Linked Lists

Kevin Browne

Data structures

- A **data structure** is a data organization format
- Data structures typically allow us to create, read, update and delete data
- Data structures may have algorithms associated with them, that allow for the access, modification, etc, of the data inside the data structure
- A data structure contains a set of data, typically data that is related in some way

Data structures

- An **array** is a relatively primitive data structure supported by most languages in some form
  - We store, retrieve, access and modify a set of data

- Other data structures may either be supported directly in the language (e.g. lists in Python)
  - ...or we may build them from more primitive types!

- The creation of data structures and the algorithms used to access them is a critical part of computer science

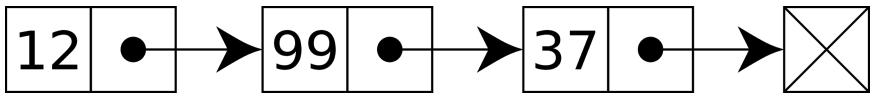
Data structures

- Some data structures perform better in certain situations than other data structures!
- When implementing programs, we can choose a "best tool for the job" based on runtime speed, memory requirements, etc.
- We'll compare the performance of a Linked List vs an array in C!


LINKED
LISTS BE LIKE:
I KNOW A GUY
WHO KNOWS A GUY
imgflip.com

Linked lists

- A linked list is a data structure made up of elements called **nodes** which each **point** to the next node in the list



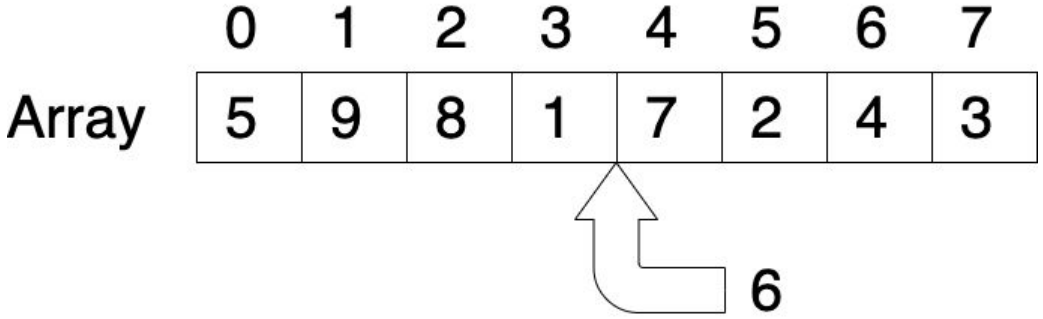
- We call the first node in the list the **head** node, and the last node in the list the **tail** node
  - The last element in the list points to "nothing" (e.g. NULL in C)

Source: https://en.wikipedia.org/wiki/Linked_list#/media/File:Singly-linked-list.svg

Why use a linked list?

- In general we use data structures because certain operations are faster or use less memory
- If we're comparing linked lists to arrays, one advantage of linked lists is the ease of inserting and deleting elements in terms of memory allocation
- If we need to reallocate space for an array, to make it smaller or larger, the entire array may need to be copied into a new space in memory

We can’t just easily insert an element into the middle of an array...

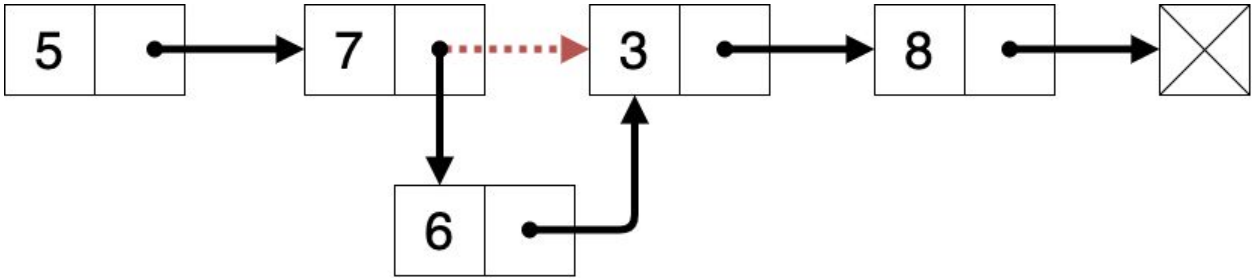


The array is not large enough to hold all of the elements. We could re-allocate for additional memory in the case of a dynamic array, but even then all of the elements to the right would need to be shifted over by after inserting this new element.

Array elements are contiguously arranged in memory…

But linked list nodes are connected by **pointers** and these can be re-arranged!

Inserting a node into the middle of a linked list...



We insert a node by having the previous node point to this new node instead, and have the new node point to the next expected node.

Why use a linked list?

- If we need to insert or delete an element from a linked list, we allocate or free space for one element and modify a pointer
- Not only is this not particularly expensive, the performance of this operation can also be relatively ***consistent*** too
- Inserting or deleting an element at the head of the linked list has an upper bound on how long it will take

Why use a linked list?

- A linked list can therefore be ***faster*** than a dynamically allocated array in some instances and have ***more consistent performance*** too

- We'll actually do a performance comparison to illustrate this later!

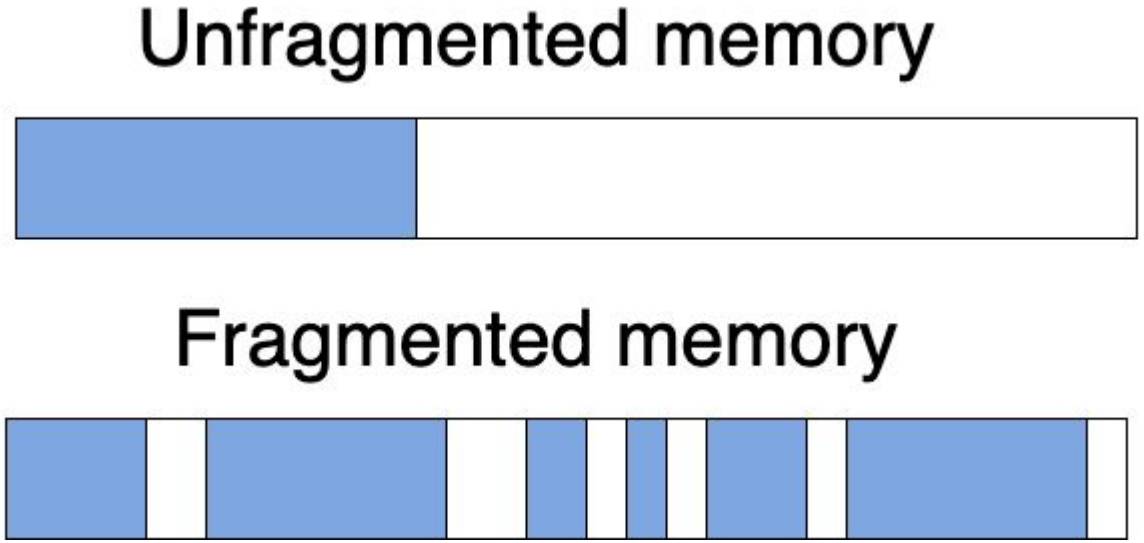
Why use a linked list?

- Linked lists can prevent **memory fragmentation** because nodes do not need to be stored contiguously in memory as with arrays

- New nodes can be stored anywhere in memory, they just need to point to each other!

- Though this can actually decrease performance too due to the way caching works on modern processors

A linked list node can ‘fit’ into the available spaces even when memory is fragmented, where as an entire re-allocated dynamic array may not.



Where do linked lists get used?

- Linked lists are really good for situations where we need to insert/remove elements **in the middle** of a list (because arrays are poorly suited for this)

- For example, a list of tasks to be done according to a priority level

- Or for example a playlist of connected songs or videos

Where do linked lists get used?

- Because some operations in linked lists have consistent performance, they can be useful in **real-time systems**
- Real-time systems operate with time constraints
  - e.g. the airbag in your car needs to deploy fast enough!
- Consistent performance can ensure time constraints are met

Linked lists

- In practice though, as with many data structures, the usage of linked lists is much more niche compared to using an array
- Data structures really give us a tool box of options that we can select the best tool for the job for
- By understanding linked lists, we add to our tool box!