Module 6 Prompt Questions
Jot down short answers for each prompt question as you go along. Review and update your answers after finishing and submit before Tuesday midnight. Click on "Module 6 - Discussion Prompt Questions" above to submit.
The prompt questions for this module are:

1. What are some scenarios in which you are likely to encounter a sparse matrix?

Sparse matrices might come up when recording the occurrence or count of an activity and that activity itself does not occur very frequently. Only 5-20% of values are non-zero in a sparse matrix.
Three examples include:
- Whether or not a user has watched a movie in a movie catalog. Most people do not watch more than 20% of the movies out there in the world.
- Whether or not a person is under 1 year old in the world. 1 if they are under 1, 0 if not. Most values in the sparse matrix will be 0.
- Championships won by a team per year. Most teams do not frequently win championships and so most recordings would be 0. Say the row is decade and column is the single digit. Most values in this matrix would be 0.

2. Why is it important to select the right data structure for storing and manipulating sparse matrices? What are the important factors to consider?

If you use a singly linked list with a head node to represent a sparse matrix where the head node contains row information, the problem is that it doesn't have column information. The head node only has row information, but it does not contain column information.

You might need to use a multi-linked list instead where each node has two pointers coming out of it which eventually lead back to row and column information. Each list needs to be doubly linked so the node can traverse back to the column. Or each list can be circular so that the node can follow the pointers until it goes back to the header node.

3. What are the pros and cons of using a linked implementation of a sparse matrix, as opposed to an array-based implementation?
An array based implementation might store the data like this:

[[1,3,4], [2,3,5]]

which is a nested array. The first element is an array with 3 values which might imply row 1, column 3, value 4. which contains all the information we need for each non-zero value. However, search is O(n) because we'd have to iterate through the array to find a value. Insertion is O(n/2) on average.

A circular multi-linked linked-list implementation also has a search time of O(n) but insertion is O(1). It might be harder though to find column and row information because the lists have to be traversed to find the row and column information whereas for the array-based implementation, all of the data is self-contained in the sub array.