Linked List vs Array

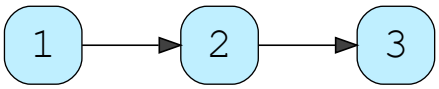
In this lesson, we'll discuss the motivation for a better data structure to represent linear data than arrays.

We'll cover the following

Why do we need a list

A major problem with arrays is that the size is fixed. For example, adjacency list representation of a graph is not possible with 2-D arrays. It's fine if you don't know what adjacency list is just yet, we'll discuss that in later chapters. But this is just an example of why we need arrays that are dynamic in size.

So, we create a linked list where each node, along with the value, stores a pointer to the next node.



Drawbacks

- Since memory is not contiguous, random access is not possible. Accessing any element is an $O(N)$ operation.
- Extra memory for storing a pointer. This is not relevant for competition.

Types

- Singly Linked List
- Doubly Linked List
- Circular Linked List

- Circular Linked List

Competitions

If you solve a problem using linked lists in a competition, you will have to implement linked lists singly or doubly, depending on the use case. Circular is used very rarely.

Now let’s discuss how to represent and perform operations on a list, starting with searching.