Discussion Prompt Questions
Module 4
Brian Loughran
Johns Hopkins Data Structures

1. What are some scenarios in which you are likely to encounter a sparse matrix?
2. Why is it important to select the right data structure for storing and manipulating sparse matrices? What are the important factors to consider?
3. What are the pros and cons of using a linked implementation of a sparse matrix, as opposed to an array-based implementation?

ANSWERS

1. Sparse matricies are somewhat common in numerical applications. Jacobian matricies are often sparse, solving elliptic partial differential equations often requires manipulation of sparse matricies. Storing boundary conditions in finite element or computational fluid dynamics analysis often is done using sparse matricies.
2. Sparse matricies can often be very large, therefore it is computationally more efficient to perform computation on the matricies if they are compressed into another data structure. Selecting the right data structure will ensure that the sparse matrix is stored in a memory-efficient and computationally efficient way.
3. In using linked list and array implementation of storing sparse matricies, the row, column and value of the entry in the sparse matrix is stored for all non-zero elements. Doing this removes the need to store all of the zero elements, therefore compressing the matrix. In an array implementation, you have the advantage of being able to access any value of the sparse matrix at any time, but you have to reserve space in memory for how many elements you expect, otherwise you will run out of memory in your array. For linked lists, you have dynamic memory allocation, but you do not have the ability to access any element without going through the entire list.