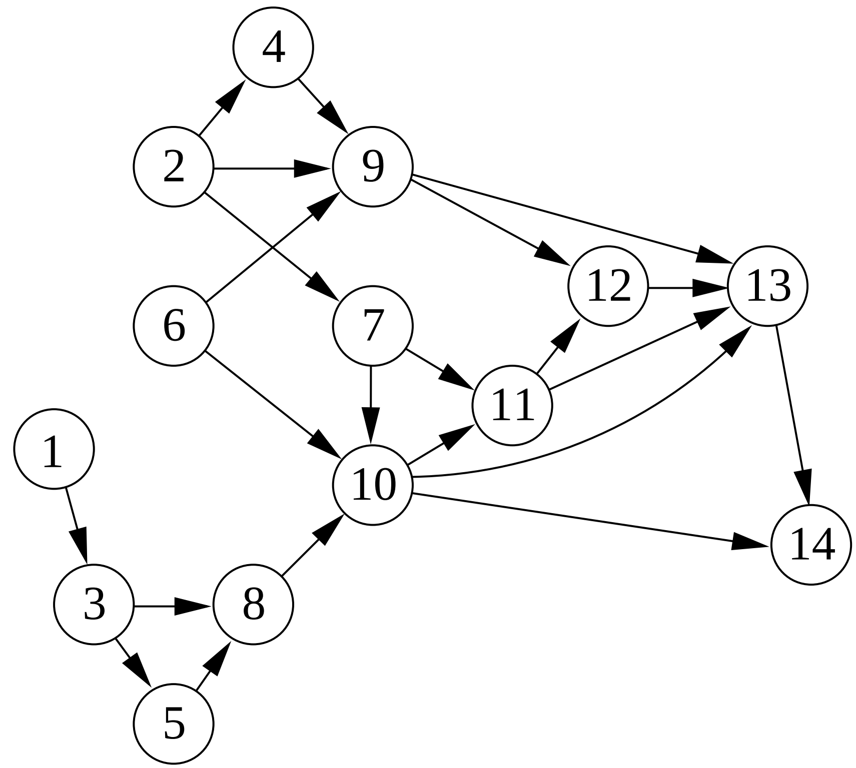


# Sparse lower triangular solve, x=L\b

Lower triangular matrix $L$

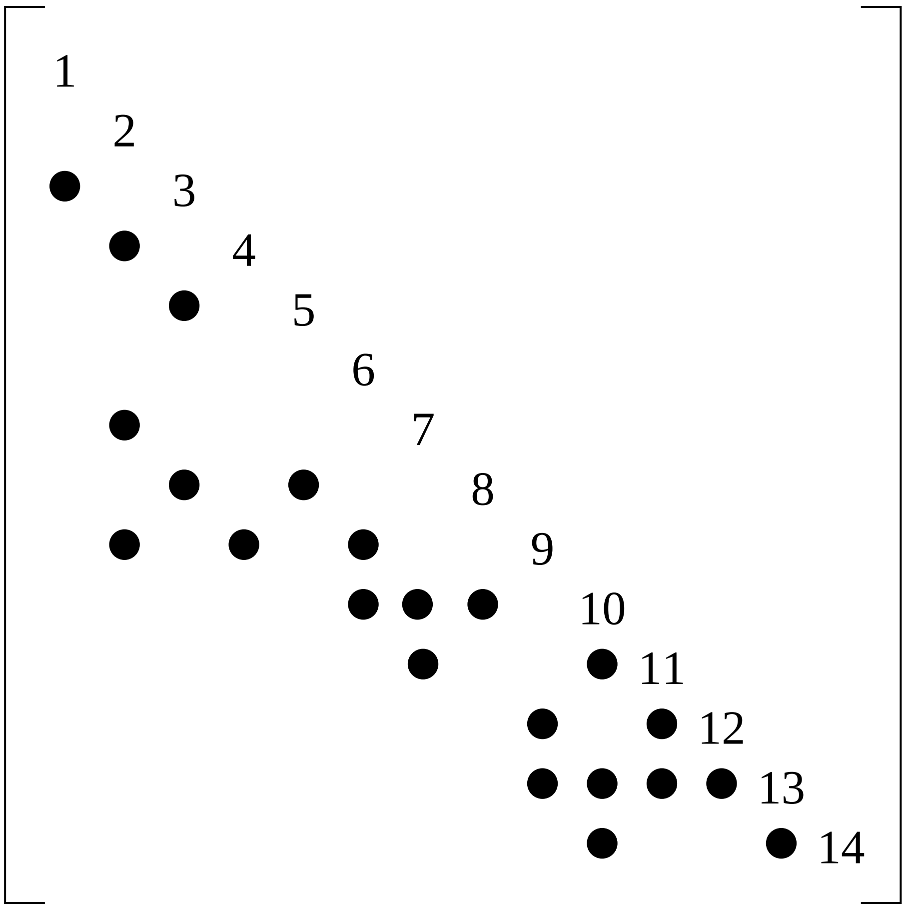



Graph $G_L$

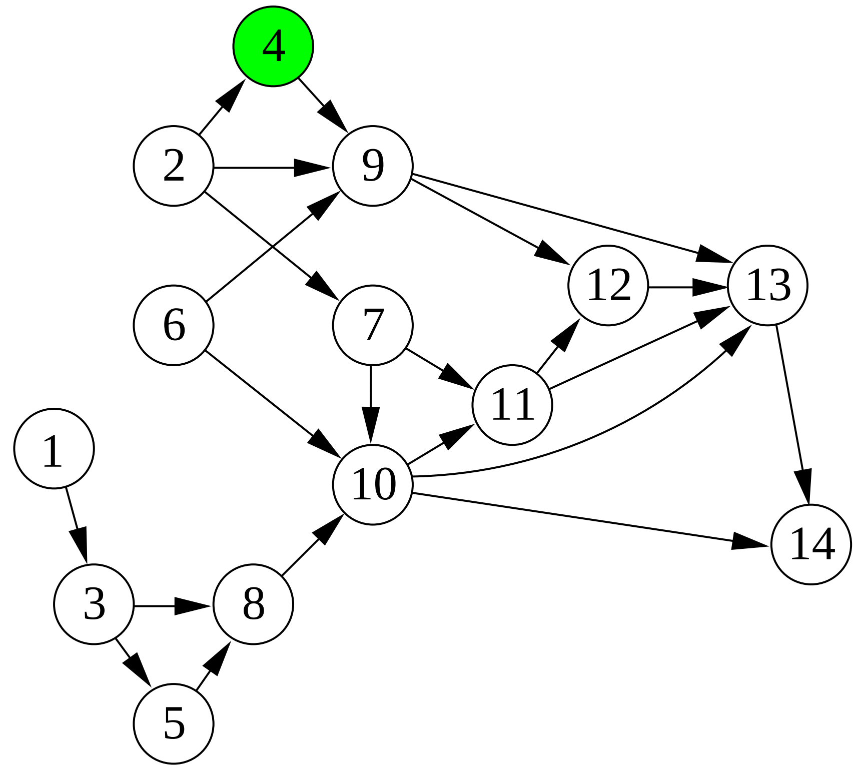


# Sparse lower triangular solve, x=L\b

Lower triangular matrix $L$



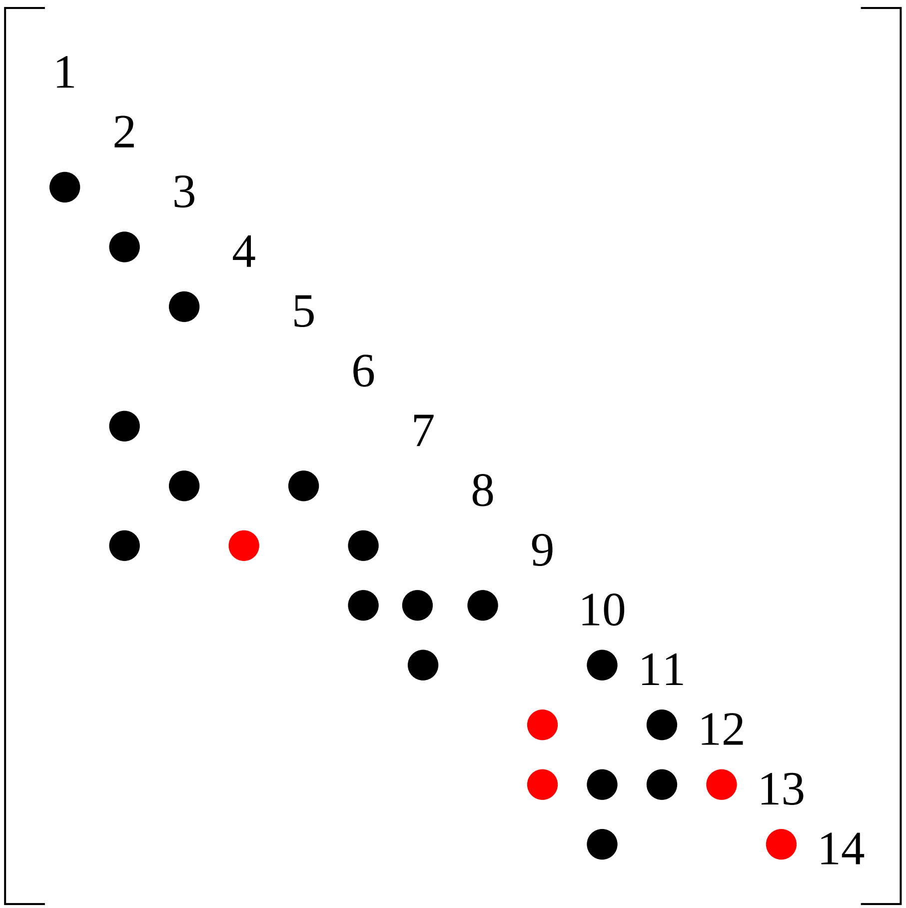

Graph $G_L$



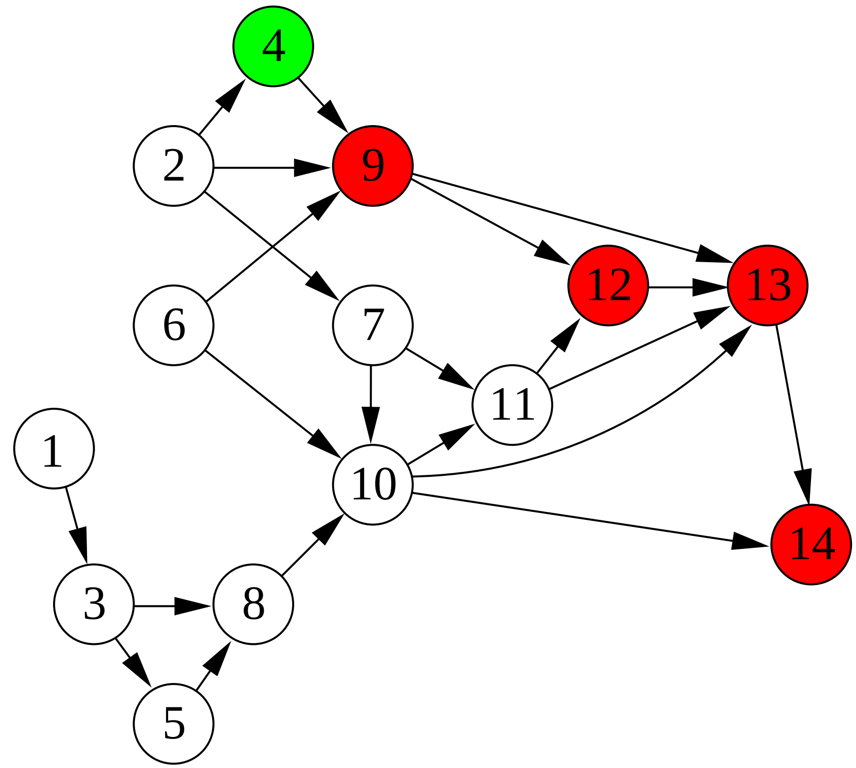

If $\mathcal{B} = \{4\}$

# Sparse lower triangular solve, x=L\b

Lower triangular matrix $L$




Graph $G_L$

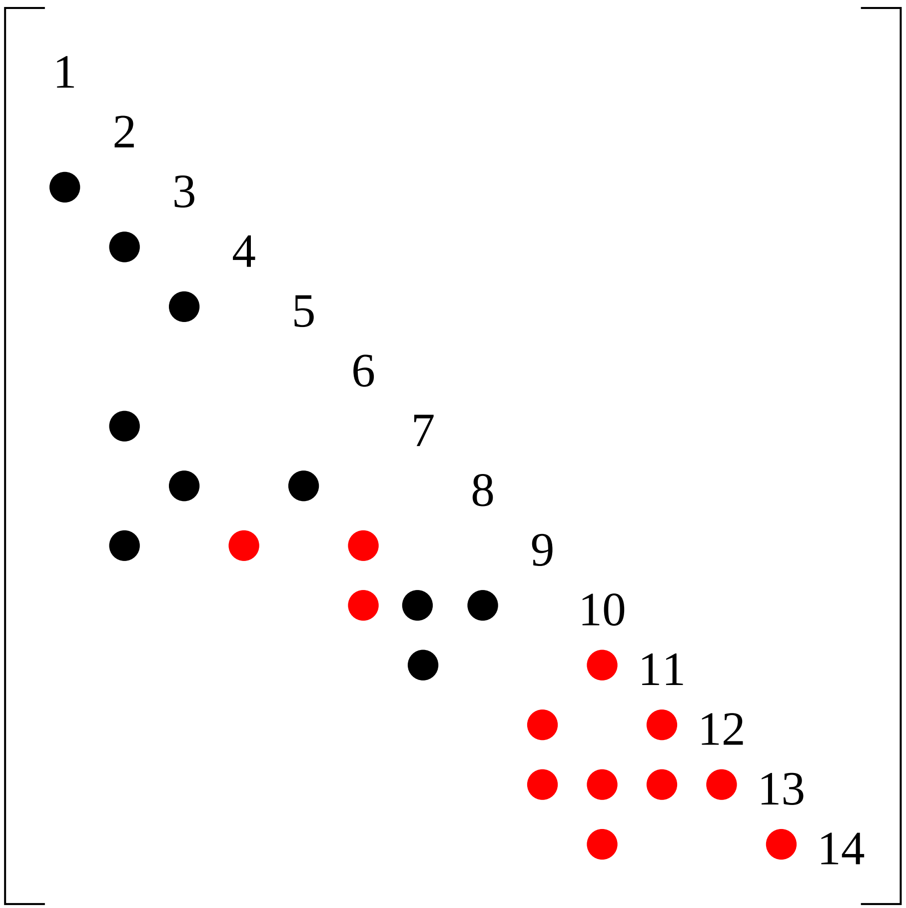



If $\mathcal{B} = \{4\}$

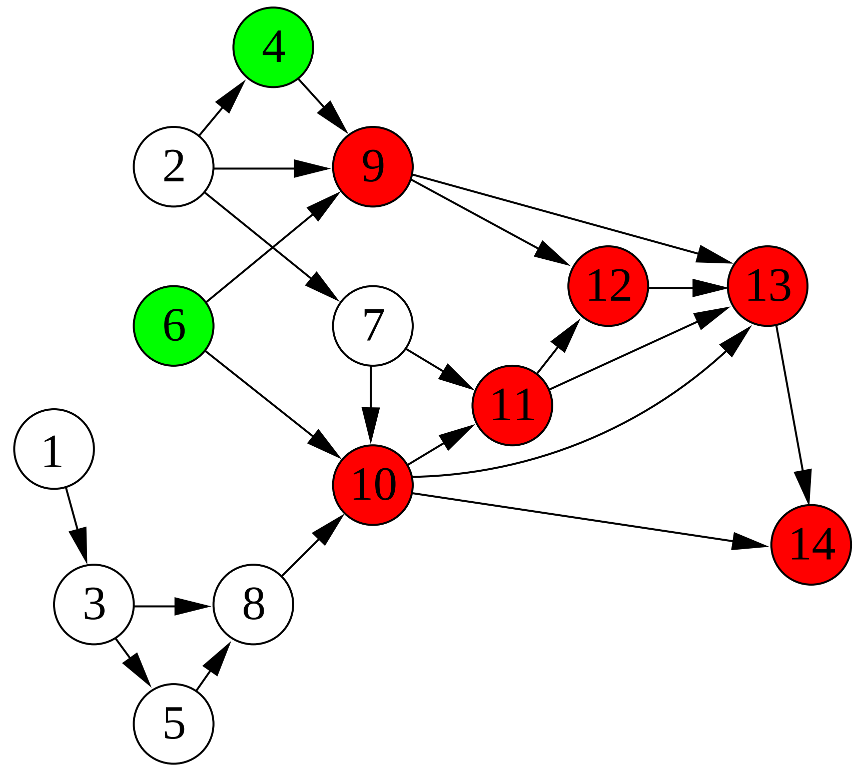
then $\mathcal{X} = \{4, 9, 12, 13, 14\}$

# **Sparse lower triangular solve,** x=L\b

Lower triangular matrix $L$




Graph $G_L$




If $\mathcal{B} = \{4, 6\}$
then $\mathcal{X} = \{6, 10, 11, 4, 9, 12, 13, 14\}$

# Sparse lower triangular solve, `x=L\b`

```
function x = lsolve(L,b)
    x = b
    for j = 1:n
        if (x(j) ≠ 0)
            x(j+1:n) = x(j+1:n) - L(j+1:n,j)*x(j)
```

Time: $O(n + \text{flops})$, need $\mathcal{X}$ to get $O(\text{flops})$

# Sparse lower triangular solve, `x=L\b`

function `x = lsolve(L,b)`

    $\mathcal{X} = \text{Reach}(L, \mathcal{B})$
    `x = b`
    for each $j$ in $\mathcal{X}$
        `x(j+1:n) = x(j+1:n) - L(j+1:n,j) * x(j)`

# Sparse lower triangular solve, `x=L\b`

function `x = lsolve(L,b)`

  $\mathcal{X} = \text{Reach}(L, \mathcal{B})$

  `x = b`

  for each $j$ in $\mathcal{X}$

    `x(j+1:n) = x(j+1:n) - L(j+1:n,j) * x(j)`

function $\mathcal{X} = \text{Reach}(\texttt{L}, \mathcal{B})$

  for each $i$ in $\mathcal{B}$ do

    if (node $i$ is unmarked) dfs($i$)

function dfs($j$)

  mark node $j$

  for each $i$ in $\mathcal{L}_j$ do

    if (node $i$ is unmarked) dfs($i$)

  push $j$ onto stack for $\mathcal{X}$

# Sparse lower triangular solve, `x=L\b`

function `x = lsolve(L,b)`
    $\mathcal{X} =$ Reach$(L, \mathcal{B})$
    `x = b`
    for each $j$ in $\mathcal{X}$
        `x(j+1:n) = x(j+1:n) - L(j+1:n,j) * x(j)`

function $\mathcal{X} =$ Reach$(\mathtt{L}, \mathcal{B})$
    for each $i$ in $\mathcal{B}$ do
        if (node $i$ is unmarked) dfs($i$)

function dfs($j$)
    mark node $j$
    for each $i$ in $\mathcal{L}_j$ do
        if (node $i$ is unmarked) dfs($i$)
    push $j$ onto stack for $\mathcal{X}$

Total time: $O(\text{flops})$

# Sparse lower triangular solve, `x=L\b`

function `x = lsolve(L,b)`
    $\mathcal{X} = \text{Reach}(L, \mathcal{B})$
    `x = b`
    for each $j$ in $\mathcal{X}$
        `x(j+1:n) = x(j+1:n) - L(j+1:n,j) * x(j)`

function $\mathcal{X} = \text{Reach}(\texttt{L}, \mathcal{B})$
    for each $i$ in $\mathcal{B}$ do
        if (node $i$ is unmarked) dfs($i$)

function dfs($j$)
    mark node $j$
    for each $i$ in $\mathcal{L}_j$ do
        if (node $i$ is unmarked) dfs($i$)
    push $j$ onto stack for $\mathcal{X}$

which can be less than $n$

# Sparse Cholesky, $LL^T = A$

$$\begin{bmatrix} L_{11} & \\ l_{12}^T & l_{22} \end{bmatrix} \begin{bmatrix} L_{11}^T & l_{12} \\ & l_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & a_{12} \\ a_{12}^T & a_{22} \end{bmatrix}$$

# Sparse Cholesky, $LL^T = A$

$$\begin{bmatrix} \mathbf{L_{11}} & \\ l_{12}^T & l_{22} \end{bmatrix} \begin{bmatrix} \mathbf{L_{11}^T} & l_{12} \\ & l_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{A_{11}} & a_{12} \\ a_{12}^T & a_{22} \end{bmatrix}$$

1. factorize $\mathbf{L_{11}L_{11}^T} = \mathbf{A_{11}}$

# Sparse Cholesky, $LL^T = A$

$$\begin{bmatrix} \mathbf{L_{11}} & \\ l_{12}^T & l_{22} \end{bmatrix} \begin{bmatrix} L_{11}^T & \mathbf{l_{12}} \\ & l_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & \mathbf{a_{12}} \\ a_{12}^T & a_{22} \end{bmatrix}$$

1. factorize $L_{11}L_{11}^T = A_{11}$
2. solve $\mathbf{L_{11}l_{12}} = \mathbf{a_{12}}$ for $l_{12}$

# Sparse Cholesky, $LL^T = A$

$$\begin{bmatrix} L_{11} & \\ \mathbf{l_{12}^T} & \mathbf{l_{22}} \end{bmatrix} \begin{bmatrix} L_{11}^T & \mathbf{l_{12}} \\ & \mathbf{l_{22}} \end{bmatrix} = \begin{bmatrix} A_{11} & a_{12} \\ a_{12}^T & \mathbf{a_{22}} \end{bmatrix}$$

1. factorize $L_{11}L_{11}^T = A_{11}$
2. solve $L_{11}l_{12} = a_{12}$ for $l_{12}$
3. $\mathbf{l_{22}} = \sqrt{\mathbf{a_{22}} - \mathbf{l_{12}^T}\mathbf{l_{12}}}$

# Sparse Cholesky, $LL^T = A$

$$\begin{bmatrix} L_{11} & \\ l_{12}^T & l_{22} \end{bmatrix} \begin{bmatrix} L_{11}^T & l_{12} \\ & l_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & a_{12} \\ a_{12}^T & a_{22} \end{bmatrix}$$

1. factorize $L_{11}L_{11}^T = A_{11}$
2. solve $L_{11}l_{12} = a_{12}$ for $l_{12}$
3. $l_{22} = \sqrt{a_{22} - l_{12}^T l_{12}}$

for $k$ = 1 to $n$

solve $L_{11}l_{12} = a_{12}$ for $l_{12}$

$l_{22} = \sqrt{a_{22} - l_{12}^T l_{12}}$

# Sparse Cholesky, $LL^T = A$

$$\begin{bmatrix} L_{11} & \\ l_{12}^T & l_{22} \end{bmatrix} \begin{bmatrix} L_{11}^T & l_{12} \\ & l_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & a_{12} \\ a_{12}^T & a_{22} \end{bmatrix}$$

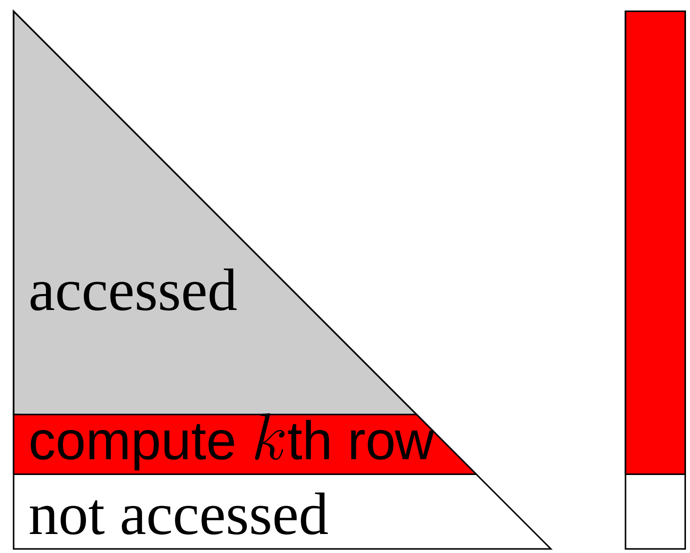
1. factorize $L_{11}L_{11}^T = A_{11}$
2. solve $L_{11}l_{12} = a_{12}$ for $l_{12}$
3. $l_{22} = \sqrt{a_{22} - l_{12}^T l_{12}}$

for $k$ = 1 to $n$

solve $L_{11}l_{12} = a_{12}$ for $l_{12}$

$l_{22} = \sqrt{a_{22} - l_{12}^T l_{12}}$

an up-looking method

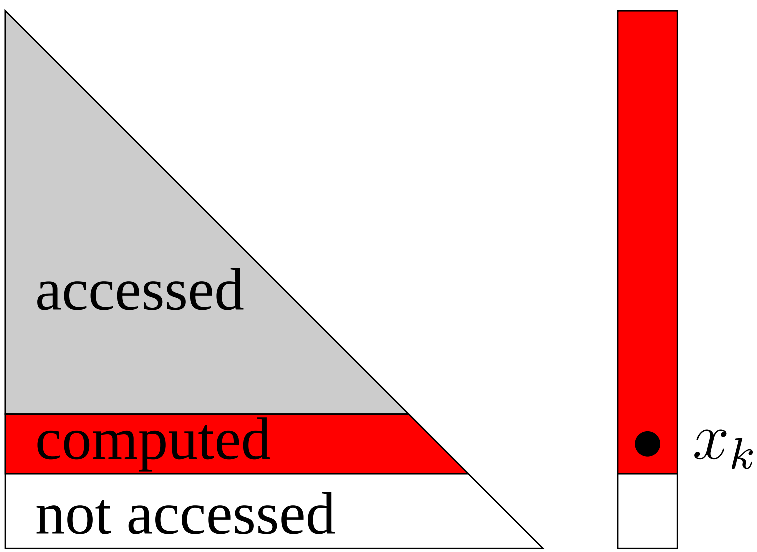


# Sparse Cholesky: etree

- elimination tree
- arises in many direct methods
  - Compute nonzero pattern of `x=L\b` for a Cholesky `L` in time $O(|x|)$, the number of nonzeros in `x`
  - ...

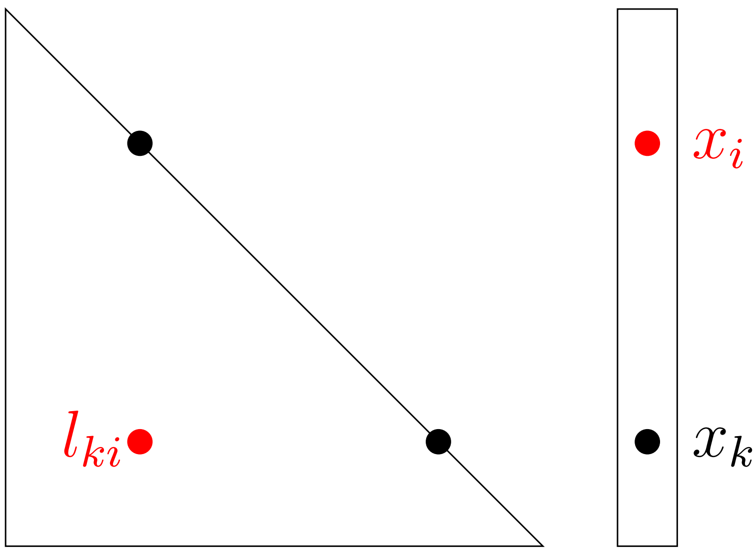
# Sparse Cholesky: etree

Elimination tree $\mathcal{T}$: pruning the graph of $L$.
Consider computing $k$th row of $L$:



# Sparse Cholesky: etree

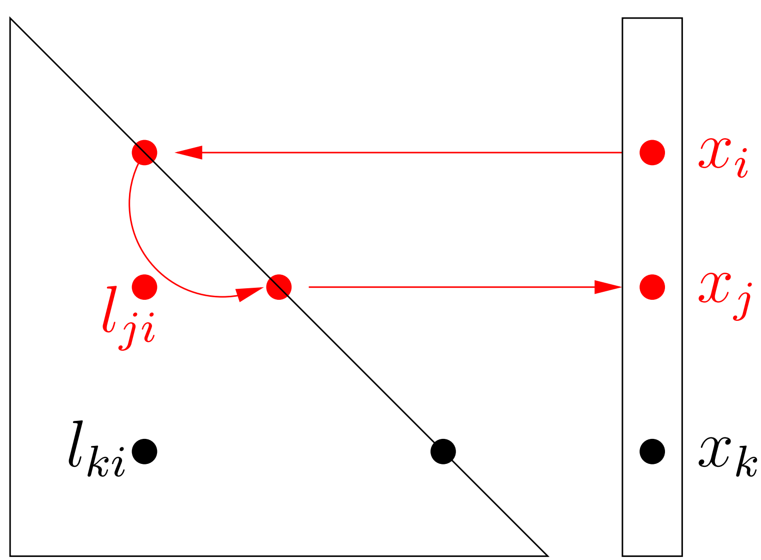
Elimination tree $\mathcal{T}$: pruning the graph of $L$.
Consider computing $k$th row of $L$:




- $l_{ki} \neq 0 \Leftrightarrow x_i \neq 0$

# Sparse Cholesky: etree

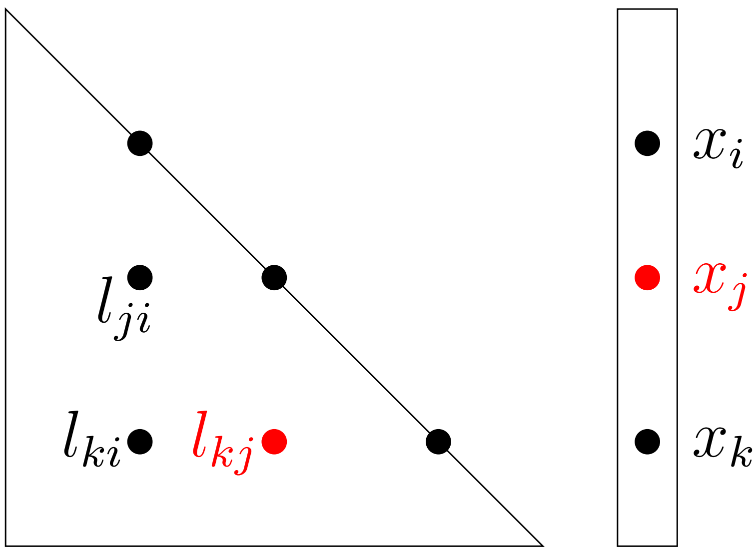
Elimination tree $\mathcal{T}$: pruning the graph of $L$.
Consider computing $k$th row of $L$:




- $l_{ki} \neq 0 \Leftrightarrow x_i \neq 0$
- ($l_{ji} \neq 0$ and $x_i \neq 0$) $\Rightarrow x_j \neq 0$

# Sparse Cholesky: etree

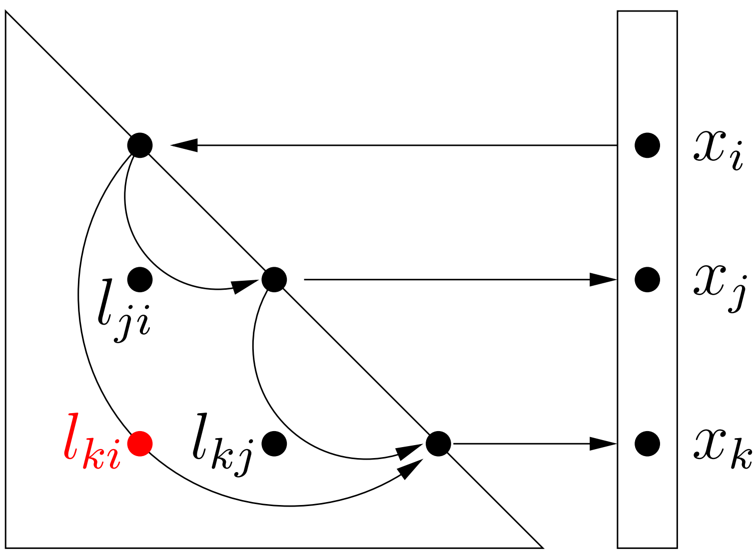
Elimination tree $\mathcal{T}$: pruning the graph of $L$.
Consider computing $k$th row of $L$:




- $l_{ki} \neq 0 \Leftrightarrow x_i \neq 0$
- ($l_{ji} \neq 0$ and $x_i \neq 0$) $\Rightarrow x_j \neq 0$
- $l_{kj} \neq 0 \Leftrightarrow x_j \neq 0$

# Sparse Cholesky: etree

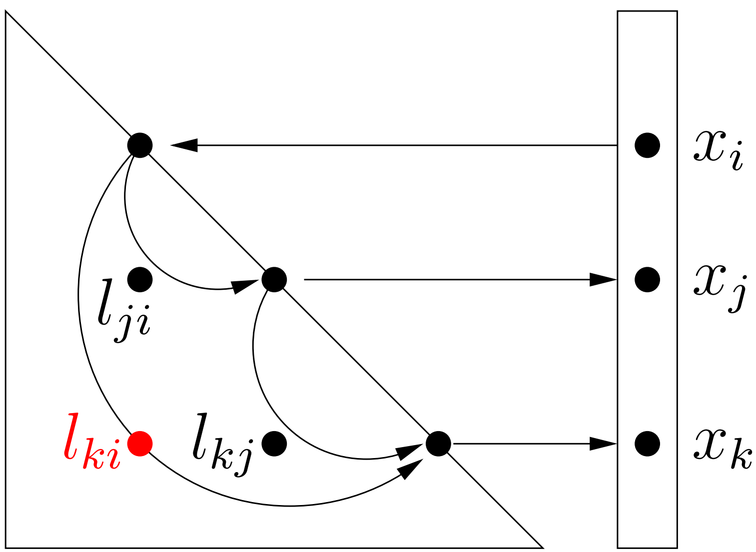
Elimination tree $\mathcal{T}$: pruning the graph of $L$.
Consider computing $k$th row of $L$:




- $l_{ki} \neq 0 \Leftrightarrow x_i \neq 0$
- ($l_{ji} \neq 0$ and $x_i \neq 0$) $\Rightarrow x_j \neq 0$
- $l_{kj} \neq 0 \Leftrightarrow x_j \neq 0$
- Thus, $l_{ki}$ redundant for $\mathcal{X} = \text{Reach}(\mathcal{B})$.

# Sparse Cholesky: etree

Elimination tree $\mathcal{T}$: pruning the graph of $L$.
Consider computing $k$th row of $L$:

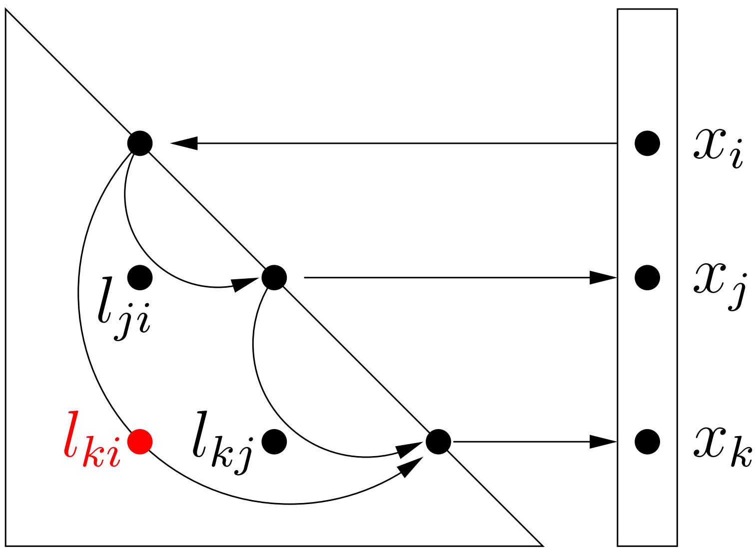



- $l_{ki} \neq 0 \Leftrightarrow x_i \neq 0$
- ($l_{ji} \neq 0$ and $x_i \neq 0$) $\Rightarrow x_j \neq 0$
- $l_{kj} \neq 0 \Leftrightarrow x_j \neq 0$
- Thus, $l_{ki}$ redundant for $\mathcal{X} = \text{Reach}(b)$.

- parent$(i) = \min\{j > i \,|\, l_{ji} \neq 0\}$; other edges redundant

# Sparse Cholesky: etree

Elimination tree $\mathcal{T}$: pruning the graph of $L$.
Consider computing $k$th row of $L$:




- $l_{ki} \neq 0 \Leftrightarrow x_i \neq 0$
- ($l_{ji} \neq 0$ and $x_i \neq 0$) $\Rightarrow x_j \neq 0$
- $l_{kj} \neq 0 \Leftrightarrow x_j \neq 0$
- Thus, $l_{ki}$ redundant for $\mathcal{X} = \text{Reach}(b)$.

- parent$(i) = \min\{j > i \,|\, l_{ji} \neq 0\}$; other edges redundant
- $\mathcal{L}_{k*} = \text{Reach}(A_{1:k,k})$ in $O(|\mathcal{L}_{k*}|)$ time

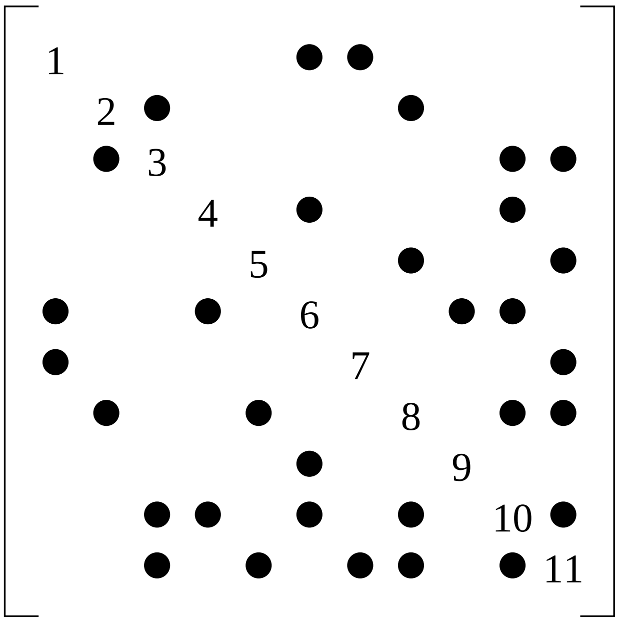
# Sparse Cholesky: etree

$$A$$

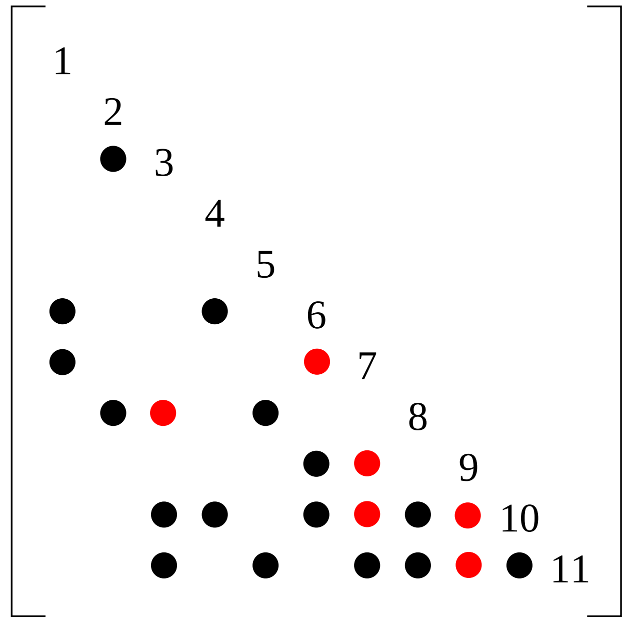
$$\begin{bmatrix}
1 & & & & & \bullet & \bullet & & & & \\
& 2 & \bullet & & & & & \bullet & & & \\
& \bullet & 3 & & & & & & & \bullet & \bullet \\
& & & 4 & & \bullet & & & & \bullet & \\
& & & & 5 & & & \bullet & & & \bullet \\
\bullet & & & \bullet & & 6 & & & \bullet & \bullet & \\
\bullet & & & & & & 7 & & & & \bullet \\
& \bullet & & & \bullet & & & 8 & & \bullet & \bullet \\
& & & & & \bullet & & & 9 & & \\
& & \bullet & \bullet & & \bullet & & \bullet & & 10 & \bullet \\
& & \bullet & & \bullet & & \bullet & \bullet & & \bullet & 11
\end{bmatrix}$$

# Sparse Cholesky: etree

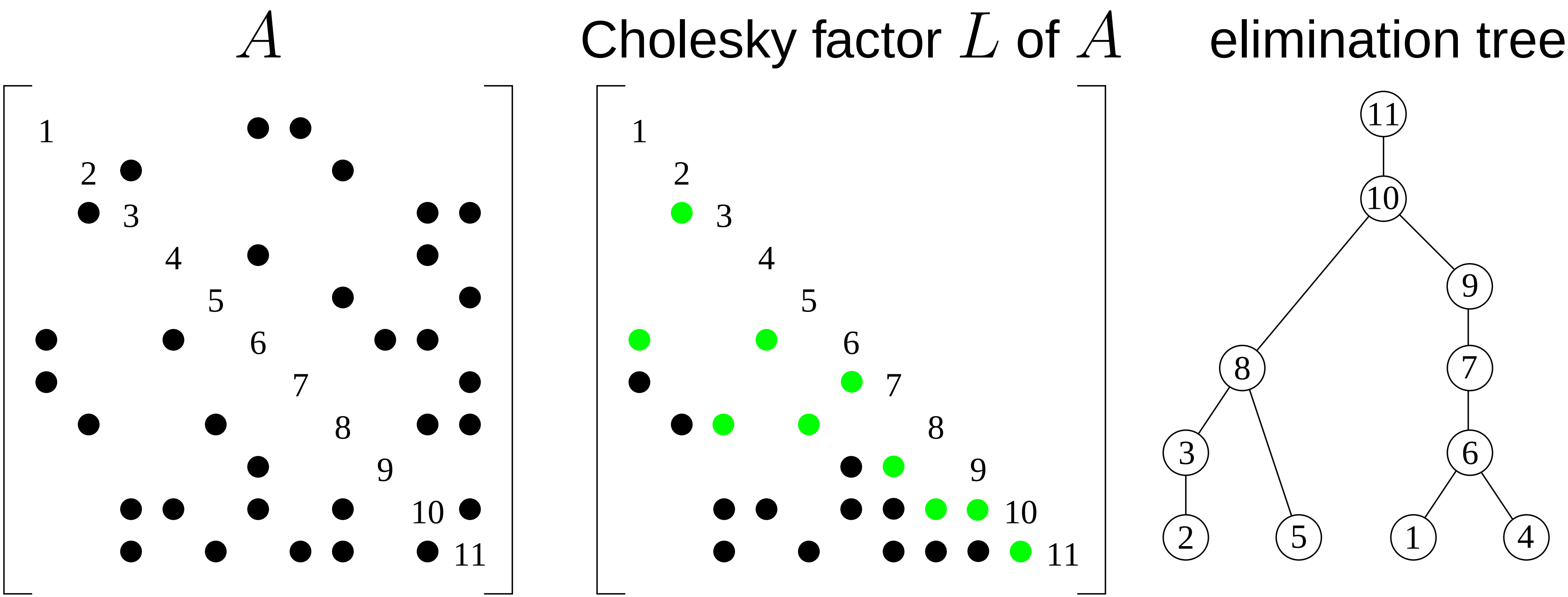
$A$




Cholesky factor $L$ of $A$



# Sparse Cholesky: etree




Can read off zero patterns of L by zero patterns of A + etree.

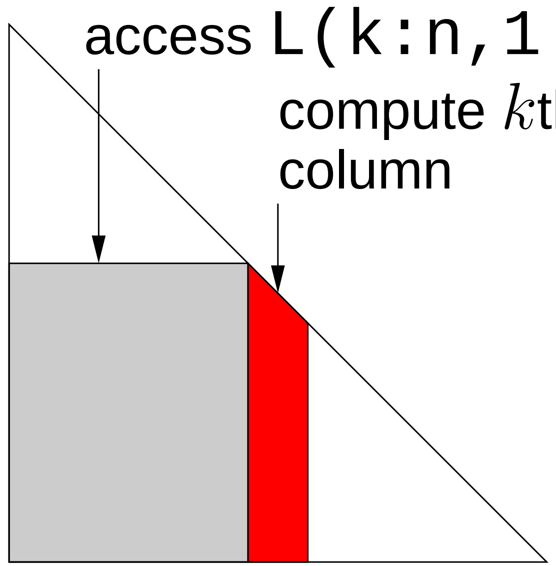
Problem: Why we can compute zero patterns of L in O~(n^2) time, but not L itself?

# Sparse Cholesky: overview

- Symbolic analysis:
  - fill-reducing ordering, $\bar{A} = PAP^T = LL^T$
  - etree of $\bar{A}$: nearly $O(|A|)$
  - depth-first postordering of etree: $O(n)$
  - column counts of $L$: nearly $O(|A|)$
  - some methods find $\mathcal{L}$: $O(|L|)$ or less
- Numeric factorization:
  - up-looking
  - left-looking, supernodal

Postorder (Left, Right, Root)

# Sparse Cholesky: left-looking



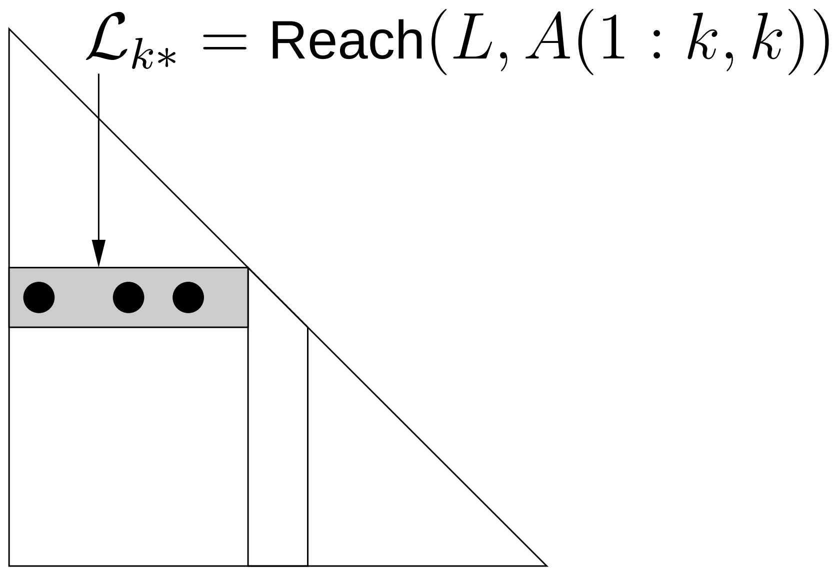

for $k = 1$ to $n$

```
    x = A(k:n,k)
```

    for each $j$ in Reach$(L, A(1:k,k))$

```
        x(k:n) = x(k:n) - L(k:n,j) * L(k,j)
    L(k,k) = sqrt(x(k))
    L(k+1:n,k) = x(k) / L(k,k)
```

# Sparse Cholesky: left-looking




for $k = 1$ to $n$

```
    x = A(k:n,k)
```

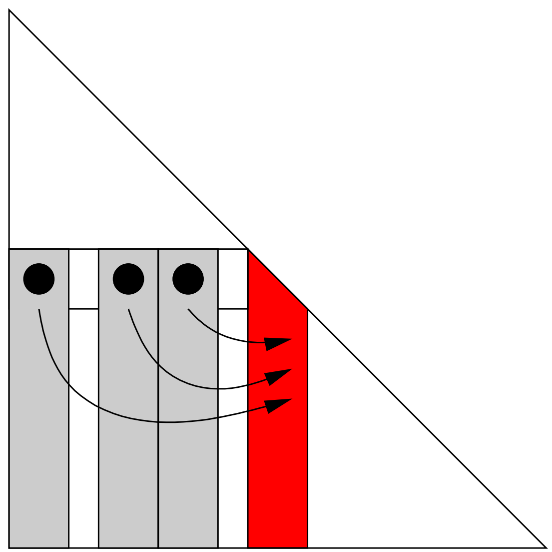
    for each $j$ in Reach$(L, A(1:k,k))$

        ...

    ...

    ...

# Sparse Cholesky: left-looking



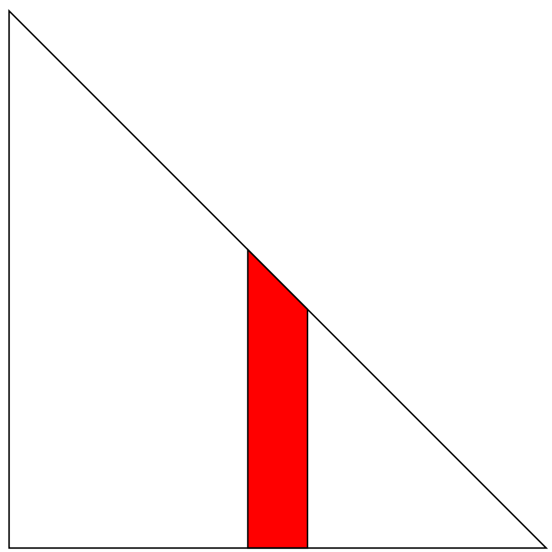
for $k = 1$ to $n$

```
    x = A(k:n,k)
```

    for each $j$ in Reach$(L, A(1:k,k))$

```
        x(k:n) = x(k:n) - L(k:n,j) * L(k,j)
    ...
    ...
```

# Sparse Cholesky: left-looking



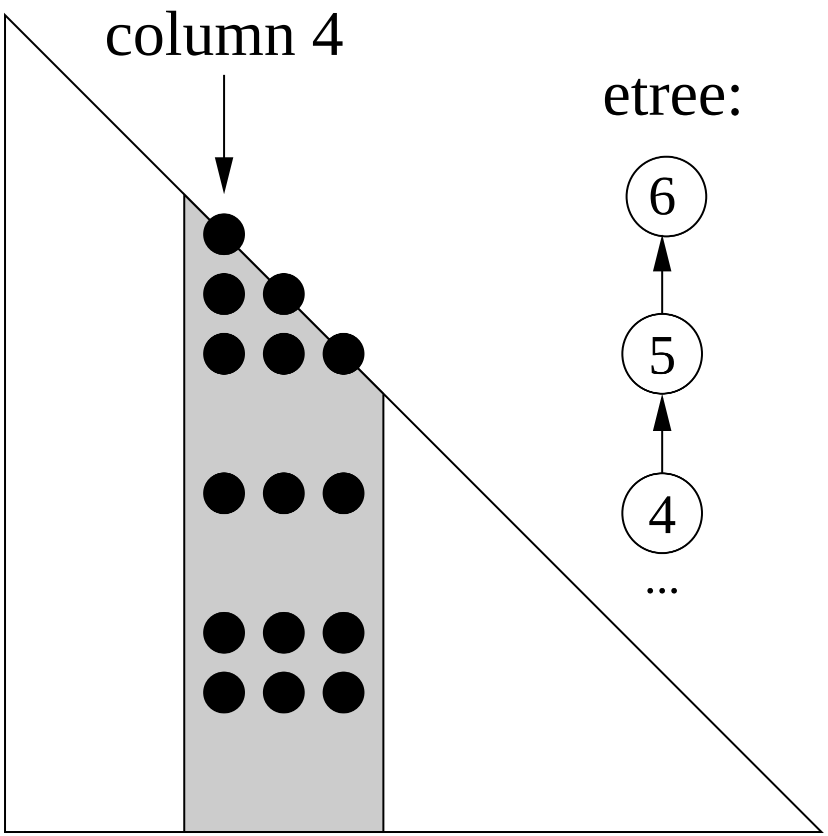
for $k = 1$ to $n$

```
    x = A(k:n,k)
```

    for each $j$ in Reach$(L, A(1:k,k))$

```
        x(k:n) = x(k:n) - L(k:n,j) * L(k,j)
    L(k,k) = sqrt(x(k))
    L(k+1:n,k) = x(k) / L(k,k)
```

# Sparse Cholesky: supernodal




- Adjacent columns of $L$ often have identical pattern
- a chain in the elimination tree
- can exploit dense submatrix operations

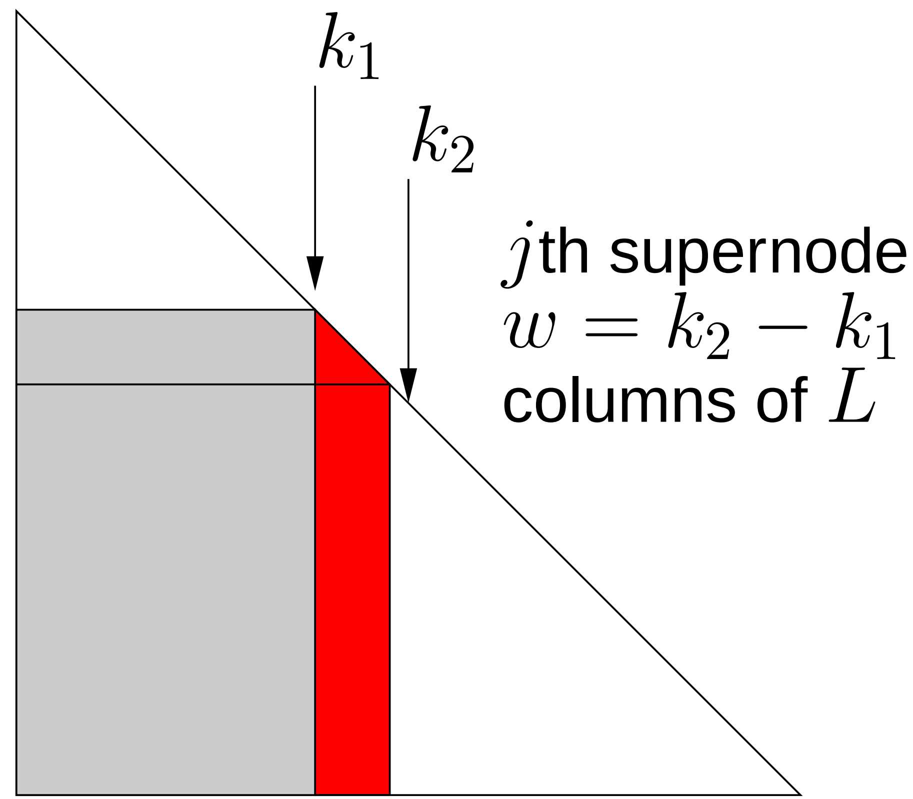
**Theorem 4.13** (George and Liu [89]). *If $\mathcal{L}_j$ denotes the nonzero pattern of the $j$th column of $L$, and $\mathcal{A}_j$ denotes the nonzero pattern of the strictly lower triangular part of the $j$th column of $A$, then*

$$\mathcal{L}_j = \mathcal{A}_j \cup \{j\} \cup \left( \bigcup_{j=parent(s)} \mathcal{L}_s \setminus \{s\} \right). \tag{4.3}$$

# Sparse Cholesky: supernodal

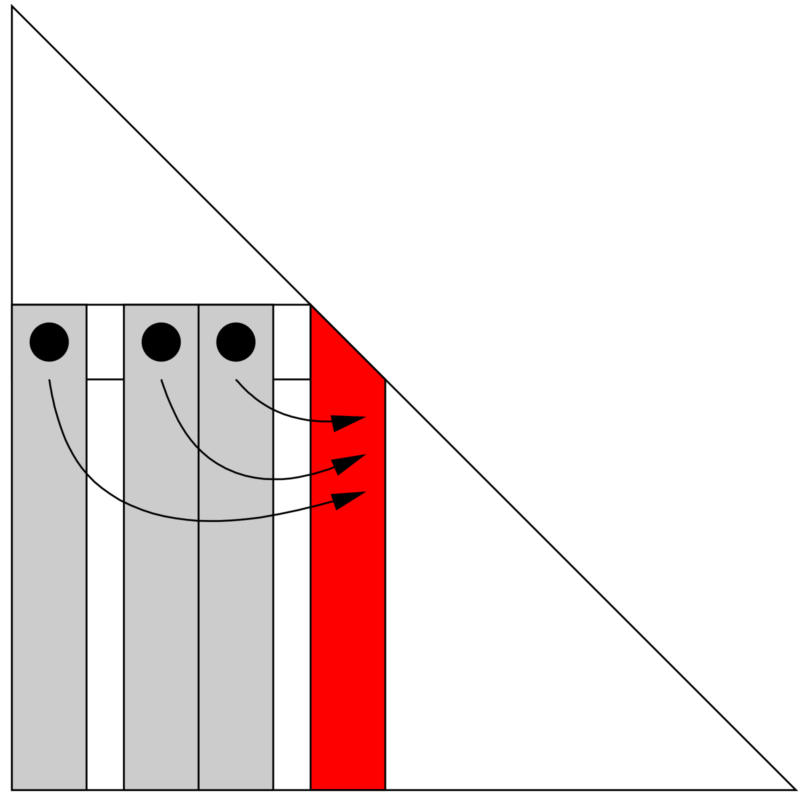
block left-looking

- for $j$th supernode:



# Sparse Cholesky: supernodal

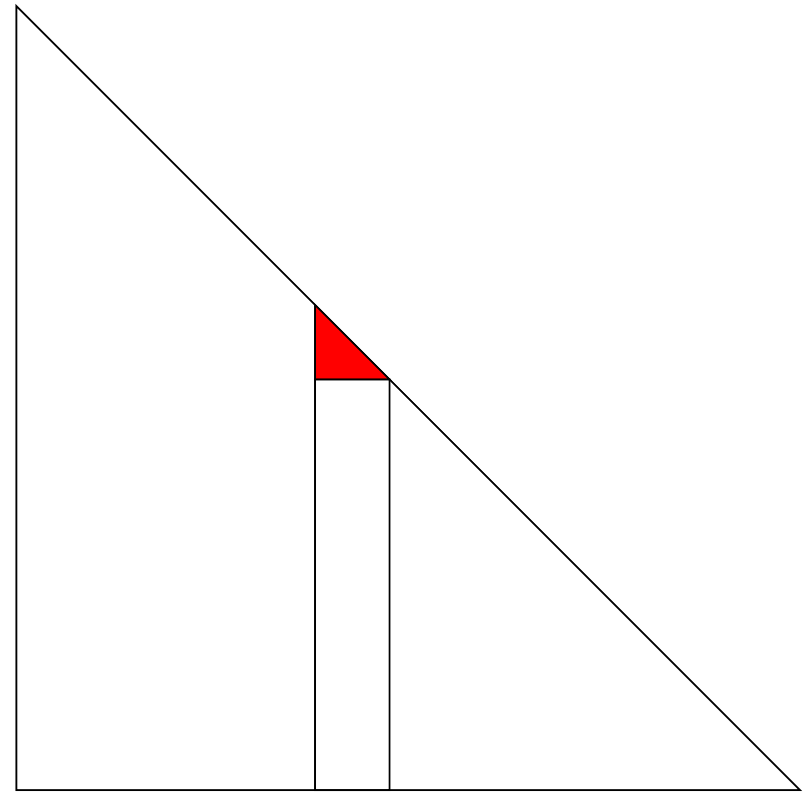
block left-looking



- for $j$th supernode:
- (1) sparse block matrix multiply

# Sparse Cholesky: supernodal

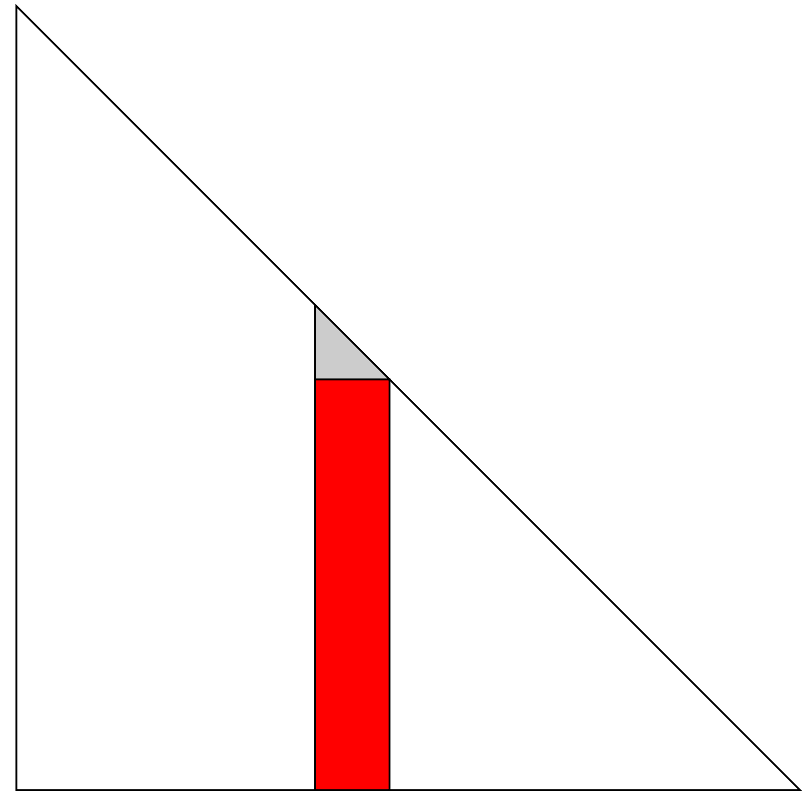
block left-looking



- for $j$th supernode:
- (1) sparse block matrix multiply
- (2) dense Cholesky

# Sparse Cholesky: supernodal

block left-looking



- for $j$th supernode:
- (1) sparse block matrix multiply
- (2) dense Cholesky
- (3) dense block $Lx = b^T$ solve