When performing LU (via LU-ki) we have a remaining reduced submatrix

\[ A^{(k)} = \begin{bmatrix} B^{(k)} \end{bmatrix} \]

"choosing kth pivot "

Choose entry from \( B^{(k)} \) that

(i) is not too small numerically

\[ |a_{ij}^{(k)}| \geq u \max_{l \leq k} |a_{lj}^{(k)}|, \quad i, j \geq k, \quad 0 < u \leq 1. \]

(Typical choice of \( u = 0.1 \))

(ii) minimizes \((r_i^{(k)} - 1)(c_j^{(k)} - 1)\)

Nonzero entries in row \( i \) of \( B^{(k)} \)
Nonzero entries in col \( j \) of \( B^{(k)} \)

Needs access to rows and columns.
"Greedy least / fill-in strategy ".

Assume matrix stored by columns.
Take column singletons first, must satisfy threshold
Take row singletons next, must check threshold.

Take cols before rows, easier to check stability.
Implementation

Need sorted lists of \( r_i \) and \( c_j \)

- For each \( r_i \), \( c_j \) keep a doubly linked list of rows (cols) with that count. Pointers to head of each list (8n storage)

- Sort pointers

- Easy to delete pivot row and column from their lists, because of double links.

- Any row or column which has had its # entries altered is removed and added to head of the correct list.

Terminate search

Working on col \( c_j \)

if i threshold acceptable and

\[
(c_i - 1)(c_j - 1) \leq (c_j - 1)^2
\]

Working on row \( r_i \)

if j threshold acceptable and

\[
(r_i - 1)(c_j - 1) \leq r_i (r_i - 1)
\]

Must not search all entries of \( B^{(k)} \).
Example

\[
A = \begin{bmatrix}
  x & x & x \\
  x & x & x \\
  x & x & x \\
\end{bmatrix}
\]

```
<table>
<thead>
<tr>
<th>Subscripts</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Row Counts</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next Row</td>
<td>0</td>
<td>5</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Prev Row</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Count Ptr</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sorted</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>According</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>To Row Count</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Col Counts</th>
<th>5</th>
<th>1</th>
<th>3</th>
<th>1</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next Col</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Prev Col</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Count Ptr</th>
<th>2</th>
<th>6</th>
<th>3</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pivot</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2, 2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pointer to List ofCols with 1 entry</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Col Counts</th>
<th>5</th>
<th>4</th>
<th>4</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next Col</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Prev Col</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Count Ptr</td>
<td>4</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

```
Now update col counts on removal of row 2. Use -
```
Before

rowPtr
1
2
3
4

colPtr
1
2
3
4
5

Pointers to rows with 1 nonzero, 2 nonzeros, etc.

After (2,2) pivot

rowPtr
1
2
3
4

colPtr
1
2
3
4
5

row 2 deleted.

col 1 deleted.

col 1 only has 4 entries left after row 2 deleted.