Identification & Use-Case

of 3 Data Structures & Algorithms

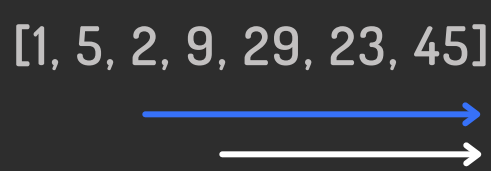
Mayank

01

Stack

Used mostly when Brute Force involves 2 loops like following ...

```
for(int i = 0;i < n;i++) {
  for(int j = i;j < n;j++) {
  // do some work
  }
}
```



Above array states that from wherever i (outer loop) starts, j(inner loop) will either cover left sub-array or right from i ... not the entire array

02

Stack

```
for(int i = 0;i < n;i++) {
  for(int j = i;j < n;j++) {
  // do some work
  }
}
```

Stack will surely be used if in the above 'Brute Force' if current jth value can remove it's immediate previous value/s which states that you want to remove last came value just before j ...i.e. Last In First Out

03

Stack

Some Leetcode Problems

- Next Greater Element to Right
- Find Most Competitive Sequence
- Longest Valid Parenthesis
- Maximum Width Ramp
- Largest Rectangle in Histogram
- 132 Pattern
- Remove K Digits

and many more...

04

RECURSION

- Used when you need to check all the possibilities for your test case

or

- Used when you need to make a choice for every element of your test case

- Here choice means either you can choose the element or discard that & move on.

05

RECURSION

- You wrote a function solve() to solve your test case t1
- Now, if same function can be reused to solve the sub test cases (smaller chunks of original test case), go for recursion

06

RECURSION

- To solve recursion first find the 'recurrence relation'
- Recurrence Relation means the pattern that makes the entire function repetitive in nature

- Ways to solve Recursion
    - IBH
      Induction, Base, Hypothesis
    - Choice Diagram etc...

will cover IBH & Choice Diagram approach in depth in coming posts...

07

Binary Search

- In Brute Force you must be using a loop (linear search)
- You need to see if the space where you are iterating is sorted or not
- Here 'sorted space' can be an array, a number line or some range of values etc...

Binary Search

- So if a 'linear search' is applied on a 'sorted space', go for 'Binary Search'

- Solving a 'Binary Search' problem requires to do following few steps
    - Finding lowest & highest limit where you will search your element
    - Keep lowest limit as high as possible & highest limit as low as possible to reduce the search time complexity

Binary Search

- Continuing steps to solve a Binary Search problem...
    - Now over entire sorted space, jump to 'mid' & decide which part of array (left or right) keeps your answer & discard the other half
    - To discard left half we do low = mid + 1
    - To discard right half we do high = mid - 1

- I already created a beautiful SlideDeck covering 'Binary Search' concepts in depth you can check out (link in comment)

Binary Search

Some problems involving 'Binary Search'

- Valid Perfect Square
- Valid Triangle Number
- Arranging Coins
- Capacity to ship packages within D days
- Koko eating bananas
- Allocate minimum number of pages
- Aggressive Cows
- Nth magical number

and many more...

Mayank

Software Engineer | StoryTeller

Code Smarter

I hope you found it useful

Follow for content related to DSA