Sean Connor
26 June 2018
Module 4 Discussion Prompt

1. What are some of the key distinguishing features of a list? Under what conditions would these be useful?

A list is an ordered collection of data items
All items in list are available for insertion or deletion
There is no limit on number of items a list can contain
There is no limit on the nature of items a list can contain

Lists seem incredibly useful in many different circumstances. For example, a linked list seems like it would be decent for storing many similar objects, like a sort of database due to the ability of dynamic allocation.

2. Should a list ADT include hints or references to the list implementation?

Yes, because the capability/features of the list cary based on its implementation, like (if I'm not mistaken) Java's LinkedList and ArrayList.

3. What might you consider to be a nice-to-have in a list ADT (not necessary, but potentially useful)?

List.search(search-term)
List.isEmpty()
List.copy()
List.length
List.print()

4. When might the implementation details of a list be an important consideration?

If the list needs to be searched frequently, an ordered array implementation may be preferable because of log n time complexity of binary search. Also, if one does not know how much space will be needed, the dynamic allocation featured in the linked list may be a better choice. Otherwise, the implementations provide very similar features.

5. List some pros and cons of an array implementation of a list.

Pro:

Sean Connor
26 June 2018
Module 4 Discussion Prompt

Random access – can easily access any item in list

Con:
Static/fixed size
List must be homogenous
Standard method insertion/deletion requires shifiting (O(n/2))

6. List some pros and cons of a linked implementation of a list.

Pro:
No size limit (dynamic allocation)

Con:
Sequential access
List must be homorgenous

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

I have little (or maybe no) experience with sparse matrices, so I can't really say where I may encounter them. One thing I found online was that sparse matrices are often encountered when dealing with partial differential equations. A practical example I found was representing a city's mass transit network.

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

A lot of space can be wasted representing the many zero values. In addition, many operations on the matrix will be redundant. Put simply, there will be increased time and space complexity when working with a sparse matrix, especially if the data structure is not well suited for the task. It is important that the data structure keep track of only relevant values, to minimize increased time and space complexity.

9. What are the pros and cons of using a linked implementation of a sparse matrix, as opposed to an array-based implementation?

The pros and cons in this case would be similar to the pros and cons of a list implementation. An array will be a fixed size, but incorporate random access to

Sean Connor
26 June 2018
Module 4 Discussion Prompt

easily access any element in the array. A linked implementation will be sequential access, but allow for dynamic allocation. If the number of relevant nodes is unknown, a linked implementation may be better – otherwise, it seems like it would quite easy to over or underestimat the size of the array one may need.