# Big O notation

Big O notation tells you how fast an algorithm can run. If you have a list of n items, a simple search needs to check each element so that it will take n operations. The run time in big O notation is O(n). Big O lets you compare the number of operations. It tells you how fast the algorithm grows.

# Arrays and Linked Lists

Items are stored in your computer in memory. Think of it as drawers in a wardrobe, and each drawer/memory has an address.

Arrays are good for storing items contiguously(right next to each other), but what if you need more space in the future?

If you want to store three items in memory next to each other, you get the “drawers” for three but if you decide to add a fourth, you’re not entirely sure you can store it next to the three, because the next drawer may have been taken up by another person.

In that case, you will have to move your three items to a place that can fit four items, so you can fit all your items contiguously. If you decide to add another, you will have to move again.

A workaround is to hold extra space for future items just in case, but then again you may end up not using them and wasting memory. You may also end up needing more.

Linked Lists solve this problem

# Linked Lists

Linked Lists store the items but with the address of the next item. It is a bunch of random memory addresses linked together. You go to an item and then it tells you the address of the next item and so on and so on.

This solves another problem arrays can encounter, let's say you need a slot of 10,000 items in your array but your memory doesn't have 10,000 slots together. You can't get space for your array. However, if there’s space in your memory you have space for your linked list.

# What are arrays good for?

Websites with top 10 lists use a tactic to get more views, they don't put all top 10 on one page instead they start at number 10 and make you click next at the end of each page until you get to number 10.  
  
With arrays you know the address of every item in the array because they are all next to each other, but with linked lists, you have to go to one to know the address of the next, just like those top 10 websites.

In the linked list, if you want to read the last item, you can't just read it because you do not know the address, instead, you have to go to item 1 then get the address of item 2 and go to the address 2 to get the address of item 3 and so on and so on.

With arrays, that's not the case. Let's say you have an array of five items and it starts at address 00, you know the address of item 5 will be 04 because elements in an array are next to each other. That is also index 04 because arrays start at 0.

You can't do that with linked lists because they aren't next to each other

# Inserting into the middle of Lists

With lists, it’s as easy as changing what the previous element points to.

NB : The below explanation is from self-research and may be advanced

It is better with lists because you just have to change the pointer that points to the next address, and make it point to the newly inserted, x and then make x point to the address of the one it shifted down.

A -> B

Insert x into the middle

A ->x -> B

END

But for arrays you have to shift all the elements down/right and if there’s no space you have to copy all to a new location.

# Pointers

Pointes are what linked lists use to point to the next item in a list. With each item a list, you use a little bit of memory to store the address of the next item, which is a pointer.

Deletions

Again lists are better because you just need to change what the previous element points to, with arrays you need to move everything up when you delete an element. Unlike insertions, deletions will always work, insertions can fail when there’s no space left in the memory.

.

# Runtimes for arrays vs lists

|  | Arrays | Lists |
| --- | --- | --- |
| Reading | O(1) | O(n) |
| Insertion | O(n) | O(1) |
| Deletion | O(n) | O(1) |

O(n) = Linear Time

O(1) = Constant Time

It is worth mentioning that insertions and deletions are O(1) only if you can access the elements to be deleted instantly. It is common practice to keep track of the first and last items in a linked list so it would take O(1) time to delete those.