Pattern Sliding Window:

Data structures used on: Strings, Linked Lists, Arrays

Data structures used to solve: Hash Maps (more advanced problems)

When to use:

* Asked to find / calculate something among sub arrays or sub lists.
* When the questions talk about overlapping.
* Question usually has the word “sub array”, or “window”, or “longest”, or “shortest” in it.

Main Idea:

* You keep track of a virtual array, that has a beginning and an end. Could shrink or increase in size.
* Sometimes to keep track of stuff you will need to use a hash map.
* Keep a running value of something while going through an array.

Time Complexity:

* Usually O(n)

Pattern Two Pointers:

Data structures: Strings, Linked Lists, Arrays

When to use:

* When ever you have a sorted array/ when you could sort the array
* Find a set of elements that satisfy a rule
* Could be linked Lists

Main Ideas:

* Could sort the problem space
* Would always have to use two different pointers doing their own thing/ working together

Time Complexity:

* All over the place, sometimes it can be O(n^3) or it could be O(n). Really depends on the solution.

Fast Slow Pointers:

Data structures: Strings, Linked Lists, Arrays, Actual numbers (like some part equation)

When to use:

* When it asks you to find anything related to cycle, or circular, or linked list. Or something that asks you to find a cycle indirectly.

Main Ideas:

* It all boils down to one number / index / pointer moving slowly, and another moving twice as fast.
* The harder it seems the more likely is solution is going to be made up of solving several smaller problems.
* Most of the solutions start with the fast pointer moving twice the speed of slow pointer.
* Sometimes you will need to reverse parts of the linked list, and then un-reverse it again.

Toolbox:

* The fast slow pointer idea can is very versatile in what it can help you do:
  + Find the middle of the linked list
  + Can check if there is a cycle in a linked list
  + If you know the length of a linked list cycle (X), you can then find where the cycle starts by taking two pointers (one at the start, and one moved X away from the start). Then keep moving the two pointers forward till they end up pointing to the same node. That node will be the start of the cycle.
* To get the right most number from a digit use MOD 10
* To remove the right most number from a digit use / 10

Time / Space Complexity:

* Usually O(n) time complexity, and O(1) space complexity.
* But can also be O(log(n)) for equation-based questions
* But can also be O(n^2) for array based questions