Elimination tree

Fanny Dufossé & Bora Uçar

Inria, Grenoble Rhône-Alpes & CNRS and ENS Lyon, France

CR12: Combinatorial scientific computing, September 2018
http://perso.ens-lyon.fr/bora.ucar/CR12/
Outline

1. The elimination tree
   - Definition and properties
   - Determination
   - Determining row and column counts
   - Some other properties

2. Closing
The role of elimination tree

There are many uses of elimination trees [Schreiber,’82] for sparse factorization.

Our focus: its use in Cholesky factorization $LL^T$ of a given symmetric positive matrix $A$.

Nonzeros in $L$ can be characterized in terms of paths in the e-tree.

In particular, the row and column structures (or the number of nonzeros in each row and column of $L$) can be expressed in terms of the elimination tree → Predicting the structure of $L$. 
The role of elimination tree

One can build the graph of $L$ using the elimination process.
This gives a running time proportional to the size of the filled-in graph $G^+(A)$ or the number of nonzeros in $L$. Not bad (Why?), but we want more.

We will see the use of elimination tree to implement this, but here is the essential idea and the algorithm that follows: ⟨⟨On the board⟩⟩
Set-up

Reminder

- A spanning tree of a connected graph \( G = (V, E) \) is a tree \( T = (V, F) \), such that \( F \subseteq E \).
- A topological ordering of a rooted tree is an ordering that numbers children vertices before their parent.
- A postorder is a topological ordering which numbers the vertices in any subtree consecutively.

Let \( A \) be an \( n \times n \) symmetric positive-definite and irreducible matrix, \( A = LL^T \) its Cholesky factorization, and \( G^+(A) \) its filled graph (graph of \( L + L^T \)).

We assume an ordering (at least partial) on \( A \).
A first definition

Since $A$ is irreducible, each of the first $n - 1$ columns of $L$ has at least one off-diagonal nonzero (proved last time).

For each column $j < n$ of $L$, remove all the nonzeros in the column $j$ except the first one below the diagonal.

Let $L_t$ denote the remaining structure and consider the matrix $F_t = L_t + L_t^T$. The graph $G(F_t)$ is a tree called the elimination tree.
A first definition

The elimination tree of $A$ is a spanning tree of $G^+(A)$ satisfying the relation $\text{parent}[j] = \min\{i > j : \ell_{ij} \neq 0\}$.

$$F = \begin{pmatrix}
  a & \bullet & \bullet & \bullet \\
  b & c & d & \bullet \\
  \bullet & \bullet & e & \bullet \\
  \bullet & \bullet & f & \bullet \\
  \bullet & \bullet & \bullet & \bullet \\
  \bullet & \bullet & \bullet & \bullet \\
  \bullet & \bullet & \bullet & \bullet \\
  \bullet & \bullet & \bullet & \bullet \\
  \bullet & \bullet & \bullet & \bullet \\
  \bullet & \bullet & \bullet & \bullet \\
\end{pmatrix}$$

$$F_t = \begin{pmatrix}
  a & \bullet & \bullet & \bullet \\
  b & c & d & \bullet \\
  \bullet & \bullet & e & \bullet \\
  \bullet & \bullet & f & \bullet \\
  \bullet & \bullet & \bullet & \bullet \\
  \bullet & \bullet & \bullet & \bullet \\
  \bullet & \bullet & \bullet & \bullet \\
  \bullet & \bullet & \bullet & \bullet \\
  \bullet & \bullet & \bullet & \bullet \\
  \bullet & \bullet & \bullet & \bullet \\
\end{pmatrix}$$
Columnwise Cholesky (left-looking)

for column \( j := 1 \) to \( n \) do

begin

\[
\begin{pmatrix}
  t_j \\
  \vdots \\
  l_{nj}
\end{pmatrix}
:=
\begin{pmatrix}
  a_{jj} \\
  \vdots \\
  a_{nj}
\end{pmatrix}
- \sum_{k<j} l_{jk}
\begin{pmatrix}
  l_{jk} \\
  \vdots \\
  l_{nk}
\end{pmatrix}
\]

\[
\begin{pmatrix}
  l_{jj} \\
  \vdots \\
  l_{nj}
\end{pmatrix}
:=
\frac{1}{\sqrt{t_j}}
\begin{pmatrix}
  t_j \\
  \vdots \\
  t_n
\end{pmatrix}
\]

end

\[
\begin{pmatrix}
  t_7 \\
  t_8 \\
  0 \\
  t_{10}
\end{pmatrix}
= \begin{pmatrix}
  a_{77} \\
  a_{87} \\
  0 \\
  a_{10,7}
\end{pmatrix}
- l_{71} \begin{pmatrix}
  \bullet \\
  \bullet \\
  \bullet \\
  \bullet
\end{pmatrix}
- l_{72} \begin{pmatrix}
  \bullet \\
  \bullet \\
  \bullet \\
  \bullet
\end{pmatrix}
- l_{74} \begin{pmatrix}
  \bullet \\
  \bullet \\
  \bullet \\
  \bullet
\end{pmatrix}
- l_{75} \begin{pmatrix}
  \bullet \\
  \bullet \\
  \bullet \\
  \bullet
\end{pmatrix}
\]
A second definition: Represents column dependencies

- Dependency between columns of $L$:
  - Column $i > j$ depends on column $j$ iff $\ell_{ij} \neq 0$
  - Use a directed graph to express this dependency (edge from $j$ to $i$, if column $i$ depends on column $j$)
  - Simplify redundant dependencies (transitive reduction)

- The transitive reduction of the directed filled graph gives the elimination tree structure [Remove a directed edge ($j, i$) if there is a path of length greater than one from $j$ to $i$.]
Directed filled graph and its transitive reduction

Directed filled graph

Transitive reduction

T(A)
A third definition: DFS tree

**Theorem**

The elimination tree $T(A)$ of a connected graph $G(A)$ is a depth-first search tree of the filled graph $G^+(A)$.

**Proof.**

Let $x_1, x_2, \ldots, x_n$ be the node ordering of $G^+(A)$. Consider the depth-first search subject to the following tie-breaking rule: when there is a choice of more than one node to explore next, always pick the one with largest subscript. With this additional rule the depth-first search will construct $T(A)$. 
A depth-first search tree of the filled graph

Any DFS on an undirected graph produces only Tree and Back edges.
Path characterization of filled edges

Nonzeros of $L$

If $l_{ij} \neq 0$, then node $x_i$ is an ancestor of $x_j$ in $T(A)$.

Corollary: Some zeros of $L$

Let $T[x_i]$ and $T[x_j]$ be two disjoint subtrees of $T(A)$. Then $l_{st} = 0$ for any $x_s \in T[x_i]$ and $x_t \in T[x_j]$.

- Simply because there is no cross edge in the DFS tree of an undirected graph.
- The corollary also shows that the elimination tree expresses potential parallelism (resulting from the sparsity).
Fill-in entries

Fill-path theorem [Rose, Tarjan, Lueker’76]

Let $G = (V, E, \alpha)$ be an ordered graph. Then $(v, w)$ is an edge of $G^*_\alpha = (V, E \cup F(G_\alpha))$ iff there exists a path $\mu = [v = v_1, v_2, \ldots, v_{k+1} = w]$ in $G$ such that

$$\alpha^{-1}(v_i) < \min\{\alpha^{-1}(v), \alpha^{-1}(w)\}, \quad 2 \leq i \leq k$$

Restating using the elimination tree

Let $i > j$. Then $\ell_{ij} \neq 0$ iff there exists a path $x_i, x_{p_1}, \ldots, x_{p_k}, x_j$ in the graph of $A$ such that $\{x_{p_1}, \ldots, x_{p_k}\} \subseteq T[x_j]$. Here $T[x_j]$ is the set of nodes in the subtree rooted at $x_j$. 
Row structure of the Cholesky factor

Row structure theorem

Let $i > j > k$. We have $\ell_{ij} \neq 0$ iff node $x_j$ is an ancestor of some node $x_k$ in the elimination tree with $a_{ik} \neq 0$.

Define $T_r[x_i]$ to be the structure of the $i$th row of $L$, i.e., $T_r[x_i] = \{x_j : \ell_{ij} \neq 0, j \leq i\}$. We have $T_r[x_i] \subseteq T[x_i]$.

A previous theorem ($\ell_{ij} \neq 0$ iff $x_i$ is an ancestor of $x_j$) states $\subseteq T[x_i]$. In other words, $T_r[x_i]$ is obtained from $T[x_i]$ by pruning: Find the leaves of $T_r[x_i]$ and prune $T[x_i]$ at those leaves.
Row structure of the Cholesky factor

Corollary

The node $x_j$ is a leaf node in the row subtree $T_r[x_i]$ iff $a_{ij} \neq 0$ and for every proper descendant $x_k$ of $x_j$, we have $a_{ik} = 0$. 

\[ F = \begin{pmatrix} 
 a & \cdot & \cdot & \cdot & \cdot \\
 b & c & \cdot & \cdot & \cdot \\
 \cdot & d & e & \cdot & \cdot \\
 \cdot & \cdot & f & g & \cdot \\
 \cdot & \cdot & \cdot & h & i \\
 \cdot & \cdot & \cdot & \cdot & j 
\end{pmatrix} \]
**Row structure of the Cholesky factor**

**Remark**

Each leaf $x_j$ of row subtree $T_r[x_i]$ corresponds to an edge $\{x_i, x_j\}$ in the original graph of $A$.

```
marker[x_i] ← i
for k < i and a_{ik} ≠ 0 do
  j ← k
  while marker[x_j] ≠ i do
    marker[x_j] ← i
    j ← parent(j)
```

![Diagram of row structures and Cholesky factorization](image)

\[
F = \begin{pmatrix}
  a & \bullet & \bullet & \bullet & \bullet & \bullet \\
  \bullet & b & \bullet & \bullet & \bullet & \bullet \\
  \bullet & \bullet & c & \bullet & \bullet & \bullet \\
  \bullet & \bullet & \bullet & d & \bullet & \bullet \\
  \bullet & \bullet & \bullet & \bullet & e & \bullet \\
  \bullet & \bullet & \bullet & \bullet & \bullet & f
\end{pmatrix}
\]
Row structure of the Cholesky factor: Efficient fill-in computation

Compare Tarjan-Yannakakis’84 and e-tree-based

**Fill-in computation.**

```plaintext
local v, w, x;
for i ∈ [1, n] →
  w := α⁻¹(i); f(w) := w; index(w) := i
for {v, w} ∈ E such that α(v) < i →
  x := v;
  do index(x) < i →
    index(x) := i;
    add {x, w} to E ∪ F(α);
    x := f(x)
  od;
  if f(x) = x → f(x) := w fi
rof
rof;
```

**E-tree based**

langle On the board angle
Column structure of the Cholesky factor

Column structure theorem

The structure of column $j$ of $L$, i.e., $\{x_i : l_{ij} \neq 0, i \geq j\}$ is given by

$$\text{Adj}_{G(A)}(T[x_j]) \cup \{x_j\}$$

Here, $\text{Adj}_{G(A)}(S) = \{x \notin S : x \in \text{Adj}_{G(A)}(v) \text{ for some } v \in S\}$

Remark: Since $i \in L_{*j}$ iff $j \in L_{i*}$, the column $j$ is included in row subtrees $T_r[i]$ that contain $j$. We can traverse all row subtrees $T_r[i]$ and determine the pattern of $L_{*j}$ by noting each $i$ such that $T_r[i]$ contains $j$.

$$\text{Adj}_{G(A)}(T[c]) \cup \{c\} = \{c, g, h\}$$

$$F = \begin{pmatrix}
    a & b & \cdot & \cdot & \cdot \\
    \cdot & c & d & \cdot & \cdot \\
    \cdot & \cdot & f & g & \cdot \\
    \cdot & \cdot & \cdot & h & \cdot \\
    \cdot & \cdot & \cdot & \cdot & i \\
\end{pmatrix}$$
Column structure of the Cholesky factor

Column structure theorem

\[ \{ x_i : \ell_{ij} \neq 0, i \geq j \} = \text{Adj}_G(\mathbf{A})(T[x_j]) \cup \{ x_j \} \]

Why? First, since \( \ell_{jj} \neq 0 \), we have to have \( x_j \) in the structure of \( L_{**j} \).

Reminder: Row structure theorem

Let \( i > j > k \). We have \( \ell_{ij} \neq 0 \) iff node \( x_j \) is an ancestor of some node \( x_k \) in the elimination tree with \( a_{ik} \neq 0 \).

Second, if \( \ell_{ij} \neq 0 \) with \( i > j \), by the theorem on the left, the node \( x_j \) is an ancestor of some node \( x_k \) with \( a_{ik} \neq 0 \).

In other words, \( x_i \in \text{Adj}_G(\mathbf{A})(x_k) \), where \( x_k \in T[x_j] \). Equivalently, \( x_i \in \text{Adj}_G(\mathbf{A})(T[x_j]) \).
Determination of elimination trees

start with an empty forest (no vertices)
for $i = 1, \ldots, n$ do
  add vertex $i$ to the forest
  for $k < i$ and $a_{ik} \neq 0$ do
    make $i$ the parent of the root of the tree containing $k$

For efficiency use disjoint set operations to find the root of the tree containing $j$. This way a complexity of $O(m\alpha(m, n))$ can be achieved.

makeset$(x)$: create a new singleton set with element $x$.

find$(x)$: return the representative of the set containing $x$.

link$(x, y)$: form the union of the two sets containing $x$ and $y$, and return the representative of the new set.

Simple yet effective techniques (path compression and balancing) should be used for $O(m\alpha(m, n))$ time — necessitates keeping two structures, in practice, a little costlier version is used.
The elimination tree

Closing

Definition and properties

Determination

Determining row and column counts

Some other properties

Determination of elimination trees

for $i = 1, \ldots, n$ do

$b$

makeset($i$)

$parent[i] \leftarrow 0$

for $k < i$ and $a_{ik} \neq 0$ do

$u \leftarrow$ find($k$)

if $parent[u] = 0$ and $u \neq i$ then

$parent[u] \leftarrow i$

$link(i, u)$

\[ F = \begin{pmatrix}
  a & \bullet & \bullet & \bullet \\
  b & \bullet & \bullet & \bullet \\
  c & \bullet & \bullet & \bullet \\
  d & \bullet & \bullet & \bullet \\
  e & \bullet & \bullet & \bullet \\
  f & \bullet & \bullet & \bullet \\
  g & \bullet & \bullet & \bullet \\
  h & \bullet & \bullet & \bullet \\
  i & \bullet & \bullet & \bullet \\
  j & \bullet & \bullet & \bullet \\
\end{pmatrix} \]
for $i = 1, \ldots, n$ do
  makeset($i$)
  parent[$i$] ← 0
for $k < i$ and $a_{ik} \neq 0$ do
  $u$ ← find($k$)
  if parent[$u$] = 0 and $u \neq i$ then
    parent[$u$] ← $i$
    link($i$, $u$)
The elimination tree

Closing

Definition and properties

Determination

Determining row and column counts

Some other properties

---

Determination of elimination trees

for $i = 1, \ldots, n$ do

makeset($i$)

$parent[i] \leftarrow 0$

for $k < i$ and $a_{ik} \neq 0$ do

$u \leftarrow \text{find}(k)$

if $parent[u] = 0$ and $u \neq i$ then

$parent[u] \leftarrow i$

$\text{link}(i, u)$

---

$F = \begin{pmatrix}
\text{a} & \cdot & \cdot & \cdot & \cdot \\
\cdot & \text{b} & \cdot & \cdot & \cdot \\
\cdot & \cdot & \text{c} & \cdot & \cdot \\
\cdot & \cdot & \cdot & \text{d} & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot
\end{pmatrix}$

---

FIG. 2.1. An example of matrix structures.

3 10
8 4

G(A G(F) G(F^t) = T(A)

FIG. 2.2. Graph structures of the example in Fig. 2.1.
Determination of elimination trees

for $i = 1, \ldots, n$ do

  makeset($i$)
  parent[$i$] ← 0

  for $k < i$ and $a_{ik} \neq 0$ do
    $u$ ← find($k$)
    if parent[$u$] = 0 and $u \neq i$ then
      parent[$u$] ← $i$
      link($i$, $u$)

$F = \begin{pmatrix}
  a & \bullet & \bullet & \bullet \\
  b & \bullet & \bullet & \bullet \\
  c & \bullet & \bullet & \bullet & \bullet \\
  d & \bullet & \bullet & \bullet & \bullet & \bullet \\
  e & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
  f & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
  g & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
  h & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
  i & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
  j & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet
\end{pmatrix}$
The elimination tree
Closing
Definition and properties
Determination
Determining row and column counts
Some other properties

Determinaton of elimination trees

for $i = 1, \ldots, n$ do
makeset($i$)
parent[$i$] ← 0
for $k < i$ and $a_{ik} \neq 0$ do
$u \leftarrow \text{find}(k)$
if parent[$u$] = 0 and $u \neq i$ then
parent[$u$] ← $i$
link($i, u$)

$F = \begin{pmatrix}
  a & \circ & \circ & \circ \\
  b & c & \circ & \circ \\
  & d & e & \circ \\
  & & f & \circ \\
  & & & g \\
  & & & h \\
  & & & \circ \circ \circ \\
  & & & \circ \circ \circ \\
  & & & \circ \circ \\
  & & & \circ \\
\end{pmatrix}$
Determination of elimination trees

for $i = 1, \ldots, n$ do
  \textit{makeset}(i)
  \textit{parent}[i] \leftarrow 0
  \textbf{for} $k < i$ and $a_{ik} \neq 0$ \textbf{do}
    $u \leftarrow \textit{find}(k)$
    \textbf{if} $\textit{parent}[u] = 0$ and $u \neq i$ \textbf{then}
      $\textit{parent}[u] \leftarrow i$
      \textit{link}(i, u)

\begin{pmatrix}
  a & \bullet & \bullet & \bullet \\
  b & c & \bullet & \bullet \\
  \bullet & d & \bullet & \bullet \\
  \bullet & e & f & \bullet \\
  \bullet & g & \bullet & \bullet \\
  \bullet & \bullet & \bullet & \bullet \\
  \bullet & \bullet & \bullet & \bullet \\
  \bullet & \bullet & \bullet & \bullet \\
  \bullet & \bullet & \bullet & \bullet \\
  i & \bullet & \bullet & \bullet \\
  j & \bullet & \bullet & \bullet
\end{pmatrix}
The elimination tree

**Closing**

**Definition and properties**

**Determination**

**Determining row and column counts**

Some other properties

---

**Determination of elimination trees**

```plaintext
for i = 1, ..., n do
    makeset(i)
    parent[i] ← 0
    for k < i and a_{ik} ≠ 0 do
        u ← find(k)
        if parent[u] = 0 and u ≠ i then
            parent[u] ← i
            link(i, u)
```

---

**FIG. 2.1.** An example of matrix structures.

\[
F = \begin{pmatrix}
    a & \cdot & \cdot & \cdot \\
    b & \cdot & \cdot & \cdot \\
    c & \cdot & d & \cdot \\
    \cdot & \cdot & \cdot & e \\
    \cdot & \cdot & \cdot & \cdot \\
    \cdot & \cdot & \cdot & \cdot \\
    \cdot & \cdot & \cdot & f \\
    \cdot & \cdot & \cdot & g \\
    \cdot & \cdot & \cdot & h \\
    \cdot & \cdot & i & \cdot \\
    \cdot & \cdot & \cdot & j
\end{pmatrix}
\]

**FIG. 2.2.** Graph structures of the example in Fig. 2.1.
**Determination of elimination trees**

for \( i = 1, \ldots, n \) do

\( \text{makeset}(i) \)

\( \text{parent}[i] \leftarrow 0 \)

for \( k < i \) and \( a_{ik} \neq 0 \) do

\( u \leftarrow \text{find}(k) \)

if \( \text{parent}[u] = 0 \) and \( u \neq i \) then

\( \text{parent}[u] \leftarrow i \)

\( \text{link}(i, u) \)

\[
F = \begin{pmatrix}
  a & \cdot & \cdot & \cdot & \cdot \\
  \cdot & b & \cdot & \cdot & \cdot \\
  \cdot & \cdot & c & \cdot & \cdot \\
  \cdot & \cdot & \cdot & d & \cdot \\
  \cdot & \cdot & \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot & \cdot & \cdot \\
\end{pmatrix}
\]
The elimination tree

**Definition and properties**

**Determination**

**Determining row and column counts**

**Some other properties**

---

**Determination of elimination trees**

for $i = 1, \ldots, n$ do

- $\text{makeset}(i)$
- $\text{parent}[i] \leftarrow 0$

for $k < i$ and $a_{ik} \neq 0$ do

- $u \leftarrow \text{find}(k)$
- if $\text{parent}[u] = 0$ and $u \neq i$ then
  - $\text{parent}[u] \leftarrow i$
  - $\text{link}(i, u)$

---

**Figure 2.1.** An example of matrix structures.

**Figure 2.2.** Graph structures of the example in Fig. 2.1.
The elimination tree

Closing

Definition and properties

Determination

Determining row and column counts

Some other properties

The row and column counts: Slow version

Traverse each row subtree to count the nonzeros in rows and columns.

\[
cc(j) \leftarrow 1 \text{ for } j = 1, \ldots, n \\
\text{for } i = 1, \ldots, n \text{ do} \\
\quad rc(i) \leftarrow 1 \\
\quad marker[x_i] \leftarrow i \\
\quad \text{for } k < i \text{ and } a_{ik} \neq 0 \text{ do} \\
\quad \quad j \leftarrow k \triangleright \text{traverse and mark the nodes in row subtree } T_r[x_i] \\
\quad \quad \text{while } marker[x_j] \neq i \text{ do} \\
\quad \quad \quad rc(i) \leftarrow rc(i) + 1 \\
\quad \quad \quad cc(j) \leftarrow cc(j) + 1 \\
\quad \quad \quad marker[x_j] \leftarrow i \\
\quad \quad j \leftarrow \text{parent}(j)
\]

Runs in time proportional to the number of nonzeros in $L$
The elimination tree

The row and column counts

Gilbert, Ng, and Peyton, ’94 compute row and column counts in time proportional to $O(m \alpha(m, n))$, where $m = \text{nnz} (A)$ and $\alpha(m, n)$ is the inverse of the Ackermann’s function (which is $\leq 4$ for sensible $m$ and $n$).

The essential ideas are as follows (to highlight the machinery used):

Row counts:

- **postorder** the elimination tree, and count the number of edges in a row subtree without building it. We know the leaves, sort them according to the postorder.
- Find the level of the **least common ancestor** of two consecutively numbered nodes and subtract it from the first of those two nodes; difference in levels give the length of the path.
- Add the resulting path lengths. \(\langle\langle\text{On the board}\rangle\rangle\)

Column counts: \(\langle\langle\text{Self study}\rangle\rangle\)
Equivalent reorderings

**Definition**

Two orderings \( P \) and \( Q \) are equivalent if the structures of the filled graphs of \( PAP^T \) and \( QAQ^T \) are the same (that is they are isomorphic).

Equivalent orderings result in the same amount of fill-in and computation during factorization. To ease the notation, we discuss only one ordering wrt \( A \), i.e., \( P \) is an equivalent ordering of \( A \) if the filled graph of \( A \) and that of \( PAP^T \) are isomorphic.
Equivalent reorderings

Any topological ordering on $T(A)$ are equivalent

Let $P$ be the permutation matrix corresponding to a topological ordering of $T(A)$. Then, $G^+(PAP^T)$ and $G^+(A)$ are isomorphic.

Any topological ordering on $T(A)$ are equivalent

Let $P$ be the permutation matrix corresponding to a topological ordering of $T(A)$. The elimination tree $T(PAP^T)$ and $T(A)$ are isomorphic.

A quoi bon? We can process the children in any order (in a parallel implementation this would help).

There are some other equivalent orderings than the postordering of a given elimination tree. For some other aspects of factorization, one may prefer one ordering to some other. See papers by Liu on the web.
Outline

1 The elimination tree
   • Definition and properties
   • Determination
   • Determining row and column counts
   • Some other properties

2 Closing
Use Cholesky to solve $Ax = b$

1. Find a good ordering $P$, replace $A$ by $PAP^T$, and $b$ by $Pb$.
2. Build e-tree, do symbolic analysis to set up data structures (e.g., the row-structure)
   
   ```plaintext
   for column $j := 1$ to $n$ do
      begin
         $t_j := \begin{pmatrix} a_{jj} \\ \vdots \\ a_{nj} \end{pmatrix} - \sum_{k < j} l_{jk} \begin{pmatrix} l_{jk} \\ \vdots \\ l_{nk} \end{pmatrix}$
         
         $l_{jj} := \frac{1}{\sqrt{t_j}} \begin{pmatrix} t_j \\ \vdots \\ t_n \end{pmatrix}$
      end
   ```

3. Factorize $A = LL^T$.
4. Triangular solves $Ly = b$, $L^Tz = y$, return $P^Tz$ as $x$
Let \( L_{*i} \) and \( L_{*j} \) be two columns of \( L \) and suppose \( i > j \).

When these columns have the same pattern at all indices \( k > i \)?
What if \( i \) is the father of \( j \)?

Given an ordered matrix and its elimination tree, describe an algorithm to test if the given ordering is perfect.

Show: If \( A = LL^T \) and \( P \) is a topological ordering of the e-tree \( T \) of \( A \). Then, \( PLP^T \) is lower triangular and is the Cholesky factor of \( PAP^T \)