

Chapter 2

In the last chapter we used the terms arrays and lists quite a lot. Now we will get to understand the difference between them, and we'll learn our first sort algorithm as you know there are some algorithms that don't work unless your data is sorted.

Memory:

To understand arrays and list you need to know how memory works. Imagine you went to the gym, and you want to store your belongings so you go to the locker room and you ask for a locker but each locker can hold one item so if you want to store your phone and glasses you'll have to book two lockers and to know where your items are you will be given the number of each locker similarly this how memory works your computer works like a huge locker room and each locker in it has an address each time you want to store an item you ask your computer for space and it'll give you a memory address.

Arrays and linked lists:

Sometimes you'll need to store a list of items not just one item for example if you are working on a to do list app, you'll need to store list of items to work with them efficiently let's try to use arrays first assume this table is memory and the slots marked with X are used.

Prepare breakfast	Wash the car	study	X	X
	X	X		
	X	X		
		X	X	X

In this case your list only have only three items if you want to add another task you will notice that the slot next to you is booked so you can't use it so may be you'll think ok I'll use the next empty space but this isn't the case with an array the array always has it's elements in neighboring cells which mean the first element is next to the third with no empty or cells used by other variables in between so if you want to add one more task your program will ask the computer for a new memory chunk that consists of 4 slots

			X	X
Prepare break fast	Wash the car	study	workout	
	X	X		
	X	X		
		X	X	X

So, this is how your memory may look like after expanding the array.

So whenever you need to add another element you will have to ask for a new chunk and move your elements to the new address so adding more elements to an array is tiring process one solution to this is you may hold more room in the memory than you need in the memory for example you may book enough memory for 10 tasks from the beginning so when you add the fourth or the fifth task you won't face a two problems:

1-you may find your self not using the space you reserved like only using 5 or 6 slots while you are reserving 10 and that's a waste of space.

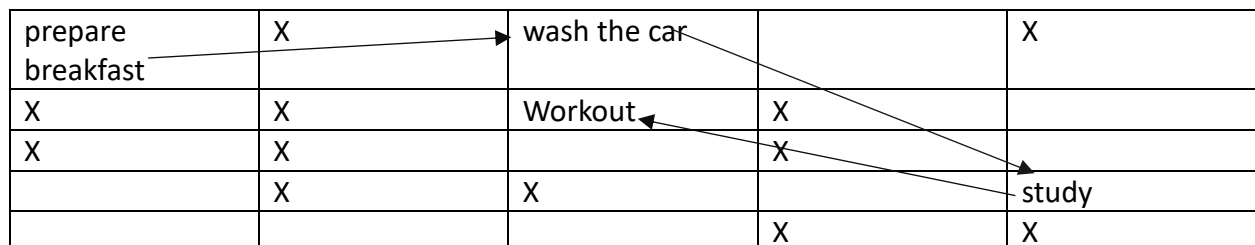
2- you may add more than 10 tasks so you eventually will need to transfer to another memory position with slots enough for new array.

So, it's a good work around but it's not the perfect solution linked lists solve this problem of adding items.

Linked List:

With linked lists your elements can be anywhere in the memory each element holds a value (which you stored in it) and the address to the next element in the list so the linked list will probably look like that:

prepare breakfast	X	wash the car		X
X	X	Workout	X	
X	X		X	
	X	X		study
			X	X



This way helps a lot with insertion all you need to do is save the address of the next element in the last element before it.

Searching for the element you want in a linked list could be a bit of a problem for example if we want to find workout you will have to first go to "prepare breakfast" then "wash the car" after that "study" and finally you will reach "workout".

With arrays this a bit different the array indexing starts from 0 not 1 so the first element is the element number 0 so in our case if wanted to find the third element a simple math will tell you it's the element number 2 this is called random access that makes the array a lot better in reading elements than linked list.

Deletions:

Lists are better also in deletions. It's as easy as changing what the previous element points to but in arrays you will have to move everything when an element is deleted.

	Array	Linked list
reading	$O(1)$	$O(n)$
insertion	$O(n)$	$O(1)$
deletion	$O(n)$	$O(1)$

It's worth mentioning that insertion and deletions will only take $O(1)$ only if you can access the position you want to delete or insert in that's why it's common practice to keep track of the first and last elements in the list.

Which is better?

It depends on the use case arrays are used a lot because they allow random access which is required a lot, but you must make the choice if your program does a lot of searches with little insertions and deletions array is better but if you do insertions and deletions a lot lists are better.

Selection sort:

Let's say that you want to sort your array ascendingly what the selection sort will do is that it'll start searching for the smallest element and add it to a new array and then you search for the second smallest element and add it to the new array and so on so if our array is like that {5,3,1,4,2}

It will be sorted like that.

- 1- {5,3,4,2}, {1}
- 2- {5,3,4}, {1,2}
- 3- {5,4}, {1,2,3}
- 4- {5}, {1,2,3,4}
- 5- {}, {1,2,3,4,5}

To determine the complexity of this algorithm let's see how it iterates (loops) over the array to sort an array of n elements you will have to loop over n elements n times which means $O(n*n)$

Which is $O(n^2)$.