Chapter 28: Matrix Operations.

A curiosity: Multiplication of two complex numbers appears to take 4 real multiplications and 2 additions.

\[(a + bi)(c + di) = (ac - bd) + i(bc + ad)\]

But, the following rearrangement obtains the result with 3 multiplications and 5 additions.

\[
\begin{align*}
    a(c + d) &= ac + ad \\
    (b - a)c &= bc - ac \\
    b(c - d) &= bc - bd \\
    bc + ad &= ac - bd
\end{align*}
\]
Matrix multiplication: Given two \( n \times n \) matrices, \([a_{ij}]\) and \([b_{ij}]\), their product is \([c_{ij}]\), where

\[
c_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj}.
\]

MatrixMultiply(a, b) {
  for i = 1 to n
    for j = 1 to n
      for k = 1 to n
        c\[[i][j]\] = a\[[i][k]\] * b\[[k][j]\];
}

\[T(n) = (n + 1) + n(n + 1) + n^2(n + 1) + n^3 = 2n^3 + 2n^2 + 2n + 1 = \Theta(n^3).\]
Recursive approach: for $n = 2^k$, divide factors into four pieces.

\[
\begin{bmatrix}
C_{11} & C_{12} \\
C_{21} & C_{22}
\end{bmatrix} =
\begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{bmatrix} \cdot
\begin{bmatrix}
B_{11} & B_{12} \\
B_{21} & B_{22}
\end{bmatrix}
\]

\[
C_{11} = A_{11}B_{11} + A_{12}B_{21},
\]
\[
C_{12} = A_{11}B_{12} + A_{12}B_{22},
\]
\[
C_{21} = A_{21}B_{11} + A_{22}B_{21},
\]
\[
C_{22} = A_{21}B_{12} + A_{22}B_{22}.
\]

Observations:

1. Can isolate components by passing row and column boundaries to recursive calls.
2. Can merge results of recursive calls by doing $4(n/2)(n/2) = n^2$ additions.
3. $T(n) = 8T(n/2) + n^2$. Glue function is $n^2$; reference function is $n^{\log_2 8} = n^3$. Case (a) might apply.

\[
\frac{n^2}{n^3} = \frac{1}{n} \to 0,
\]

implying $n^2 = o(n^3)$, which in turn implies $n^2 = O(n^3)$, which enables case (a). Therefore $T(n) = \Theta(n^3)$. 
Further Observations:

1. Notice that addition is much less expensive than multiplication. Given two $n \times n$ matrices, $[a_{ij}]$ and $[b_{ij}]$, their sum is $[c_{ij}] = a_{ij} + b_{ij}$.

```java
MatrixAdd(a, b) {
    for i = 1 to n + 1
        for j = 1 to n + 1
            c[i][j] = a[i][j] + b[i][j];
}
```

$T(n) = (n + 1) + n(n + 1) + n^2 = 2n^2 + 2n + 1 = \Theta(n^2)$.

2. Very clever observation for case of $2 \times 2$ matrices: 8 multiplications, 4 additions.

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad B = \begin{bmatrix} e & f \\ g & h \end{bmatrix} \quad AB = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}$$

becomes 7 multiplication and (about) 18 additions/subtractions.

$$AB = \begin{bmatrix} -(a + b)h + d(g - e) + (a + d)(e + h) & a(f - h) + (a + b)h \\ +(b - d)(g + h) & a(f - h) - (c + d)e + (a + d)(e + h) \\ +(c + d)e + d(g - e) & -(a - c)((e + f) \end{bmatrix}$$

7 multiplications are:

$(a + b)h, d(g - e), (a + d)(e + h), (b - d)(g + h), a(f - h), (c + d)e, (a - c)(e + f)$
Strassen’s Method, for \( n = 2^k \).

Step 1: Divide the input matrix into four \((n/2) \times (n/2)\) matrices as above. \( \Theta(1) \), that is, constant time via index calculations.

Step 2: Compute 10 matrices \( S_1, \ldots, S_{10} \), each \((n/2) \times (n/2)\), in \( \Theta(n^2) \) time. These correspond to the factors in the 7 multiplications above:

\[
(a+b), (g-e), (a+d), (e+h), (b-d), (g+h), (f-h), (c+d), (a-c), (e+f)
\]

Step 3: Recursively compute 7 matrix products: \( P_1, \ldots, P_7 \), each of size \((n/2) \times (n/2)\). These correspond to the 7 multiplications themselves:

\[
(a+b)h, d(g-e), (a+d)(e+h), (b-d)(g+h), a(f-h), (c+d)e, (a-c)(e+f)
\]

Step 4: Create the components \( C_{11}, C_{12}, C_{21}, C_{22} \) by adding and subtracting various combinations of the matrix products in \( \Theta(n^2) \) time.

\[
T(n) = 7T(n/2) + \Theta(n^2)
\]

Reference function is \( n^{\log_2 7} = n^{2.8074} \). Glue function is \( f(n) = \Theta(n^2) \).

\[
\frac{f(n)}{n^{2.8074-\epsilon}} \leq \frac{K_2n^2}{n^{2.8074-\epsilon}} = \frac{K_2}{n^{0.8074-\epsilon}} \rightarrow 0, \text{ for } 0 < \epsilon < 0.8074.
\]

\( f(n) = o(n^{2.8074-\epsilon}) \), which implies \( f(n) = O(n^{2.8074-\epsilon}) \), which enables case(a).

\( T(n) = \Theta(n^{2.8074}) \).
3. For matrices which are not of size equal to a power of 2, pad with zeros to the next power of two. Desired product appears in the upper left corner of the larger product.

4. As of 2011, the record is $O(n^{2.375})$, held by Virginia Williams at Stanford — an improvement of 0.0033 in the exponent over the previous record.
Solving linear equations. $Ax = y.$

Cramer’s Rule:

$$x_1 = \frac{1}{|A|} \begin{vmatrix} y_1 & a_{1,2} & \cdots & a_{1,n} \\ y_2 & a_{2,1} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ y_n & a_{n,1} & \cdots & a_{n,n} \end{vmatrix}$$

Obtain determinants via minors. For example,

$$|A| = \sum_{j=1}^{n} a_{1,j}(-1)^{j+1}|A(1,j)|,$$

where $A(1,j)$ is the $(n - 1) \times (n - 1)$ matrix obtained from $A$ by deleting row 1 and column $j$.

Unfortunately, computing the determinant in this fashion is exponential:

$$T(n) = nT(n - 1) + \Theta(n)$$

$$T(n) \geq nT(n - 1) \geq n(n - 1)T(n - 2) \geq \ldots \geq n!T(1) \geq n!$$
Better approach: The lower-upper (LU) decomposition.

To solve: $Ax = y$.

Suppose we can express $A = LU$, where $L$ is lower-triangular (zeros above the main diagonal) and $U$ is upper-triangular (zeros below the main diagonal).

$$
\begin{align*}
\begin{array}{c}
A \\
L \\
\end{array}
\begin{array}{c}
U \\
\end{array}
= \\
\begin{array}{c}
x \\
y \\
\end{array}
\end{align*}
\begin{align*}
\begin{array}{cccc}
z_1 & L_{11}z_1 & & \\
z_2 & L_{21}z_1 + L_{22}z_2 & & \\
z_3 & L_{31}z_1 + L_{32}z_2 + L_{33}z_3 & & \\
& \vdots & \vdots & \\
z_n & L_{n,1}z_1 + L_{n,2}z_2 + \ldots + L_{n,n}z_n & & \\
\end{array}
\end{align*}
$$

Op count

\begin{align*}
(1) & \quad z_1 = y_1 / L_{11} \\
(3) & \quad z_2 = (y_2 - L_{21}z_1) / L_{22} \\
& \quad \vdots \\
(2n-1) & \quad z_n = (y_n - (L_{n,1}z_1 + L_{n,2}z_2 + \ldots + L_{n,n-1}z_{n-1})) / L_{n,n}
\end{align*}

$$
T(n) = \sum_{j=1}^{n} (2j - 1) = 2 \cdot \frac{n(n+1)}{2} - n = n^2.
$$
Now solve $Ux = z$.

\[
\begin{align*}
z_1 &= U_{11}x_1 + U_{12}x_2 + U_{13}x_3 + \ldots + U_{1n}x_n \\
z_2 &= U_{22}x_2 + U_{23}x_3 + \ldots + U_{2n}x_n \\
z_3 &= U_{33}x_3 + \ldots + U_{3n}x_n \\
&\vdots \quad \vdots \\
z_n &= U_{nn}x_n
\end{align*}
\]

Op count

\[
\begin{align*}
(1) & \quad x_n = z_n/U_{nn} \\
(3) & \quad x_{n-1} = (z_{n-1} - U_{n-1,n}x_n)/U_{n-1,n-1} \\
&\vdots \quad \vdots \\
(2n-1) & \quad z_1 = (z_1 - (U_{12}x_2 + U_{13}x_3 + \ldots + U_{1n}x_n))/U_{nn}
\end{align*}
\]

\[T(n) = \sum_{j=1}^{n} (2j - 1) = 2 \cdot \frac{n(n + 1)}{2} - n = n^2.\]

Observation: Solution of $Ax = y$ is now $\Theta(n^2)$ plus the work necessary to express $A = LU$. 

9
LU Decomposition of nonsingular $A$ — when the coefficients behave.

Suppose $A = [a_{ij}]$ and $a_{11} \neq 0$. Write

$$A = \begin{bmatrix} a_{11} & w^T \\ v & A' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ v/a_{11} & I_{n-1} \end{bmatrix} \cdot \begin{bmatrix} a_{11} & w^T \\ 0 & A' - vw^T/a_{11} \end{bmatrix}$$

1. Left factor is lower triangular.
2. Column one of right factor has zeros below the main diagonal.
3. Lower right expression in the right factor is an $(n-1) \times (n-1)$ matrix. That is, it is a subproblem for which an $LU$ decomposition might advance the right factor toward an upper-triangular state.

Note that $A$ nonsingular implies $\det(A) \neq 0$. Then,

$$0 \neq \det(A) = \det\left( \begin{bmatrix} 1 & 0 \\ v/a_{11} & I_{n-1} \end{bmatrix} \right) \cdot \det\left( \begin{bmatrix} a_{11} & w^T \\ 0 & A' - vw^T/a_{11} \end{bmatrix} \right)$$

$$= 1 \cdot a_{11} \cdot \det\left( A' - \frac{vw^T}{a_{11}} \right).$$

So, the Schur complement, $A' - vw^T/a_{11}$, is also non-singular. Continuing to assume that the upper left entry is not zero, we can write a recursive solution.
If \( n = 1 \), \( A = I_1 A = LU \).

If \( n = 2 \) and \( a_{11} \neq 0 \),

\[
A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{a_{21}}{a_{11}} & 1 \end{bmatrix} \cdot \begin{bmatrix} a_{11} & a_{12} \\ 0 & a_{22} - \frac{a_{21}a_{12}}{a_{11}} \end{bmatrix} = LU
\]

For the inductive step, suppose we have \( A' - vw^T/a_{11} = L'U' \).

\[
A = \begin{bmatrix} a_{11} & w^T \\ v & A' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ v/a_{11} & I_{n-1} \end{bmatrix} \cdot \begin{bmatrix} a_{11} & w^T \\ 0 & A' - vw^T/a_{11} \end{bmatrix} \\
= \begin{bmatrix} 1 & 0 \\ v/a_{11} & I_{n-1} \end{bmatrix} \cdot \begin{bmatrix} a_{11} & w^T \\ 0 & L'U' \end{bmatrix} = \begin{bmatrix} W & X \\ Y & Z \end{bmatrix},
\]

where

\[
W = 1 \cdot a_{11} + 0 \cdot 0 = a_{11} \\
X = 1 \cdot w^T + 0 \cdot L'U' = w^T \\
Y = \frac{v}{a_{11}} \cdot a_{11} + I_{n-1} \cdot 0 = v \\
Z = \frac{vw^T}{a_{11}} + I_{n-1}L'U' = \frac{vw^T}{a_{11}} + L'U'
\]

But

\[
\begin{bmatrix} 1 & 0 \\ v/a_{11} & L' \end{bmatrix} \cdot \begin{bmatrix} a_{11} & w^T \\ 0 & U' \end{bmatrix} = \begin{bmatrix} W & X \\ Y & Z \end{bmatrix}
\]

Hence

\[
A = \begin{bmatrix} 1 & 0 \\ v/a_{11} & L' \end{bmatrix} \cdot \begin{bmatrix} a_{11} & w^T \\ 0 & U' \end{bmatrix} = L \cdot U.
\]
Example from text:

\[
\begin{bmatrix}
2 & 3 & 1 & 5 \\
6 & 13 & 5 & 19 \\
2 & 19 & 10 & 23 \\
4 & 10 & 11 & 31
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
2 & 3 & 1 & 5 \\
3 & 4 & 2 & 4 \\
1 & 16 & 9 & 18 \\
2 & 4 & 9 & 21
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
4 & 2 & 4 & 4 \\
16 & 9 & 18 & 4 \\
4 & 9 & 21 & 1 \\
1 & 7 & 17 & 2
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
4 & 2 & 4 & 4 \\
1 & 7 & 17 & 2
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
1 & 2 & 1 & 2 \\
7 & 17 & 3
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
2 & 3 & 1 & 5 \\
6 & 13 & 5 & 19 \\
2 & 19 & 10 & 23 \\
4 & 10 & 11 & 31
\end{bmatrix} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
3 & 1 & 0 & 0 \\
1 & 4 & 1 & 0 \\
2 & 1 & 7 & 1
\end{bmatrix} \cdot \begin{bmatrix}
2 & 3 & 1 & 5 \\
4 & 2 & 4 & 4 \\
0 & 0 & 1 & 2 \\
0 & 0 & 0 & 3
\end{bmatrix} = LU
\]
LU-Decomp($A$) {
    $n = A$.rowCount;
    for $k = 1$ to $n - 1$
        for $i = k + 1$ to $n$
            $a_{ik} = a_{ik}/a_{kk}$;
        for $j = k + 1$ to $n$
            $a_{ij} = a_{ij} - a_{ik}a_{kj}$;
    }

Upper bound on $T(n)$: Run all loops from 1 to $n$.

$$T(n) \leq 1 + 1 + \sum_{k=1}^{n} \left(1 + \sum_{i=1}^{n} \left[1 + \sum_{j=1}^{n} (2)\right]\right) = 2 + n + \sum_{k=1}^{n} \sum_{i=1}^{n} (1 + 2n)$$
$$= 2 + n + n^2 + 2n^3 = O(n^3)$$

Lower bound: ignore everything except last assignment.

$$T(n) \geq \sum_{k=1}^{n-1} \sum_{i=k+1}^{n} \sum_{j=k+1}^{n} (1) = \sum_{k=1}^{n-1} \sum_{i=k+1}^{n} [n - (k + 1) + 1] = \sum_{k=1}^{n-1} (n - k) \sum_{i=k+1}^{n} (1)$$
$$= \sum_{k=1}^{n-1} (n - k)^2 = \sum_{k=1}^{n-1} k^2 = \frac{(n - 1)(n)[2(n - 1) + 1]}{6} = \frac{n(n - 1)(2n - 1)}{6}$$
$$= \frac{n^3}{3} - \frac{n^2}{2} + \frac{n}{6} = \Omega(n^3).$$

We conclude $T(n) = \Theta(n^3)$. But, algorithm depends on the upper left corner of the matrix at each level being nonzero.
If we can extend this algorithm to an $\Theta(n^3)$-time algorithm for a general nonsingular $A$, we can solve linear equations in

$$T'(n) = \Theta(n^3) + 2 \cdot \Theta(n^2) = \Theta(n^3) \text{ time.}$$

A permutation matrix $P = [p_{ij}]$ has these properties.

1. Entries are zero, except for a single one in each row, arranged such there is also a single one in each column.

2. Determinant is $\pm 1$. If $C$ is the collection of all permutations of $\{1, 2, \ldots, n\}$, then

$$\det(P) = \sum_{\pi \in C} \text{sign}(\pi) \prod_{i=1}^{n} p_{i, \pi(i)}$$

sign$(\pi)$ is plus one for even permutations (even number of inversions) and minus one for odd permutations. All of the products are zero, except the one associated with the unique $\pi \in C$ for which $\pi(i)$ is the column in which the single one of row $i$ resides. Hence $\det(P) = \pm 1$.

3. $PA$ is $A$ with some of its rows permuted.

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} = \begin{bmatrix} 9 & 10 & 11 & 12 \\ 1 & 2 & 3 & 4 \\ 13 & 14 & 15 & 16 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

4. $AP$ is $A$ with some of its columns permuted.

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 4 & 1 & 3 \\ 6 & 8 & 5 & 7 \\ 10 & 12 & 9 & 11 \\ 14 & 16 & 13 & 15 \end{bmatrix}$$

5. The product of two permutation matrices is a permutation matrix.

6. We can solve $Ax = y$ by solving $PAx = Py$. The latter simply rewrites the equations in a different order.
7. We now seek $P, L, U$ such that $PA = LU$.

Given nonsingular $A$, let $QA$ be the row permutation that brings the largest absolute value in column one to the upper left column. Then

$$\det(QA) = \det(Q) \cdot \det(A) = \pm \det(A) \neq 0.$$ 

Then

$$QA = \begin{bmatrix} a_{11} & w^T \\ v & A' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ v/a_{11} & I_{n-1} \end{bmatrix} \cdot \begin{bmatrix} a_{11} & w^T \\ 0 & A' - vw^T/a_{11} \end{bmatrix}$$

By induction we have $P', L', U'$ such that

$$P' \left( A' - \frac{vw^T}{a_{11}} \right) = L'U'.$$

Let $P$ be the permutation matrix

$$P = \begin{bmatrix} 1 & 0 \\ 0 & P' \end{bmatrix} \cdot Q$$

Now,

$$PA = \begin{bmatrix} 1 & 0 \\ 0 & P' \end{bmatrix} \cdot QA = \begin{bmatrix} 1 & 0 \\ 0 & P' \end{bmatrix} \begin{bmatrix} 1 & 0 \\ v/a_{11} & I_{n-1} \end{bmatrix} \cdot \begin{bmatrix} a_{11} & w^T \\ 0 & A' - vw^T/a_{11} \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 \\ P'v/a_{11} & P' \end{bmatrix} \cdot \begin{bmatrix} a_{11} & w^T \\ 0 & (A' - vw^T/a_{11}) \end{bmatrix}$$

↑ want $P'$ here to apply induction
\[
PA = \begin{bmatrix} 1 & 0 \\ 0 & P' \end{bmatrix} \cdot QA = \begin{bmatrix} 1 & 0 \\ 0 & P' \end{bmatrix} \begin{bmatrix} 1 & 0 \\ v/a_{11} & I_{n-1} \end{bmatrix} \cdot \begin{bmatrix} a_{11} & w^T \\ 0 & A' - vw^T/a_{11} \end{bmatrix} \\
= \begin{bmatrix} 1 & 0 \\ P'v/a_{11} & P' \end{bmatrix} \cdot \begin{bmatrix} a_{11} & w^T \\ 0 & (A' - vw^T/a_{11}) \end{bmatrix} \\
\uparrow \text{want } P' \text{ here to apply induction} 
\]

Look at four components of

\[
PA = \begin{bmatrix} A & B \\ C & D \end{bmatrix} : \\
A = 1 \cdot a_{11} + 0 \cdot 0 \\
B = 1 \cdot w^T + 0 \cdot \left(A' - \frac{vw^T}{a_{11}}\right) \\
C = \frac{P'v}{a_{11}} \cdot a_{11} + P' \cdot 0 \\
D = \frac{P'vw^T}{a_{11}} + P' \cdot \left(A' - \frac{vw^T}{a_{11}}\right) 
\]

These four product components are unchanged if we replace \((A' - vw^T/a_{11})\) with \(P' \cdot (A' - vw^T/a_{11})\) in the second matrix, provided we also replace \(P'\) in the first matrix with \(I_{n-1}\).

That is,

\[
PA = \begin{bmatrix} 1 & 0 \\ P'v/a_{11} & I_{n-1} \end{bmatrix} \cdot \begin{bmatrix} a_{11} & w^T \\ 0 & P'\left(A' - vw^T/a_{11}\right) \end{bmatrix} \\
= \begin{bmatrix} 1 & 0 \\ P'v/a_{11} & I_{n-1} \end{bmatrix} \cdot \begin{bmatrix} a_{11} & w^T \\ 0 & L'U' \end{bmatrix} \\
\uparrow \text{want } L' \text{ here for lower triangularity} 
\]
\[ PA = \begin{bmatrix} 1 & 0 \\ P'v/a_{11} & I_{n-1} \end{bmatrix} \cdot \begin{bmatrix} a_{11} & w^T \\ 0 & P'(A' - vw^T/a_{11}) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ P'v/a_{11} & I_{n-1} \end{bmatrix} \cdot \begin{bmatrix} a_{11} & w^T \\ 0 & L'U' \end{bmatrix} \]

\[ \uparrow \text{want } L' \text{ here for lower triangularity} \]

Again,

\[ PA = \begin{bmatrix} A & B \\ C & D \end{bmatrix} : \begin{align*}
A &= 1 \cdot a_{11} + 0 \cdot 0 \\
B &= 1 \cdot w^T + 0 \cdot L'U' \\
C &= \frac{P'v}{a_{11}} \cdot a_{11} + I_{n-1} \cdot 0 \\
D &= \frac{P'vw^T}{a_{11}} + I_{n-1} \cdot L'U' 
\end{align*} \]

These four product components are unchange if we replace \( I_{n-1} \) with \( L' \) in the first matrix and \( L'U' \) with simply \( U' \) in the second. That is,

\[ PA = \begin{bmatrix} 1 & 0 \\ P'v/a_{11} & L' \end{bmatrix} \cdot \begin{bmatrix} a_{11} & w^T \\ 0 & U' \end{bmatrix} = LU. \]
Example. We track the permutation matrix with a column vector \( P \), in which \( P[i] \) records the original location of row \( i \).

\[
\begin{bmatrix}
1 & 2 & 0 & 2 \cdot 0.6 \\
2 & 3 & 3 & 4 \cdot -2 \\
3 & 5 & 5 & 4 \\
4 & -1 & -2 & 3.4 \cdot -1
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
3 & 5 & 4 & 2 \\
2 & 3 & 3 & 4 \cdot -2 \\
1 & 2 & 0 & 2 \cdot 0.6 \\
4 & -1 & -2 & 3.4 \cdot -1
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
2 & 0.6 & 0 & 1.6 \cdot -3.2 \\
1 & 0.4 & -2 & 0.4 \cdot -0.2 \\
4 & -0.2 & -1 & 4.2 \cdot -0.6 \\
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
1 & 0.4 & -2 & 0.4 \cdot -0.2 \\
2 & 0.6 & 0 & 1.6 \cdot -3.2 \\
4 & -0.2 & 0.5 & 4 \cdot -0.5 \\
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
2 & 0.6 & 0 & 1.6 \cdot -3.2 \\
4 & -0.2 & 0.5 & 4 \cdot -0.5 \\
2 & 0.6 & 0 & 0.4 \cdot -3
\end{bmatrix}
\]

Recombining,

\[
\begin{bmatrix}
1 & 2 & 0 & 2 \cdot 0.6 \\
2 & 3 & 3 & 4 \cdot -2 \\
3 & 5 & 5 & 4 \\
4 & -1 & -2 & 3.4 \cdot -1
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
3 & 5 & 4 & 2 \\
1 & 0.4 & -2 & 0.4 \cdot -0.2 \\
4 & -0.2 & 0.5 & 4 \cdot -0.5 \\
2 & 0.6 & 0 & 0.4 \cdot -3
\end{bmatrix}
\]

\[
P = \begin{bmatrix}
0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0
\end{bmatrix}
L = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0.4 & 1 & 0 & 0 \\
-0.2 & 0.5 & 1 & 0 \\
0.6 & 0 & 0.4 & 1
\end{bmatrix}
U = \begin{bmatrix}
5 & 5 & 4 & 2 \\
0 & -2 & 0.4 & -0.2 \\
0 & 0 & 4 & -0.5 \\
0 & 0 & 0 & -3
\end{bmatrix}
\]

Check \( PA = LU \) ....
LUP-Decomp($A$) {
(a) $n = A$.length;
(b) for $i = 1$ to $n$
  $\pi(i) = i$;

for $k = 1$ to $n - 1$ {
(c) $p = 0$;
  for $i = k$ to $n$
    if $|a_{ik}| > p$
      $p = |a_{ik}|$;
      $k' = i$;
  
(d) if $p = 0$
    exit(“Singular Matrix”);

(e) swap $\pi[k]$ and $\pi[k']$;
(f) for $i = 1$ to $n$
    swap $a_{k,i}$ and $a_{k',i}$;

(g) for $i = k + 1$ to $n$
    $a_{ik} = a_{ik}/a_{kk}$;
    for $j = k + 1$ to $n$
      $a_{ij} = a_{ij} - a_{ik} \cdot a_{kj}$;

(h) return ($\pi$, $A$);
}

(a) $O(1)$
(b) $O(n)$
(c) $O(n^2)$
(d) $O(n)$ — inside outer $k$-loop
(e) $O(n)$
(f) $O(n^2)$
(g) $O(n^3)$
(h) $O(1)$

$T(n) = O(n^3)$
$T(n) \geq (g)$ activity

$$
= \sum_{k=1}^{n-1} \sum_{i=k+1}^{n} \sum_{j=k+1}^{n} (1)
= \sum_{k=1}^{n-1} \sum_{i=k+1}^{n} (n - k)
= \sum_{k=1}^{n-1} (n - k) \sum_{i=k+1}^{n} (1) = \sum_{k=1}^{n-1} (n - k)^2
= \sum_{k=1}^{n-1} k^2 = \frac{(n - 1)(n)(2n - 1)}{6} = \Omega(n^3).
$$

$T(n) = \Theta(n^3)$. 
Matrix Inversion in $\Theta(n^3)$ time.

For nonsingular $A$, we want to solve $AB = I$ for $B$. Consider the columns of $B$ individually:

$$
\begin{align*}
    b_{*j} &= \begin{bmatrix} b_{1,j} \\ b_{2,j} \\ \vdots \\ b_{n,j} \end{bmatrix} \\
    B &= \begin{bmatrix} b_{*1} & b_{*2} & \cdots & b_{*n} \end{bmatrix}
\end{align*}
$$

Now, we want to solve $A \begin{bmatrix} b_{*1} & b_{*2} & \cdots & b_{*n} \end{bmatrix} = \begin{bmatrix} e_{*1} & e_{*2} & \cdots & e_{*n} \end{bmatrix}$, where $e_{*j}$ is a column vector having a one in position $j$ and zeros elsewhere.

In $\Theta(n^3)$ time, we compute $PA = LU$.

For each $j$, we solve $Ab_{*j} = e_j$ in $\Theta(n^2)$ time. Therefore in an additional $n^3$ time we have $B$ such that $AB = I$. 
At this point, we have

\[
AB = I
\]
\[
(AB)A = A
\]
\[
A(BA) = A.
\]

Letting \( a_{*j} \) and \( x_{*j} \) denote the \( j \)th column of \( A \) and \( BA \) respectively, we then have

\[
Ax_{*j} = a_{*j}
\]

Since \( A \) is nonsingular, this equation has a unique solution for \( x_{*j} \). As \( x_{*j} = e_{*j} \) is an obvious solution, we conclude that \( x_{*j} = e_{*j} \). That is, \( BA = I \). Then \( AB = I = BA \) implies \( B = A^{-1} \), computed in \( \Theta(n^3) \) time.
Theorem 28.1 (Matrix multiplication is no harder than matrix inversion).

If we can invert an $n \times n$ matrix in time $I(n)$, where $I(n) = \Omega(n^2)$ and $I(n)$ satisfies the regularity condition $I(3n) = O(I(n))$, then we can multiply two $n \times n$ matrices in time $O(I(n))$.

Pre-proof comments.

1. The inversion constraint, $I(n) = \Omega(n^2)$, is not a serious constraint because any inversion algorithm must fill the $n^2$ entries in the inverse, which in itself requires $\Omega(n^2)$ time.

2. We know we can invert in $\Theta(n^3)$ time. Then

$$
\frac{K_1 n^3}{K_2 n^3} \leq \frac{I(n)}{I(n)} \leq \frac{K_1 (3n)^3}{K_2 (3n)^3}
$$

$$
27 \frac{K_1}{K_2} \leq \frac{I(3n)}{I(n)} \leq 27 \frac{K_2}{K_1}
$$

$$
I(3n) = O(I(n)).
$$

That is the regularity condition holds for our $\Theta(n^3)$ inversion method. A similar calculation shows that it will also hold for any better inversion time $I(n) = \Theta(n^c \lg^d n)$ for $c > 0$ and $d \geq 0$. 
Proof: We summarize the constraints as \( I(n) \geq C_1 n^2 \) and \( I(3n) \leq C_2 I(n) \). Let \( A, B \) be two \( n \times n \) matrices for which we want the product \( C = AB \). Define the \( 3n \times 3n \) matrix \( D \) by

\[
D = \begin{bmatrix} I_n & A & 0 \\ 0 & I_n & B \\ 0 & 0 & I_n \end{bmatrix},
\]

where \( I_n \) is the \( n \times n \) identity matrix. Note that

\[
D \cdot \begin{bmatrix} I_n & -A & AB \\ 0 & I_n & -B \\ 0 & 0 & I_n \end{bmatrix} = \begin{bmatrix} I_n & A & 0 \\ 0 & I_n & B \\ 0 & 0 & I_n \end{bmatrix} \cdot \begin{bmatrix} I_n & -A & AB \\ 0 & I_n & -B \\ 0 & 0 & I_n \end{bmatrix} = \begin{bmatrix} I_n & 0 & 0 \\ 0 & I_n & 0 \\ 0 & 0 & I_n \end{bmatrix}.
\]

That is,

\[
D^{-1} = \begin{bmatrix} I_n & -A & AB \\ 0 & I_n & -B \\ 0 & 0 & I_n \end{bmatrix}.
\]

It follows that we can multiply \( AB \) by inverting \( D \) and extracting the upper right \( n \times n \) corner.

We can construct \( D \) in \( O(n^2) \) time by filling the \( 3n \times 3n \) components from the data \( (A \text{ and } B) \) and from constants 0 and 1. The inversion then takes \( I(3n) \leq C_2 I(n) \) time. The total for the multiplication operation is then

\[
T(n) \leq Kn^2 + C_2 I(n) = \frac{K \cdot I(n)}{C_1} + C_2 I(n) = \left( \frac{K}{C_1} + C_2 \right) \cdot I(n) = O(I(n)).
\]
Also, Theorem 28.2 (Inversion is no harder than multiplication).

For this result, we need to investigate positive definite matrices.

Definitions.

1. For \( n \times n \) matrix \( A \) and column vector \( x \), the expression \( x^T Ax \) is a **quadratic form**.

2. \( A \) has **full column rank** if all columns of \( A \) are independent.

3. \( A \) is **positive definite** if \( x^T Ax > 0 \) for all non-zero \( x \).
Theorem: \( A \) has full column rank implies \( A^T A \) is positive definite.

Proof. Let the columns of \( A \) be \( a_{*1}, a_{*2}, \ldots a_{*n} \). Let \( x_1, x_2, \ldots, x_n \) be the elements of column vector \( x \). Then

\[
Ax = \begin{bmatrix} a_{*1} & a_{*2} & \ldots & a_{*n} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \sum_{j=1}^{n} x_j \cdot a_{*j}.
\]

So, \( Ax = 0 \) implies \( x = 0 \). Then, for nonzero \( x \),

\[
x^T (A^T A)x = (x^T A^T)(Ax) = (Ax)^T (Ax) = \sum_{j=1}^{n} (Ax)_j^2 > 0,
\]

unless \( (Ax)_i = 0 \) for all \( 0 \leq i \leq n \). But, in that case, \( Ax = 0 \) and therefore \( x = 0 \) — a contradiction. We conclude that \( A^T A \) is positive definite. □

Lemma 28.3 A positive definite matrix is nonsingular.

Proof by contradiction. Suppose positive definite $A$ is singular. Then there exists $x \neq 0$ such that $Ax = 0$, which implies $x^T Ax = 0$ contradicting the positive definite property of $A$. ■
Goal: To show that the LUP decomposition of a symmetric positive definite matrix can omit the pivot step.

Definition: A **leading submatrix** of a square matrix is a square upper-left corner.
Lemma 28.4: Every leading submatrix of a symmetric positive definite matrix is symmetric positive definite.

Proof. Let $A_k$ denote the upper left $k \times k$ corner. We note that $A_k$ is symmetric. Now, for purposes of deriving a contradiction, suppose $A_k$ is not positive definite. Then there exists a $k$-vector $x_k$ such that $x_k \neq 0$ and $x_k^T A_k x_k \leq 0$. Define $n$-vector $x$ as

$$x = \begin{bmatrix} x_k \\ 0 \end{bmatrix} \neq 0.$$

Then

$$x^T A x = \begin{bmatrix} x_k^T \\ 0 \end{bmatrix} \begin{bmatrix} A_k & B^T \\ B & C \end{bmatrix} \begin{bmatrix} x_k \\ 0 \end{bmatrix} = \begin{bmatrix} (x_k^T A_k + 0 \cdot B) & (x_k^T B^T + 0 \cdot C) \end{bmatrix} \begin{bmatrix} x_k \\ 0 \end{bmatrix} = x_k^T A_k x_k + x_k^T B^T \cdot 0 = x_k^T A_k x_k \leq 0,$$

contradicting the positive definite nature of $A$. □
Definitions.

1. Recall the Schur complement of nonsingular $A$ with respect to $a_{11}$:

   \[
   A = \begin{bmatrix}
   a_{11} & w^T \\
   v & A'
   \end{bmatrix}
   \quad \text{Schur}(A) = A' - \frac{vw^T}{a_{11}}.
   \]

2. The Schur complement of symmetric positive definite $A$ with respect to leading submatrix $A_k$:

   \[
   A = \begin{bmatrix}
   A_k & B^T \\
   B & C
   \end{bmatrix}
   \quad \text{Schur}(A) = C - BA_k^{-1}B^T.
   \]

3. Note that $A_k^{-1}$ exists since $A$ symmetric positive definite implies $A_k$ symmetric positive definite, which, in turn, implies $A_k$ is nonsingular.
Lemma 28.5 (Schur complement lemma) If $A$ is a symmetric positive definite matrix and $A_k$ is a leading $k \times k$ submatrix of $A$, then the Schur complement with respect to $A_k$ is symmetric positive definite.

Proof. A symmetric implies both $A_k$ and $C$ symmetric. Let $e_i$ denote the column vector with a one in position $i$ and zeros elsewhere.

\[
(A_k^{-1})_{ij} = e_i^T A_k^{-1} e_j = (e_i A_k^{-1} e_j)^T, \quad \text{since scalars are symmetric}
\]
\[
= e_j^T (A_k^{-1})^T e_i = e_j^T (A_k^T)^{-1} e_i = e_j^T (A_k)^{-1} e_i, \quad \text{since $A_k$ is symmetric}
\]
\[
= (A_k^{-1})_{ji}.
\]

We conclude that $A_k^{-1}$ is symmetric. Note that we have used $(M^{-1})^T = (M^T)^{-1}$, which follows from

\[
(M^{-1})^T \cdot M^T = (M \cdot M^{-1})^T = I^T = I
\]
\[
M^T \cdot (M^{-1})^T = (M^{-1} \cdot M)^T = I^T = I.
\]

Also,

\[
(BA_k^{-1}B^T)_{ij} = \sum_{p=1}^{k} \sum_{q=1}^{k} B_{ip}(A_k^{-1})_{pq}B_{qj}^T = \sum_{p=1}^{k} \sum_{q=1}^{k} B_{jq}(A_k^{-1})_{qp}B_{pi}^T = (BA_k^{-1}B^T)_{ji}.
\]

We have $C$ and $(BA_k^{-1}B^T)$ both symmetric, which implies the Schur complement $C - BA_k^{-1}B^T$ is symmetric. It remains to show $C - BA_k^{-1}B^T$ is positive definite.

Now, for any nonzero $(n - k)$-vector $z$, write

\[
x = \begin{bmatrix} -A_k^{-1}B^Tz \\ z \end{bmatrix} \neq 0.
\]
\[ 0 < x^T Ax = \begin{bmatrix} -z^T B (A_k^T)^{-1} & z^T \end{bmatrix} \cdot A \cdot \begin{bmatrix} -A_k^{-1}B^T z \\ z \end{bmatrix} \]

\[ = \begin{bmatrix} -z^T B A_k^{-1} & z^T \end{bmatrix} \begin{bmatrix} A_k & B & C \\ B & C & \end{bmatrix} \begin{bmatrix} -A_k^{-1}B^T z \\ z \end{bmatrix}, \quad \text{since } A_k \text{ is symmetric} \]

\[ = \begin{bmatrix} (-z^T B + z^T B) \quad (-z^T B A_k^{-1}B^T + z^T C) \end{bmatrix} \begin{bmatrix} -A_k^{-1}B^T z \\ z \end{bmatrix} \]

\[ = \begin{bmatrix} (0) \quad (-z^T B A_k^{-1}B^T + z^T C) \end{bmatrix} \begin{bmatrix} -A_k^{-1}B^T z \\ z \end{bmatrix} \]

\[ = -z^T B A_k^{-1}B^T z + z^T C z = z^T (C - BA_k^{-1}B^T)z \]

We conclude that \( C - BA_k^{-1}B^T \) is positive definite. \( \blacksquare \)
We return to

Theorem 28.2 (Inversion is no harder than multiplication)

Suppose we can multiply two \(n \times n\) matrices in time \(M(n)\), where

1. \(M(n) = \Omega(n^2)\),
2. \(M(n + k) = O(M(n))\) for any \(k \in \{0, 1, 2, \ldots, n\}\), and
3. \(M(n/2) \leq cM(n)\) for some constant \(c < 1/2\).

Then we can compute the inverse of any nonsingular matrix in time \(O(M(n))\).
Proof. We can assume \( n = 2^k \) since

\[
\begin{bmatrix}
A & 0 \\
0 & I_k
\end{bmatrix}^{-1} = \begin{bmatrix}
A^{-1} & 0 \\
0 & I_k
\end{bmatrix}.
\]

It requires \( \Theta(n^2) \) time to enlarge a matrix to a size equal to the next largest power of 2, because we add less than \( 3n^2 \) zeros and ones. The enlarged size is \((2^k \times 2^k) = (n + k) \times (n + k)\) for some \( k \in \{0, 1, \ldots, n\} \). If the theorem holds for \( n = 2^k \), we can then invert this larger matrix in time

\[
T(n) = O(M(n + k)) + O(n^2), \quad \text{the second term accounts for the padding}
\]
\[
= O(M(n)) + O(n^2), \quad \text{by the second condition in the hypothesis}
\]
\[
\leq O(M(n)) + K \cdot M(n), \quad \text{by the first condition in the hypothesis}
\]
\[
= O(M(n))
\]
We first consider the subcase where, in addition to \( n = 2^k \), we also have \( A \) symmetric positive definite. We write

\[
A = \begin{bmatrix} B & C^T \\ C & D \end{bmatrix},
\]

where \( A, B, C, D \) are of size \((n/2) \times (n/2)\) and \( n/2 \) is also a power of 2. Let \( S = D - CB^{-1}C^T \) be the Schur complement with respect to \( A \).

Deus ex machina, we have

\[
A^{-1} = \begin{bmatrix} B & C^T \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} B^{-1} + B^{-1}C^T S^{-1} CB^{-1} & -B^{-1}C^T S^{-1} \\ -S^{-1} CB^{-1} & S^{-1} \end{bmatrix}.
\]

Confirming by direct multiplication, the product is

\[
AA^{-1} = \begin{bmatrix} I + C^T S^{-1} CB^{-1} - C^T S^{-1} CB^{-1} & -C^T S^{-1} + C^T S^{-1} \\ CB^{-1} + CB^{-1} C^T S^{-1} CB^{-1} - DS^{-1} CB^{-1} & -CB^{-1} C^T S^{-1} + DS^{-1} \end{bmatrix}
\]

\[
= \begin{bmatrix} I & 0 \\ CB^{-1} + [CB^{-1} C^T - D] S^{-1} CB^{-1} & -[CB^{-1} C^T - D] S^{-1} \end{bmatrix}
\]

\[
= \begin{bmatrix} I & 0 \\ CB^{-1} - SS^{-1} CB^{-1} & SS^{-1} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}.
\]

Note that \( B^{-1} \) and \( S^{-1} \) both exist since \( B \) and \( S \) are both positive definite and therefore nonsingular.
Let $I(n)$ denote the time complexity of this approach. We break down the operations as follows.

1. Obtain $B^{-1}$ by recursion: $I(n/2)$
2. Multiply $C \cdot B^{-1}$: $M(n/2)$
3. Multiply $(CB^{-1}) \cdot C^T$: $M(n/2)$
4. Assemble $S = D - CB^{-1}C^T$: $\Theta(n^2)$
5. Obtain $S^{-1}$ by recursion: $I(n/2)$
6. Multiply $S^{-1} \cdot (CB^{-1})$: $M(n/2)$
7. Multiply $(CB^{-1})^T \cdot (S^{-1}CB^{-1})$: $M(n/2)$
8. Assemble $A^{-1}$: All of the following are $\Theta(n^2)$ operations as they involve only matrix sums.

(a) From (1) and (7)

\[
(A^{-1})_{11} = B^{-1} + [(CB^{-1})^T(S^{-1}CB^{-1})]^T = B^{-1} + [(B^{-1})^T C^T S^{-1} CB^{-1}]^T \\
= B^{-1} + (B^{-1})^T C^T (S^{-1})^T CB^{-1} = B^{-1} + B^{-1} C^T S^{-1} CB^{-1}
\]

The last equality holds because $B$ and $S$ are symmetric. Therefore

\[
(B^{-1})^T = (B^T)^{-1} = B^{-1} \\
(S^{-1})^T = (S^T)^{-1} = S^{-1}.
\]

(b) From (6)

\[
(A^{-1})_{12} = -[S^{-1}CB^{-1}]^T = -B^{-1}C^T S^{-1}
\]

(c) From (6)

\[
(A^{-1})_{21} = -S^{-1}CB^{-1}
\]

(d) From (5)

\[
(A^{-1})_{22} = S^{-1}
\]

Therefore $I(n) = 2I(n/2) + 4M(n/2) + \Theta(n^2)$. 

[35]
\begin{align*}
I(n) &= 2I(n/2) + 4M(n/2) + \Theta(n^2) \\
&\leq 2I(n/2) + 2M(n) + \Theta(n^2), \quad \text{since } M(n/2) \leq cM(n) \text{ for some } c < 1/2 \\
&\leq 2I(n/2) + O(M(n)), \quad \text{since } M(n) = \Omega(n^2) \\
&\leq 2I(n/2) + K \cdot M(n).
\end{align*}

The reference function for this recursion is \( g(n) = n^{\log_2 2} = n \). The glue function is \( f(n) = KM(n) \). Now, \( M(n) = \Omega(n^2) \) forces

\[
\frac{f(n)}{g_+(n)} = \frac{KM(n)}{n^{1+\epsilon}} \geq \frac{KK'n^2}{n^{1+\epsilon}} \to \infty,
\]

for small positive \( \epsilon \). That is, \( f = \Omega(g_+) \) and case (c) of the master template may apply. We need \( 0 < \hat{c} < 1 \) with \( 2f(n/2) \leq \hat{c}f(n) \) for all sufficiently large \( n \). As we have the regularity condition \( M(n/2) \leq cM(n) \) for some \( c < 1/2 \), we then have

\[
2f(n/2) = 2KM(n/2) \leq (2c)KM(n) = \hat{c}KM(n) = \hat{c}f(n),
\]

where we have chosen \( \hat{c} = 2c < 1 \). We conclude \( I(n) = O(M(n)) \). That is, inversion is no harder than multiplication, provided the matrix is symmetric positive definite.
We now consider a general nonsingular $A$ and note that $A^T A$ is symmetric positive definite. Specifically $A^T A$ is certainly symmetric and $A$ nonsingular implies $A$ has full column rank, which we have shown implies positive definite. We also note

$$(A^T A)^{-1} A^T = A^{-1} (A^T)^{-1} A^T = A^{-1}.$$ 

So, we proceed as follows to invert $A$ in $O(M(n))$ time.

1. Construct $A^T A$ in $O(M(n))$ time.
2. Invert $A^T A$ in $O(M(n))$ time.
3. Post-multiply by $A^T$ in $O(M(n))$ to obtain $A^{-1}$.

We conclude that $I(n) = O(M(n))$, where now $I(n)$ is the time complexity to invert any nonsingular $n \times n$ matrix. Inversion is no harder than multiplication. ■
Corollary 28.6 (to the Schur complement lemma): LU-decomposition of a positive definite matrix never causes a division by zero.

Proof. Recall that the Schur complement lemma shows that the Schur complement of a positive definite matrix is itself positive definite.

The LU-decomposition starts by decomposing the given matrix $A$ as

$$A = \begin{bmatrix} a_{11} & w^T \\ v & A' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ v/a_{11} & I_{n-1} \end{bmatrix} \cdot \begin{bmatrix} a_{11} & w^T \\ 0 & A' - vw^T/a_{11} \end{bmatrix}$$

The scalar $a_{11}$ is a $1 \times 1$ leading submatrix of a positive definite matrix, and therefore $[a_{11}]$ is also positive definite. For a $1 \times 1$ matrix, the quadratic form $x^T[a_{11}]x = a_{11}x^2$, since $x$ must also be a scalar. Then $a_{11}x^2 > 0$ for all nonzero $x$ forces $a_{11} > 0$. The decomposition then produces no division-by-zero error.

The LU-decomposition then proceeds on the Schur complement $A' - vw^T/a_{11}$, which is itself positive definite. We may assume by induction that we can achieve

$$A' - vw^T/a_{11} = L'U'$$

without encountering a division-by-zero. We then have

$$A = \begin{bmatrix} 1 & 0 \\ v/a_{11} & I_{n-1} \end{bmatrix} \cdot \begin{bmatrix} a_{11} & w^T \\ 0 & L'U' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ v/a_{11} & L' \end{bmatrix} \cdot \begin{bmatrix} a_{11} & w^T \\ 0 & U' \end{bmatrix} = LU.$$
We can avoid pivots with a general nonsingular matrix as follows.

1. The solutions to $Ax = y$ and $A^T Ax = A^T y$ are identical.

2. $A^T A$ is symmetric positive definite.

3. We can obtain $A^T A = LU$ without pivoting, with no division-by-zero errors, and in $O(n^3)$ time.

4. As computing $A^T y$ and solving $LUx = A^T y$ are both $O(n^2)$, we obtain the general solution in $O(n^3)$ time — without pivoting.

5. The authors suggest that this method is not widely used due to numerical instability. The division element may indeed all be non-zero, but they can be very small.
Least Squares Approximation: Given \( m \) data points \((x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\), we seek a function \( F(x) \) such that

\[
\sum_{i=1}^{m} [y_i - F(x_i)]^2 = |y - F(x)|^2
\]

is minimum. The typical exploration of this topic restricts the search space for \( F \) to functions of the form

\[
F(x) = \sum_{j=1}^{n} c_j f_j(x),
\]

where the collection \( \{f_1(x), f_2(x), \ldots, f_n(x)\} \) are called basis functions. Moreover, the usual situation is \( m << n \). That is, the data set is much larger than the number of basis functions. We will use basis functions \( \{1, x, x^2, \ldots, x^{n-1}\} \). That is, \( f_j(x) = x^{j-1} \), for \( 1 \leq j \leq n \).

\( n = 0 \) is called a best constant fit to the data. \( n = 1 \) is a best linear fit. \( n = 2 \) is a best quadratic fit. And so forth.
The matrix formulation is

$$
\begin{bmatrix}
F(x_1) \\
F(x_2) \\
\vdots \\
F(x_m)
\end{bmatrix} = 
\begin{bmatrix}
f_1(x_1) & f_2(x_1) & \cdots & f_n(x_1) \\
f_1(x_2) & f_2(x_2) & \cdots & f_n(x_2) \\
\vdots & \vdots & \ddots & \vdots \\
f_1(x_m) & f_2(x_m) & \cdots & f_n(x_m)
\end{bmatrix}
\begin{bmatrix}
c_1 \\
c_2 \\
\vdots \\
c_n
\end{bmatrix}
$$

or $F = Ac$ with $A = [a_{ij}] = f_j(x_i) = x_i^{j-1}$. $|\eta|^2 \equiv |F(x) - y|^2 = |Ac - y|^2$ is then the minimization goal.
\[ |\eta|^2 = \sum_{i=1}^{m} \left( y_i - \sum_{j=1}^{n} a_{ij}c_j \right)^2 \]

\[ \frac{d|\eta|^2}{dc_k} = \sum_{i=1}^{m} 2 \left( y_i - \sum_{j=1}^{n} a_{ij}c_j \right) \cdot a_{ik} = 0, \quad \text{for } k = 1, 2, \ldots, n \]

\[ \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}c_j a_{ik} = \sum_{i=1}^{m} y_i a_{ik} \]

\[ \sum_{j=1}^{n} c_j \sum_{i=1}^{m} a_{ij} a_{ik} = \sum_{i=1}^{m} y_i a_{ik} \]

\[ \sum_{j=1}^{n} c_j (A^T)_{ji} A_{ik} = \sum_{i=1}^{m} y_i A_{ik} \]

\[ \sum_{j=1}^{n} c_j (A^T A)_{jk} = (y^T A)_k \]

\[ (c^T A^T A)_k = (y^T A)_k \]

\[ c^T A^T A = y^T A \]

\[ (A^T A)c = A^T y \quad \text{normal equations} \]
We note that $A^T A$ is symmetric and of size $n \times n$. If $A^T A$ is full column rank, then it is positive definite and consequently nonsingular. In that case, we proceed via the LU decomposition.

1. Compute $A^T y$, which is $(n \times m) \cdot (m \times 1) = (n \times 1)$ and requires $\Theta(mn)$ operations.

2. Compute $A^T A$, which is $(n \times m) \cdot (m \times n) = (n \times n)$ and requires $\Theta(n^2 m)$ operations.

3. Decompose $A^T A = LU$ in $\Theta(n^3)$ operations.

4. Solve for $c$ via triangular systems in two $\Theta(n^2)$ operations.

5. As $m >> n$, the most expensive operation is $O(n^2 m)$, and therefore the entire algorithm is $O(n^2 m)$.