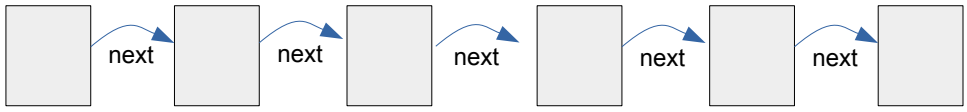
Data Structures

Basic idea is to practice implementing basic (or not so basic) data structures without the help of the libraries your language allows.

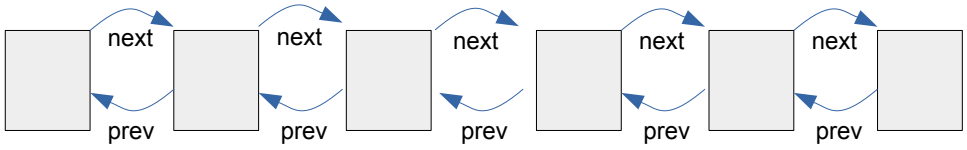
Linked Lists

1) Write a data structure for a singly linked list element. Just define as a POD (plain old data type). Don't get carried away with constructors, etc.



- What did you choose as your payload? Why?
- What is the best choice your language could support?

2) Write a structure for a doubly linked list.



3) Write a function to reverse a linked list.

- The old list may be unusable afterwards.
- Do both types (single and double).
- What are the space and time complexity?
- How did you test it? Could you show an interviewer that you tested it on a whiteboard?
- Can you break it? Make your code fool proof.
- What happens if there is a cycle in the singly linked list? Are you sure what you think happens happens?
- If you were a functional programmer, you may have used Rev(h::t)=Rev(t):::h. What are the complexities of that?

4) Can you reverse a doubly linked list in T(n) = O(1)?

- Hint: use an extra bit (not so easy for Java programmers)

5) Write a function to test if a singly linked list has a cycle.

- First do this by remembering nodes as you traverse the list.
- Then add a field to each node to remember if you have traversed it. If using C, add that field without increasing the size of your struct (hint: on most systems, how are structs aligned?)
- Then do it by the Tortoise and the Hare - two pointers travelling at different speeds.
- How can you tell by iteratively reversing the list?
- What are the complexities of all the above?

6) Write code to check if a double linked list is well formed.

7) Write code to traverse a singly linked list.

- What did you do at each element?
- Did you allow the user to specify what to do?
- What if the user wants to finish early?

8) Write a function to insert a node at the nth position in a list (do it for both single and double).

9) Sort a linked list.

- How do users specify the element comparison function?
- When does it become more efficient to copy to an array, sort, then rebuild the list?
- Have you tried this on your machine?

10) Remove duplicates from a linked list

11) What is an XOR list?

Notes:
With linked list problems, always ask “Single or Double?”

Vector (Array List)

Implement a vector (or array list) and associated operations (append, get, insert, etc).

- Does it allow generic elements?
- What are the runtimes of the operations (esp. append)?

Hashtable

Implement a hash table and associated operations.

- What methods of collision resolution are there to choose from? What are the pros and cons?
- What happens when a hash table grows/shrinks?
- How much does your choice of hash function matter? Try running some runtime experiments to find out.

Binary Tree

Implement a binary tree and associated operations.

- What are the different walk functions you can use? Implement them.
- Compare runtimes of operations across different data structures

Others

Implement:

- *Tree*
- *Heap*
- *Red-black tree*
- *Skip List*

-What other data structures can you find? Implement and analyse as many as you find.