1. (i). Prove that for any n greater than or equal to 2 there is a non-singular square (n by n) matrix which has no LU decomposition with L unit lower triangular and U upper triangular. In addition, give a specific example of your matrix when n=3 and show its LUP decomposition.

Answer: Let \( M_n \) be the n by n matrix with 1's along its anti-diagonal (this is the diagonal which goes from its upper right to its lower left corner.).

Such and \( M \) is nonsingular as its determinant is 1 or -1. It has no LU decomposition because assuming there is such an L and U you would conclude that the top left corner of U would be 0, but then U would be singular and hence so would be \( M \). (The proof of this is a simple proof by contradiction.)

For the n=3 example, take \( M_3 \) as described above then let permutation P to be the one that exchanges the first and third row of M, making PM = the identity which clearly can be decomposed as LU.

(ii). Give an examples of a singular 2 by 2 matrix A and a singular 3 by 3 matrix B which have LU decompositions.

Answer: Really this problem was too simple and should have at least said that the matrices A and B could not be the all zero matrices, or even have no all 0 rows or columns.

\[
\begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} 3 & 2 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 3 & 2 \\ 6 & 4 \end{pmatrix} \text{ is such a 2 by 2 example.}
\]

and so

\[
\begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 3 & 2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 3 & 2 & 0 \\ 6 & 4 & 0 \\ 0 & 0 & 0 \end{pmatrix} \text{ is such a 3 by 3 example.}
\]

2. In order to find the global min cut of \( G = (V, E) \) we create a set of flow networks \( S = \{G_{(1,1)}, \ldots, G_{(1,n)}, G_{(2,1)}, \ldots, G_{(n,n)}\} \) and run the Edmonds-Karp algorithm on each of the flow networks in the set. For each undirected edge \((u, v) \in E\), each \( G_{(i,j)} \) will have the directed edge \((u, v)\) and the directed edge \((v, u)\). All of the edges in \( G_{(i,j)} \) will have capacity 1. Graph \( G_{(i,j)} \) will use node \( i \) from \( G \) as a source and node \( j \) from \( G \) as the sink. The min cut of \( G \) is the min cut of all of the graphs in \( S \). There are \( |V|^2 \) graphs which we run the Edmonds-Karp algorithm on and each of the graphs have the same number of nodes as \( G \) and twice as many edges as \( G \). Therefore, the running time of the algorithm is \(|V|^2 |V^2(2|E|)|^2 = O(|V|^9 |E|^2)\).
3. This problem is a fairly simple application of finding a max matching in a bipartite graph, however the proof that it is correct is not easy. The details are given below, but we did not expect that you would get all the details of the proof. Mostly you should be able to write the algorithm and understand the main idea of why it works. In the answer the word rearrangeable was used instead of switchable but they have the same meaning.

Part a

A matrix that is not rearrangeable but has at least one 1 on every line and every column is:

\[
\begin{align*}
0 & 1 & 0 \\
1 & 1 & 1 \\
0 & 1 & 0 \\
\end{align*}
\]

Part b

Let \( A \) be an array \( n \times n \) of 0s and 1s. Construct the following bipartite graph \( G \):

\[
V' = \{x_1, x_2, \ldots, x_n\} \cup \{y_1, y_2, \ldots, y_n\}
\]

\[
E' = \{x_i, y_i | A(i,j) = 1\}
\]

And make it a flow graph by adding a source and a sink from each side of the graph.

\[
V' = V' \cup \{s, t\}
\]

\[
E' = E' \cup \{(s, x_i) | i \in V\} \cup \{(y_i, t) | i \in V\}
\]

Make the capacity of each link 1.
Each node on the left side corresponds to one row and each node on the right corresponds to one column. If there is an edge \((x_i, y_j)\) in the graph that means that in the array \( A(i, j) = 1 \).

Find the maximum matching, (as described at section 26.3 of the book). If the maximum matching is \( n \), then the matrix is rearrangeable and vice versa. We are going to prove both directions.

If the matrix is rearrangeable then the maximum matching of \( G \) is \( n \).

Notice that every row-row switch is the same as relabeling two nodes from the right side of the graph and every column-column switch corresponds to the relabeling of two nodes of the right side of the graph., but is doesn’t affect the maximum flow of the graph, since we don’t add, remove or change edges. Since the maximum flow of the graph is the same the maximum matching also don’t change. Let \( R \) be the set of rearrangements, switches between rows and columns, that will create a matrix from \( A \), that is going to have all ones on the diagonal. We will denote as \( R(A) \) the matrix that results after applying the rearrangements. Find \( A' = R(A) \).

In the graph apply all the operations in \( R \) by relabeling the nodes, e.g if in \( R \) there is the operation \( r_i \rightarrow r_j \), \( r_i \) denotes the row \( i \), then relabel \( x_i = x_{jand}x_j = x_i \), and get a new graph \( G' \), that has the same maximum matching as \( G \). This \( G' \) is the bipartite graph that corresponds to matrix \( A' \). Since in \( A' \) there are ones on the main diagonal, on \( G' \) there is an edge \((x_i, y_i)\) \( \forall i \leq n \). So the maximum matching is \( n \).
PROB. 3. If the maximum matching of $G$ is $n$, then the matrix is rearrangeable.

Let $M$ be the set of matchings, where $|M| = n$. For every node on the right part of the graph $x_i$ find its matching in $M$, $y_j$. Relabel $y_j$ as $y_i$ and vice versa. In the matrix switch $c_i \rightarrow c_j$, where $c_i$ is the column $i$. In the end the resulting graph $G'$ will have an edge $(x_i, y_i) \forall i \leq n$. The corresponding $A'$, that resulted by just switching columns, must have ones on the main diagonal. So $A$ is rearrangeable.

This algorithm takes $O(n^5)$ time to build the graph and then it takes $O(n^5)$ to find the matching. Note that the running time of Edmond Karp is $O(|V||E|^2)$ and we know that $|E| \in O(|V|^2)$, so Edmond Karp runs in $O(|V|^5)$.

4. For the first step in computing $L$ and $U$, $a_{11} = -1$, $v = \begin{bmatrix} -3 \\ -2 \end{bmatrix}$, $w' = \begin{bmatrix} -2 \\ 3 \end{bmatrix}$, and $A' = \begin{bmatrix} -12 & 10 \\ 7 & -5 \end{bmatrix}$. First we compute the Schur complement of $A$ with respect to $a_{11}$, or $A' - uv'^T/a_{11}$.

$$
A' - uv'^T/a_{11} = \begin{bmatrix} 12 & 10 \\ 7 & -5 \end{bmatrix} - \begin{bmatrix} -5 \\ 2 \end{bmatrix} \begin{bmatrix} 3 \\ -2 \end{bmatrix} = \begin{bmatrix} 17 & 10 \\ 7 & -5 \end{bmatrix} - \begin{bmatrix} 15 \\ 6 \end{bmatrix} / -1 = \begin{bmatrix} -2 \\ 15 \\ -6 \end{bmatrix}, 
$$

so $A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix}$ but the second matrix is not yet upper triangular so we recurse on $A' - uv'^T/a_{11} = \begin{bmatrix} -27 \\ 20 \\ 13 \end{bmatrix}$. Now $a_{11} = -27$, $v = \begin{bmatrix} 13 \\ 20 \end{bmatrix}$, $w' = \begin{bmatrix} 20 \\ -9 \end{bmatrix}$ and $A' = \begin{bmatrix} -9 \end{bmatrix}$.

Thus, $A = LU = \begin{bmatrix} 1 & 0 & 0 \\ 5 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 3 \\ 0 & -27 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 2 \\ -2 \\ -\frac{13}{27} \end{bmatrix}$.

To solve $Ax = b$ we first solve $Ly = b$ so we have $\begin{bmatrix} 1 & 0 & 0 \\ 5 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 2 \\ -2 \\ -\frac{13}{27} \end{bmatrix}$.

Lastly, we solve $Ux = y$ or $\begin{bmatrix} -1 & 3 & -2 \\ 0 & -27 & 20 \\ 0 & 0 & \frac{13}{27} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ -10 \\ -\frac{13}{27} \end{bmatrix}$. $Ux = y$ gives us that $\frac{17}{27}x_3 = -\frac{1093}{27}$ or $x_3 = -\frac{1093}{17}$. $-27x_2 + 20 \cdot \frac{1093}{17} = -10$ or $x_2 = \frac{70}{17}$ and $-x_1 + 3 \cdot \frac{70}{17} = 2$ or that $x_1 = \frac{39}{17}$. Thus, $x_1 = \frac{39}{17}, x_2 = \frac{70}{17}$ and $x_3 = \frac{-1093}{17}$ is the solution to $Ax = b$. 

a. By applying the procedure of the chapter, we obtain that

\[
L = \begin{pmatrix}
  1 & 0 & 0 & 0 & 0 \\
  -1 & 1 & 0 & 0 & 0 \\
  0 & -1 & 1 & 0 & 0 \\
  0 & 0 & -1 & 1 & 0 \\
  0 & 0 & 0 & -1 & 1 \\
\end{pmatrix}
\]

\[
U = \begin{pmatrix}
  1 & -1 & 0 & 0 & 0 \\
  0 & 1 & -1 & 0 & 0 \\
  0 & 0 & 1 & -1 & 0 \\
  0 & 0 & 0 & 1 & -1 \\
  0 & 0 & 0 & 0 & 1 \\
\end{pmatrix}
\]

\[
P = \begin{pmatrix}
  1 & 0 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 & 0 \\
  0 & 0 & 1 & 0 & 0 \\
  0 & 0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 0 & 1 \\
\end{pmatrix}
\]

b. We first do back substitution to obtain that

\[
Ux = \begin{pmatrix}
  5 \\
  4 \\
  3 \\
  2 \\
  1 \\
\end{pmatrix}
\]

So, by forward substitution, we have that

\[
x = \begin{pmatrix}
  5 \\
  9 \\
  12 \\
  14 \\
  15 \\
\end{pmatrix}
\]

c. We will set \(Ax = e_i\) for each \(i\), where \(e_i\) is the vector that is all zeroes except for a one in the \(i\)th position. Then, we will just concatenate all of these solutions together to get the desired inverse.

This gets us the solution that

\[
A^{-1} = \begin{pmatrix}
  1 & 1 & 1 & 1 & 1 \\
  1 & 2 & 2 & 2 & 2 \\
  1 & 2 & 3 & 3 & 3 \\
  1 & 2 & 3 & 4 & 4 \\
  1 & 2 & 3 & 4 & 5 \\
\end{pmatrix}
\]