What is a sparse matrix and what features should it possess?

Sparse matrix is a matrix that has very few non-zero values.

Some features are:

1. Less memory is needed for storing a sparse matrix than storing a dense matrix of the same dimensions (by using a more efficient storage scheme).
2. Matrix calculations can be simplified by using more efficient algorithms for sparse matrices.

What sources of information should you consult to derive features? Do analogies exist? If so with what?

We can consult existing libraries for sparse matrices. For example, MatLab/Octave/Python (scipy) have very extensive implementations for sparse matrices. They can serve as a baseline from where we choose to derive features.

Additionally, we can also derive features from actual applications. For example, sparse matrices are used in some machine learning algorithms, so we can derive features by seeing what are some commonly used operations.

An analogy would be designing a building. Depending on the purpose of the building (laboratory, health care facility etc), the characteristics of the design would be chosen. Domain knowledge of the purpose of the building and its inhabitants would first need to be elicited to understand what needs should be satisfied by the design and what problems would need to be solved.

Who is likely to be the user of a sparse matrix package? What features are they likely to demand?

Some potential users are:

1. Mathematicians/Physicists: For example, solving partial differential equations.
2. Scientists in various areas: machine learning, computer graphics, statistics, and etc.
3. Game Engines often require fast matrix calculations to be done

Scientific applications would emphasize accuracy while engineering applications would emphasize optimizations in speed and storage. Some potential features required are:

1. Matrix addition
2. Matrix multiplication
3. Matrix transpose
4. Finding the determinant
5. Finding the inverse

What is a tri-diagonal matrix?

In linear algebra, a tridiagonal matrix is a band matrix that has nonzero elements only on the main diagonal, the first diagonal below this, and the first diagonal above the main diagonal.(Wikipedia). It is important to note that tri-diagonal matrices are square matrices.
E.g:

$$\begin{pmatrix} 1 & 4 & 0 & 0 \\ 3 & 4 & 1 & 0 \\ 0 & 2 & 3 & 4 \\ 0 & 0 & 1 & 3 \end{pmatrix}.$$

What is the relationship between a tri-diagonal matrix and a generic sparse matrix?

Tri-diagonal matrices are a special type of sparse matrix (given large dimensions).

Are tri-diagonal matrices important? And should they impact your design? If so, how?

Tri-diagonal matrices are important, because although it is a type of sparse matrix, it is even simpler than a general sparse matrix. Computations with tri-diagonal matrices can use even more efficient algorithms.

It does impact our design, if a tri-diagonal matrix is being operated on, we can employ the more efficient algorithms for the operation and even store it differently. Further details on how this can implemented is provided in the answers to the next questions.

What is a good data representation for a sparse matrix?

We can store a sparse matrix using linked lists and 2D arrays or multiple arrays. One particular way for the last option is the Compressed Sparse Row form.

The CSR or the Yale Format is similar to the Array Representation of Sparse Matrix. We represent a matrix M (m * n), by three 1-D arrays or vectors called as A, IA, JA. Let the number of non-zero elements in M and note that 0-based indexing is used.

1.The A vector is of size non-zero elements and it stores the values of the non-zero elements of the matrix. The values appear in the order of traversing the matrix row-by-row

2.The IA vector is of size m+1 stores the cumulative number of non-zero elements upto (not including) the i-th row. It is defined by the recursive relation :

IA[0] = 0 I
A[i] = IA[i-1] + number of non-zero elements in the (i-1) th row of the Matrix

3.The JA vector stores the column index of each element in the A vector. Thus it is of size NNZ as well.

Assume that you have a customer for your sparse matrix package. The customer states that their primary requirements as: for a N x N matrix with m non-zero entries

- **Storage should be ~O(km), where k << N and m is any arbitrary type defined in your design.**
- **Adding the m+1 value into the matrix should have an execution time of ~O(p) where the execution time of all method calls in standard Ruby container classes is considered to have a unit value and p << m ideally p = 1 In this scenario, what is a good data representation for a sparse matrix?**

The good data representation for the sparse matrix is using CSR yale format.

Design Patterns are common tricks which normally enshrine good practice

- Explain the design patterns: Delegate and Abstract Factory
- Explain how you would approach implementing these two patterns in Ruby
- Are these patterns applicable to this problem? Explain your answer.

The Delegate pattern uses composition to delegate tasks to other classes. When a functionality is called for the class, it delegates that task to one of its composite classes. This hides the complexity of implementation from the client code.

The Abstract Factory pattern produces families of related objects without specifying their concrete classes. Therefore, to the client code, only the abstract class gets created but within the implementation, we can construct the concrete class required by the particular situation.

In ruby, the delegate pattern can be implemented by creating the delegate class and then the functional classes to which all tasks would be delegated to are created. Any interaction with the functional classes would be through the delegate class.

To implement the abstract factory class in ruby, the abstract factory class is created which would contain an abstract method for creation and then the concrete factory classes would be implemented which would override the abstract method.

Yes, these patterns greatly streamline the design of the library and increase maintainability and extensibility. The abstract factory could be an abstract MatrixFactory which would be inherited by SparseMatrixFactory and TriDiagonalMatrixFactory. The delegate class would be essential in this respect, because if we are creating a special type of sparse matrix (e.g tri-diagonal matrix) whose operations can be significantly sped up by using a specialized algorithm the delegate class could create a sub-class that can handle such operations. This complexity can be hidden from the client code. And the delegate pattern would allow us to delegate functionality to other, potentially better and more robust, code.

The Abstract Factory method can also be used to later add more specializations for sparse matrices without much effort.

What implementation approach are you using (reuse class, modify class, inherit from class, compose with class, build new standalone class); justify your selection.

Composition will be used for our sparse matrix. In particular, our sparse matrix will contain a NMatrix object from NMatrix (Fast Numerical Linear Algebra Library for Ruby).

NMatrix already has a sparse implementation using the yale format, and being able to reuse that code is going to save a lot of engineering effort. Additionally, we chose composition over inheritance because we do not want the dependency on NMatrix to be exposed to the outside world.

Are iterators a good technique for sparse matrix manipulation? Are "custom" iterators required for this problem?

Yes, iterators, and more importantly, custom iterators may prove invaluable for sparse matrix manipulation. Since they could be different representations in data for different matrices, a custom iterator could provide a unified way to access elements of the matrices.

What exceptions can occur during the processing of sparse matrices? And how should the system handle them?

The major input exceptions that might occur are:
Creating matrices with sizes < 1.
This can be handled by checking the size of the matrix during construction.
Accessing an element that is out of range
This can similarly be done by checking the size of the matrices against the index being accessed.

Finding the inverse of a matrix or the determinant where no solution exists
This can sometimes be checked before executing the operation but in other times, this is only known after attempting to find a solution, at which point an exception would need to be raised.

What information does the system require to create a sparse matrix object?

The client code would need to provide the type of data stored in the matrix and the dimensions of the required matrix.

What are the important quality characteristics of a sparse matrix package?

Efficiency: The entire purpose of a sparse matrix package is to more efficiently operate perform matrix calculations on a subset of matrices. Therefore, efficiency in operations is very important.

Accuracy: Since this is maths package, accuracy of results is crucial to be useful

Extendibility: After providing base functionality, it is important to allow extensions to the package according to the specific needs of the client.