1. CLR 25.2-1
2. CLR 25.2-2
3. CLR 25.2-4
4. Here is an application of Strongly Connected Components (SCCs) to another field of study, Linear Algebra. You do not need to know linear algebra to do this question, but it will help you to understand the motivation below.

   In linear algebra the standard problems we solve are to solve linear systems of equations defined by matrices, compute determinants of matrices, and find eigenvalues of matrices. In engineering and science the matrices we often encounter are very large and *sparse*, which means they have a lot of zero entries. (One popular matrix people are analyzing these days is the Web: matrix A has one row and column for each document on the Web, and $A(i,j)$ is the number of hyperlinks from document i to document j. Google uses this matrix in its web search algorithm. Other applications of sparse matrices include earthquake engineering, circuit analysis, computer vision, building nuclear bombs etc.)

   It turns out that SCCs can make solving all these linear algebra problems much faster for large sparse matrices.

   We consider determinants here, for illustration. The matrix for which it is easest to compute the determinant a *diagonal matrix*, i.e. one that is nonzero only on the diagonal, because its determinant is just the product of the diagonal entries. For example

$$\det\left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 7 \end{bmatrix}\right) = 1 \cdot 2 \cdot 5 \cdot 7 = 70$$

   (here $\det(A)$ means the determinant of matrix A). The next simplest matrix is an *upper triangular matrix*, that is one that is nonzero only on and above the diagonal, because its determinant is also just the product of the diagonal entries. For example

$$\det\left(\begin{bmatrix} 1 & 3 & 1 & -8 \\ 0 & 2 & -4 & 100 \\ 0 & 0 & 5 & 6 \\ 0 & 0 & 0 & 7 \end{bmatrix}\right) = 1 \cdot 2 \cdot 5 \cdot 7 = 70$$

   The next simplest matrix is *block upper triangular*. This is a matrix that can be devided into rectangular blocks as shown below, and is triangular by blocks. Its determinant is just the product of the determinants of the diagonal blocks:

$$\det\left(\left[\begin{array}{cc|ccc|c} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 1 & 7 & 8 & 2 & 1 \\ \hline 0 & 0 & 3 & 1 & 1 & 1 \\ 0 & 0 & 1 & 3 & 1 & 6 \\ 0 & 0 & 1 & 1 & 3 & 8 \\ \hline 0 & 0 & 0 & 0 & 0 & 7 \end{array}\right]\right) = \det\left(\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}\right) \cdot \det\left(\begin{bmatrix} 3 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{bmatrix}\right) \cdot (7) = (-3) \cdot (20) \cdot (7) = -420$$

More generally, a block upper triangular matrix with 3 blocks looks like

$$\det\left(\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A_{22} & A_{23} \\ 0 & 0 & A_{33} \end{bmatrix}\right) = \det(A_{11}) \cdot \det(A_{22}) \cdot \det(A_{33})$$

where each A_{ij} is itself a matrix with n_i rows and n_j columns. In general there may be any number of diagonal blocks of any sizes. When the diagonal blocks A_{ii} are much smaller than the original matrix, this can be much faster than computing the determinant of a general matrix, lowering the cost of computing a determinant as much as from $O(n^3)$ to $O(n)$.

The situation is similar for matrix inversion, where we really only have to invert the diagonal blocks A_{ii}, and for computing the eigenvalues, where we only have to compute the eigenvalues of the diagonal blocks. In both cases, the cost drops significantly when the diagonal blocks are small.

Unfortunately, most matrices are not triangular or block-triangular. However, they more often turn out to have these structures "in disguise." For exmple, consider

$$A = \begin{bmatrix} 2 & 100 & -4 & 0 \\ 0 & 7 & 0 & 0 \\ 0 & 6 & 5 & 0 \\ 3 & -8 & 1 & 1 \end{bmatrix}$$

It turns out that if we change the order of rows and columns of this matrix, so that the rows and columns appear in the order 4,1,3,2 instead of the initial order 1,2,3,4, then this does not change the determinant (or change the eigenvalues, and only changes the order of the rows and columns of the inverse), and yields the matrix we have seen before:

$$\det(A) = \det\left(\begin{bmatrix} 1 & 3 & 1 & -8 \\ 0 & 2 & -4 & 100 \\ 0 & 0 & 5 & 6 \\ 0 & 0 & 0 & 7 \end{bmatrix}\right) = 1 \cdot 2 \cdot 5 \cdot 7 = 70$$

Given a large A, how would we ever compute the permutation 4,1,3,2 that makes the matrix triangular, or block triangular, and so make computing determinants etc. much cheaper? The answer is SCCs and Topological Sorting.

To present the algorithm, we need the following definition:

Def. Let A be an n-by-n matrix. Its *graph* $Gr(A)$ is the directed graph with n vertices numbered from 1 to n, with an edge (i, j) if and only if $A_{ij} \neq 0$, and with nonzero edge weights $w(i, j) = A_{ij}$.

It is easy to see that if we were to change the nonzero entries of A to 1, we would get the adjacency matrix of $Gr(A)$. $Gr(A)$ is essentially just another data structure for representing A; they contain the same information. Said another way: to every directed graph G with nonzero edge weights there is a corresponding matrix A such that $G = Gr(A)$.

```
MATRIX-BLOCK-TRIANGULARIZE(A)
    Compute the SCCs of Gr(A), and topologically sort them
    Renumber the vertices of Gr(A) in increasing topological order;
        call this new graph G' (this order is not always unique)
    Output the matrix A' whose graph is G': G' = Gr(A').
```

(a) Show that the matrix A' computed by MATRIX-BLOCK-TRIANGULARIZE(A) is just A with its rows and columns reordered by the same permutation.

(b) Show that A' is block upper triangular, with k diagonal blocks A'_{ii} of dimension n_i-by-n_i, where k is the number of SCCs of $Gr(A)$, and n_i is the number of vertices in the i-th SCC, where the SCCs are numbered in increasing topological order.

(c) Let A'_{ij} be the i, j-th block entry in A'. Show that $A'_{ij} \neq 0$ if and only if there is an edge in G from the i-th SCC to the j-th.

(d) Show that if B is a block upper triangular matrix with diagonal blocks B_{ii} of dimension n_i-by-n_i, then each group of vertices numbered $1 + \sum_{k<i} n_k$ through $\sum_{k \leq i} n_k$ (for any choice of i) is a union of SCCs of $Gr(B)$. This question shows that the sizes of the strongly connected components are the minimum size of the diagonal blocks in any block diagonal form; in other words computing the SCCs of $Gr(B)$ gives the "best possible" block triangular form.

Postscript: We note that there are many other, more sophisticated algorithms for handling large sparse matrices that are much faster for problems like determinats, matrix inversion etc., even when there is only one SCC. These algorithms are collectively called *sparse matrix algorithms.*