careers N' OPTIONS
Careers N Options Services Pvt. Ltd.

DD/MM/YYYY

#45DAYSDSA #DAY1

1. Time & Space Complexity

- Time complexity of an algorithm quantifies the amount of time taken by a program to run as a function of length of the input. (Checks the running time of a program)
- Space Complexity of an algorithm quantifies the amount of space taken by a program to run as a function of a length of the input. It is directly proportional to the largest memory your program acquires at any instance during run time. (Checks the space/memory occupied) → linear of n
- If the program has less running time (Time complexity) and less space occupied (Space complexity) than that program is said to be efficient, fast, & reliable.

Big 'O' Notation.

- Any program can have the 3 possibilities

1. Worst Case [O (big oh) notation]
2. Best Case [Ω (big omega)]
3. Average Case [Θ (big theta)]

For example: Search

20, 10, 5, 100, 300, 17, 238

find 20 → Ω(1) } Avg Case → Θ(n)
find 238 → O(n) }

Comparison of functions

n	n^2	n^3	
n=1	1 unit	1 unit	1 unit	O(n) → F
n=2	2 unit	4 unit	8 unit	O(n³) → S
n=3	3 unit	9 unit	27	↓

$O(n) < O(n^2) < O(n^3)$

Fast Avg Slow

	n	log n
n = 1	1	0
n = 2	2	$\log_2 2 = 1$
⋮		
n = 1024	1024	10 → $\log_2 (2)^{10} = 10$

$O(\log n) < O(n)$

	Sqrt(n)	log n
n = 1	1	0
n = 2	$\sqrt{2}$	1
⋮		
	$\sqrt{n}$	log n
	$\log n^{\frac{1}{2}}$	log (log n)
	$\frac{1}{2} \log n$	log (log n)

$O(\log n) < O(\sqrt{n})$

[0, 1, 2, 3, 4, 5, 6, 7, 8]

2	5	7	10	21	27	50	71	85

low = 0
high = 9
mid 4 → 21.
high → = 4 + 1
~~high~~ = 3
low = 4 + 1
5
⇒ i == log n

Day2

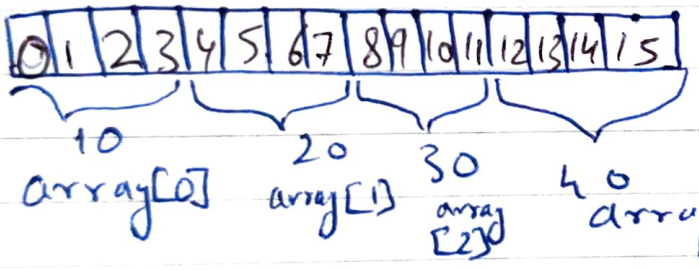
2] Arrays (with searching) Sorting

Arrays: It is a list of similar type

Example: {1, 2, 3, 4}, {'e', 'F', 'h', 'i'}

datatype arrayName[size];
int array[4] = {10, 20, 30, 40};

int -> 4 bytes, array[4] → 4×4 = 16



1th array of 4 size will take 16 bytes of memory

1. Linear Search

{12, 18, 20, 42, 8, 10}
key = 8

linear search will find the key in a linear manner

DD/MM/YYYY

```
import numpy as np

li = [ ]
size = int(input("Enter Size:"))
for i in range(size):
    x = int(input("Element:"))
    li.append(x)
```

Creates a numpy array with custom size by user.

```
arr = np.array(li)
key = int(input("Enter the key to search"))

# Binary Search and Linear Search
# 1. Linear Search
def Linear_Search(arr, size, key):
    for i in range(size):
        if arr[i] == key:
            return i
    return -1
# Calling linear search

Linear_Search(arr, size, key)
```

DD/MM/YYYY

2. Binary_Search

```
Def Binary_Search (arr, size, key):
    low=0
    high = size - 1
    for i in range(size):
    while low <= high:
        mid = int((low + high) /2)
        if arr[mid] == key:
            return mid
        elif arr[mid] > key:
            high = mid -1
        else:
            low = mid +1
    return -1

Binary_Search (arr, size, key)
```

DD/MM/YYYY

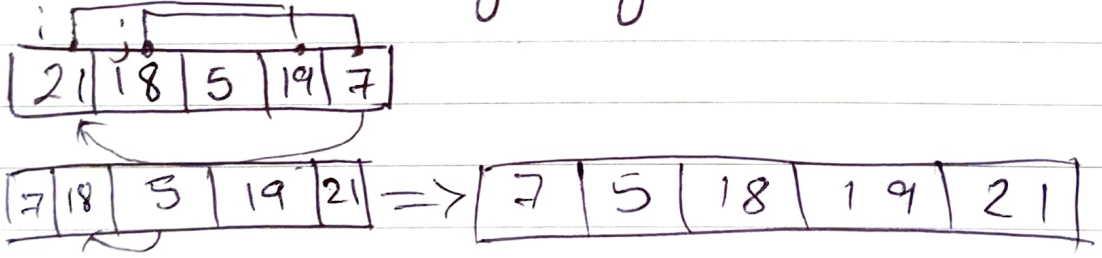
Day 3

3. Sorting in Arrays (Selection, Bubble, Insertion)

- Sorting means ordering the elements in ascending / decending order.

* Sorting Techniques

1. Selection Sort: Find the minimum element in unsorted array and swap it with element in the begining.

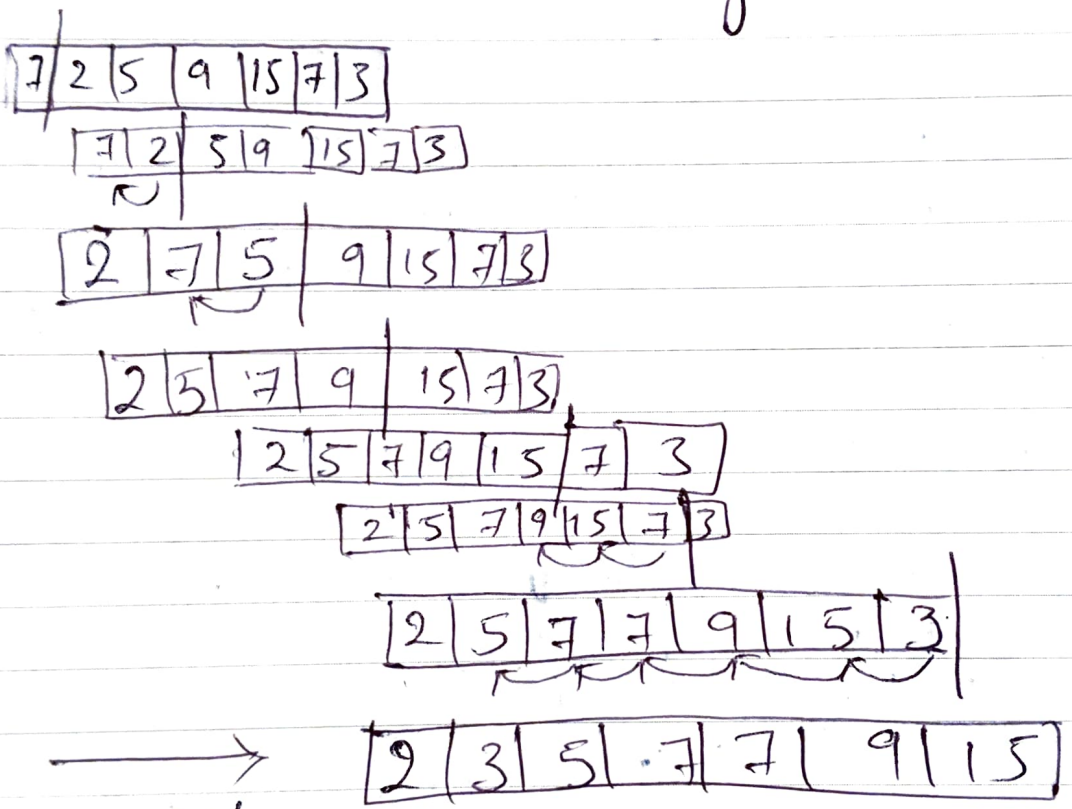


2. Bubble Sort: Repetedly swap two adjacent elements if they are in wrong order

12, 45, 23, 51, 19, 8
no yes NO yes yes
12, 45, 23, 51, 19, 8
12, 23, 45, 51, 19, 8
12, 23, 45, 19, 51, 8
12, 23, 19, 45, 51, 8

12, 19, 23, 45, 51, 8
12, 19, 23, 45, 8, 51
⋮
8, 12, 19, 23, 45, 51

3. Insertion Sort : Insert an element from unsorted array to its correct position in sorted array.



The sorted array

DD/MM/YYYY

Day 4: Array Challenges

(i) Max till i:

1	0	5	4	6	8
(0)	(1)	(2)	(3)	(4)	(5)

→ 1 1 5 5 6 8

— Approch: 1. Keep a variable mx which stores the maximum till 9^{th} element. 2. Iterate over the array and update, i.e. mx = (max(mx, a[i])

(ii) Subarray vs Subsequence

— Subarray: Continuous part of the array

1	2	0	7	2	0	2

Array

→ | 2 | 0 | 7 | 2 | Subarray

* Number of subarrays of an array with n elements = $^{n}C_{2} + n = n*(n+1)/2$

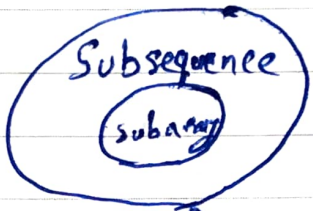
(iii) - Subsequence: A subsequence is a sequence that can be derived an array by selecting zero or more elements, without changing the order of the remaining elements.

* Number of subsequences of an array with n elements = 2^n

DD/MM/YYYY

Every Subarray is a Subsequence but every subsequence is not a Subarray

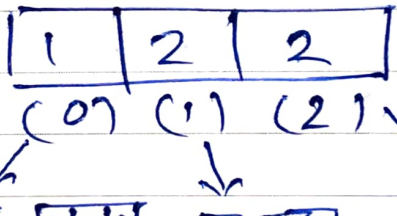
Venn diagram.



(iii) Sum of all Sub Arrays.

Problem: Given an array a[] of size n. Output sum of each subarray of the given array.

Example :-



Sub array: [1] [1,2] [1,2,2] [2] [2,2] [2]

Total subarrays = 6

Sum of subarrays = 1, 3, 5, 4, 2

Approach: Iterate over all the subarrays. Using Nested loop

```
for i in range(n)
    for j in range(i, n)
```

(iv) Longest Arithmatic Subarray
- Asked in Google kick start.

Problem: Array that contains at least two integers and the differences between consecutive integers are equal.
for example: (i) [9, 10] → 10-9=1
(ii) [3, 3, 3] → 3-3=0, 3-3=0, 3-3=0
(iii) [9, 7, 5, 3] → 9-7=2, 7-5=2, 5-3=2

Constraints: (i) Time limit: 20 seconds per test
(ii) Memory limit: 1 GB.
$1 \leq T \leq 100$, $0 \leq A_i \leq 10^9$

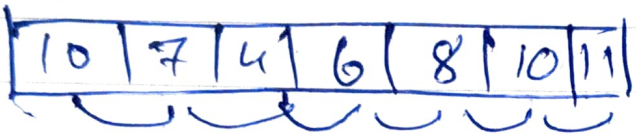
1 sec = 10^8 operations (approx)
2 sec = 2×10^8 operations (approx)

Time Complexity is generally calculated by number of times n is iterated

* We have to find the maximum length of Arithmetic subarray of an Array

DD/MM/YYYY

Sample Test Case



maximum Arithmetic subarrays are of 2 differences.

4 6 8 10

So length = 4

Approach and Intuition:

* Loop over the array and find the answer:

Maintain the following variables:

1. Previous Common Difference (pd)
2. Current Arithmetic Subarray length (curr)
3. Max arithmetic subarray length (ans)

2 cases

pd == A[i] - A[i-1]
Current ans increases by 1
ans = max(ans, curr)

pd != A[i] - A[i-1]
Update current ans to 2.

(v) Record Breaker – (Google Kickstarts)

N Consecutive days. → No. of visitors on the i^{th} day is V. A day is Record breaking if it satisfies 2 conditions

1. The number of visitors on the day is strictly larger than the number of visitors on each of the previous days
2. Either it is the last day, or the no. of visitors on the day is strictly larger than the no. of visitors on following day

Note:- The very first day could be a record brk

2 Cases

Strictly greater than all the previous value (1)

Strictly greater than following value. (2)

Sample Test Case

1	2	0	7	2	0	2	2
↑	↑	↑	↑	↑	↑	↑	↑
x	✓	x	✓	x	x	x	x
(2)		(1) (2)		(1)	(1) (2)	(1) (2)	(1)

Brute force: 1. Iterate over all the elements and check if it is record breaking day or not, To check if a[i] is a record breaking day, we have to iterate over a[0], a[1], ... [a[i-1].

Time Complexity for this operation: O(n)

Overall Time Complexity: $O(n^2)$

Optimised Approach:

If we can optimise step 1, then we can optimise overall solution.

for step (1) : we need to check if a[i] > {a[i-1], a[i-2], ... a[0]}, which is same as 'a[i] > max(a[i-1], a[i-2].. a[0])

for this, we can keep a variable mx, which will store the maximum value till a[i]. Then we ~~can~~ just need a check,

a[i] > mx,

a[i] > a[i+1], {if i+1 < n}

and update mx, mx = max(mx, a[i])

So step(1) time complexity reduces to O(1)

Overall time complexity: O(n)

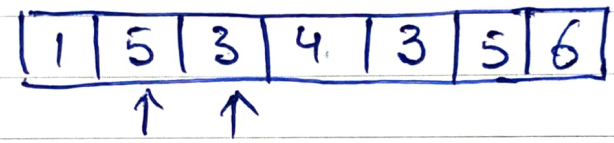
DD/MM/YYYY

Day 5 Arrays - Q's asked by Top MNC's

Q1. First Repeating Element

Problem: Given an array arr[] of size N. The task is to find the first repeating element in the array of integers, i.e, an element that occurs more than once and whose index of first occurence is smallest.

Given Array:



In above array 5 and 3 are repeating but 5's index is the smallest so ans = 1

Q.2. Subarray with given sum:

Problem: Given an unsorted array A of size N of non-negative integers, find a continuous subarray which adds to a given number S.

Example: Input → N=5, S=12
A[] = {1, 2, 3, 7, 5}

Output: 2 4 because the sum of elements from 2nd position to 4th position is 12 i.e {2, 3, 7}
2 + 3 + 7 = 12

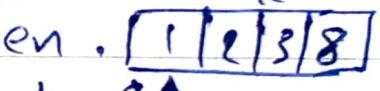
Note: index starts from 1 in this Q's

Brute force: Find sum of all possible sub arrays. If any of the sum equates to S, output the starting and ending index of the subarray.

Time Complexity: $O(n^2)$

Optimized: 2 Pointer Approach

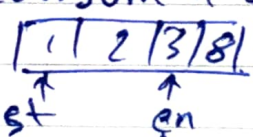
(i) Keep 2 points st and en, and a variable currSum. sum from st to en.



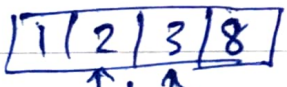
where, given S = 5, currSum = 1

(ii) increment en till currSum + a[en] > S

now, currSum = 6



(iii) Start increasing st until currSum <= S



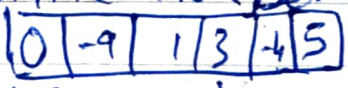
now, currsum = 5 = S.

Time Complexity: $O(n)$

Q.3. Smallest Positive Number.

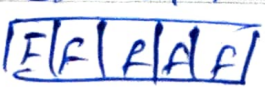
Problem: You are given an array a[] of N integers including 0. The task is to find the smallest positive number missing from the array

Example:



Here first positive missing number is 2

Base Idea: Build a boolean check[] array that will depict the element x is present in the array or not.



Iterate over the array and mark non-negative a[i] as True.

Day6 Subarray Challenges.

1. Max Sum of Subarray: [-1 | 4 | 7 | 2]

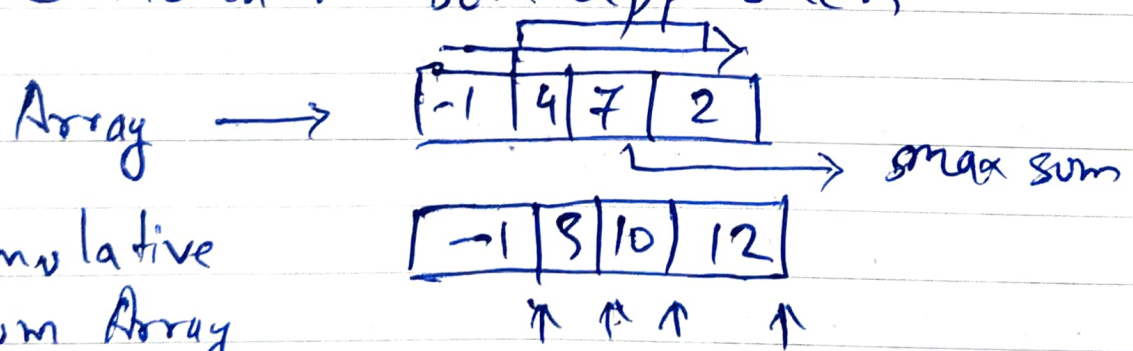
* Possible Subarrays: [-1], [-1, 4], [-1, 4, 7], [-1, 4, 7, 2], [4], [4, 7], [4, 7, 2], [7], [7, 2], [2]

* Max sum of Subarray is [4, 7, 2] = 13

* Brute force Approach: To create 3 nested loop and add the sum of subarrays elements and return the max sum subarray but this way the time complexity will be $O(n^3)$.

* Optimized Approach:

Cumulative sum approach



More Optimized: Kadane's algorithm

Array [-1, 4, -6, 7, -4]

Curr sum [-1, 4, -2, 7, -3]

0 4 0 7 3

2. Maximum Circular Subarray Sum

Case1 {-1, 4, -6, 7, 5, -4}

No need do wrap 12

Case 2 {4, -4, 6, -6, 10, -11, 12}

We need do wrap 2 2

Max subarray sum = Total sum of array - Sum of non-contributing elements.

apply Kadane's Algo on Reverse array %

{-4, 4, -6, 6, -10, 11, -12}

max

Now remove the max from old array

3. Pair Sum Problem : Check if there exists two elements in an array such that their sum is equal to given k

0	1	2	3	4
5	7	9	12	15

K = 28

Pair → 1, 3

Brute force : Using two nested loop to create a pair.

Optimized: 2 Pointer Approach

2	4	7	11	14	16	20	21

low → + ← high

K = 37

```
while low < high:
    if a[low] + a[high] == K
      True
    elif a[low] + a[high] > K
      high -= 1
    else:
      low += 1
```


Day 7 Tu

1D Arra

2D Arr

arr 3x3

Creating

arr =

for i

for

ar

Printing

for i

for

prin

Output:

n=3, 1

m=3 4

7

Day 7 Two-Dimentional Array

1D Array → | 1 | 2 | 3 | [1, 2, 3]

2D Array → arr 3×3 ←

1	2	3
4	5	6
7	8	9

or [[1, 2, 3
4, 5, 6
7, 8, 9]]

Creating 2D Array in Python using user

```
arr = [ ], n = int(input( )), m = int(input( ))
for i in range(n):
    for j in range(m):
        a = [ ]
        num = int(input("Enter number:"))
        a.append(num) # [no][no][no][no]
    arr.append(a) # row 0, 1, 2,
```

Printing the 2D array

```
for i in range(n)
    for j in range(m)
        print(arr[i][j], end = "  ")
    print( )
```

Output:

n=3, m=3

```
1  2  3
4  5  6
7  8  9
```

☆Spiral Order Matrix Traversal

What is spiral Order matrix?

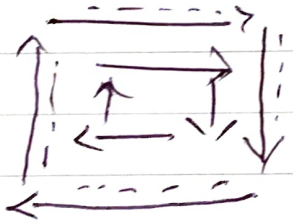
1 → 2 → 3 → 4 → 5 → 6 → 7
20 → 7 ↓
8 ← 9 ← 10 ← 11 ← 12 ← 13 ← 14 ← 15 ← 16 ← 17 ← 18 ← 19 ← 20

(i) It will print in a spiral formal
(ii) The spiral will repeat

How to print Spiral Matrix?

n * m size Matrix (4 variables needed)

(i) row_start +1
(ii) Column_end - 1
(iii) row_end -1
(iv) Column_start +1



while (row_start <= row_end) and (col_start <= col_end)

→ loop through col_start to col_end
row_start incremented

↓ loop through row_start to row_end
col end decremented

← loop through col_end to col_start
row_end decremented

↑ loop through row_end to row_start
col_start incremented

Day 9: Character Arrays

Arrays of Character

R	A	H	U	L

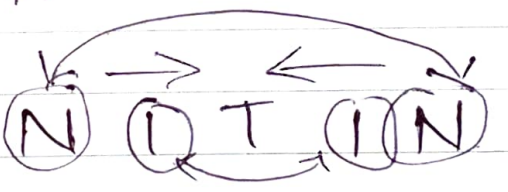
Character array ends with '\0' Null
So that the compiler understands that the array is complete and the end is that null element, for example

R	A	H	U	L	\0

↑ Null

Declaration: Char arr[100] = "apple"
But in python its dynamic
for eg: arr = "apple", no need to define the size.

(1) Check palindrome:



(2) Largest word in a sentence.

"Do or die

i i

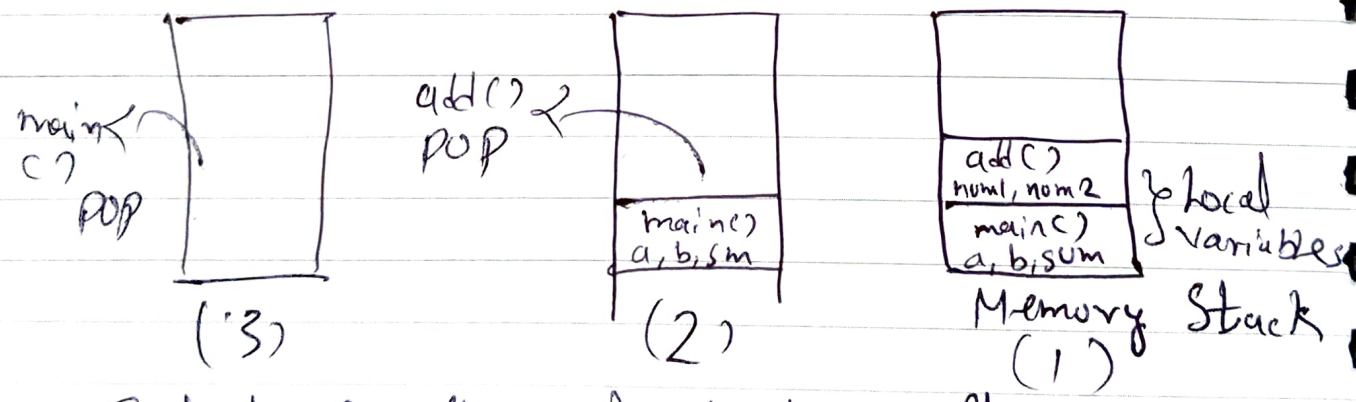
ans = maxlength

Day 11 Stack & Heap

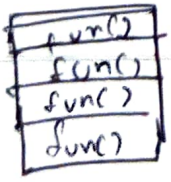
Fixed Dynamic

*Stack: A stack is an abstract data type that serves as a collection of elements, with two main principal operations: Push, which adds an element to collection, and Pop, which removes the most recently added element that was not yet removed.

Stack in memory is a memory usage mechanism that allows the system memory to be used as temporary data storage that behaves as a first-in-last out buffer.



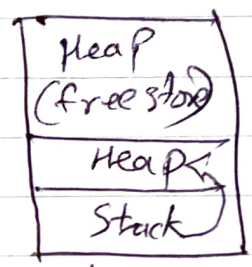
Stack Overflow: A stack overflow is an undesirable condition in which a particular computer program tries to use more memory space than the call stack has available.



The size of stack frame is fixed that's why if function call is greater then the stack size it overflows.

* Heap: A heap is an area of pre-reserved computer main storage (memory) that a program process can use to store data in some variable amount that won't be known until the program is running.

- Heap is a dynamic memory and it is called only when called
- Size is not fixed methods
- In python functions and ~~object~~ variables are created in stack memory. whereas Objects and instance variables in Heap memory.



Day 10 Strings (Basics)

Strings are the collection of characters we can say string are array of characters. In python storings are defined easily by keeping everything inside quotes " ".
For example : str = "Rahul"

String operations.

(1) Concatenation of strings
```
str1 = "fam" , str2 = "ily"
str1.append(str2)  ---> only for c++
print(str1) # Output : family
    or .join() for python.
print(str1 + str2)
```

(2) clearing the storing using clear()
but again clear() only works on C++ for python we have to reassign empty string example : str1 = "".

(3) Comparision of strings.
```
str1 == str2  # False
str1 is str2  # False
```

(4) Remove something from string
use slicing example str = "nincompoop"
print(str[:3] + str[6:]) # nin Poop

or replace()
for example: print(str.replace("com", ""))

(5) find(): It finds the index value of a string which is first encountered in it
example: s1.find("com") # 5.

(6) Insert: In Python the strings are immutable so we will have to create new string in which we will insert are new word or string. We use slicing to accomplish this
for example:
```
line1 = "Kung Panda"
line2 = "Fu"
index = line1.find("Panda")
line3 = line[:index] + line2 + [index:]
print(line3) # Kong fu Panda
```

(7) Sorting a string: Use sorted() to sort the string;
for example:
```
a = "xyzfdgdh"
a1 = sorted(a) # [d, d, f, g, h, x, y, z]
a2 = "".join(a1) # converts list to str
print(a2) # ddfghxyz
```

Day 11 Questions on Strings.

1. Convert lowercase string to uppercase and viceversa.

Solution: ASCI value of characters can be used to convert them.
step 1. find the ASCI difference between smaller and uppercase
step 2. Subtracting or adding the value from the character depending on the result.
- this way at the end we get the converted string.

2. Form the biggest number from the numeric string.
- Using sort method to sort them in decendig

3. Maximum number occuring in string
- Use dictionary and max
- We have to find the maximum occuring of a string.

Day 12 Recursion

Def: When a function call itself to make the problem smaller.

For eg: Sum till n.

Traditional Sum till n = n + n-1 + n-2 + n-3 ... +1

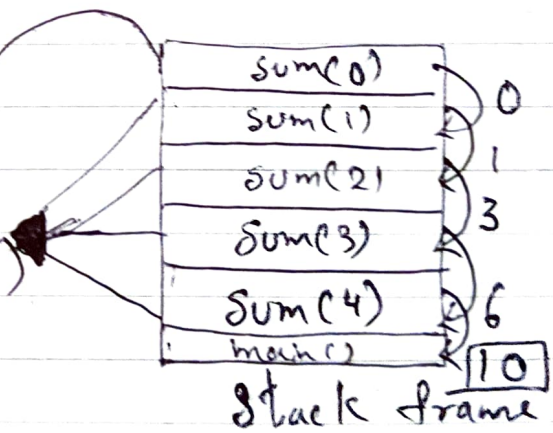
Recursion: Sum till n-1 = n-1 + Sum till n-2

Sum till n-2 = n-2 + Sum till n-3

⋮ ⋮

Q.1 Sum till 0 = 0

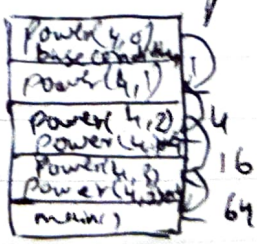
```
def sum(n):
    if n == 0:
        return 0
    prev_sum = sum(n-1)
    return n + prev_sum
```



Q.2. Calculalate n raised to power of p

$n^p = n * n * n \ldots$ p times

also $n^p = n * n^{p-1}$



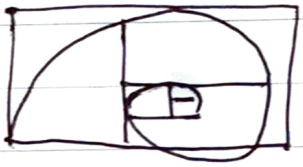
Day 13 → Question on Recursion.

Q.3. Factorial of number (Recursion)

$n! = n * n-1 * n-2 * \dots * 1$

$n! = n * (n-1)!$

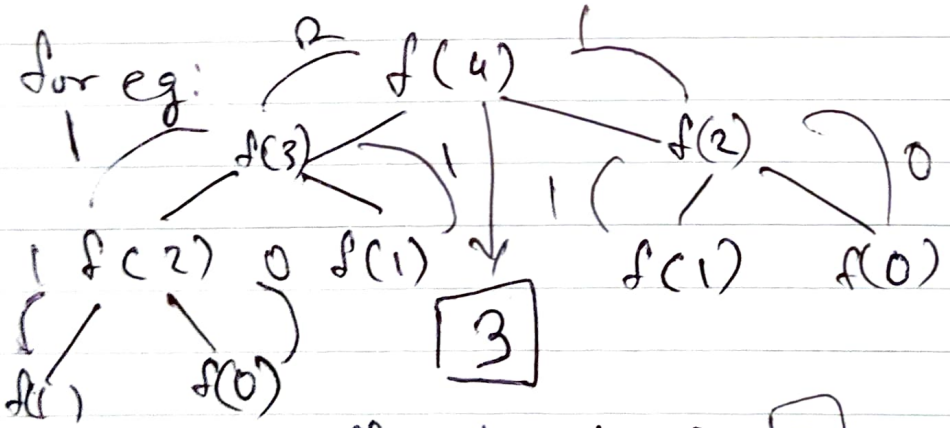
Q.4. Fibonacci number.



0, 1, 1, 2, 3, 5, 8, 13 ...

$fib(n) = fib(n-1) + fib(n-2)$

for eg:



0, 1, 1, 2, 3

Q.5 Print number till n for decreasing order

(i) print (n)
(ii) num_dec (n-1)

and vice versa for increasing

Q.s on Recursion Day 13

Q. Find the first and last occurence of a number in an array

1. First Occurence

for Ex: {4, 2, 1, 2, 5, 2, 7}

Problem: find 2

Steps: (1) Traverse through array
(2) If found print the index else
(3) traverse recursively

{4, 2, 1 2, 5, 2 7}

2. Last Occurence

Steps (1) Traverse through array
(2) If found then traverse till end of array
(3) print the last occurence.

Most famous Questions of Recursion:

Q.1. Reverse a string using recursion

→ "bino d" Print "b"
"ino d" Print "i"
"no d" Print "n"
"o d" Print "o"
"d" Print "d"
"" Return

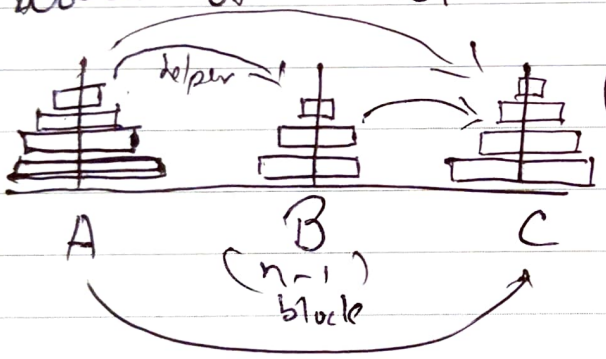
Steps (i) Calling for the rest of string
(ii) Print first character of string.

Q.2 Replace Pi with 3.14 in string.

for eg: "pippxxppipixipi"

⟶ "3.14pp xx 3.14 ixi 3.14"

Q3. ~~T~~ower of Hanoi



Conditions:

① Only one block can be placed

② smaller block cannot be placed before bigger block

Q.4. Remove all duplicates from the string

for eg: "aaaa bbbeeee cddd"

"abecd"

Q5. Move all x to the end of the string

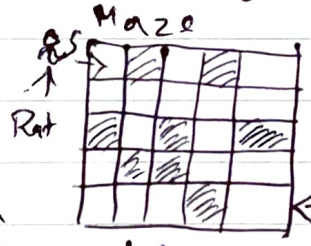
for eg: "axx bdxce fxhix"

to "abdcefhxxxx"

Day 14 Backtracking

Defn: Backtracking is an algorithmic technique for solving recursive problems by trying to build every possible solution incrementally and remove those solutions that fail to satisfy the constraints of the problem at any point of time.

Ex.s Rat in Maze



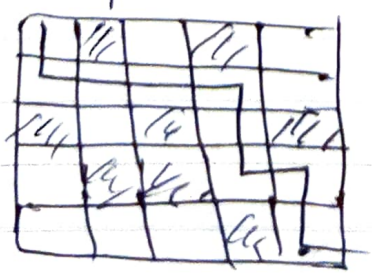
→ {1 0 1 0 1 2D
1 1 1 1 1 array
0 1 0 1 0}
← E

Conditions: (i) Rat can only move in 2 directions down & right
(ii) find the path of start and end

Input {1, 0, 1, 0, 1
1, 1, 1, 1, 1
0 1, 0, 1, 0
1, 0, 0, 1, 1
1, 1, 1, 0, 1}

Output: {1, 0, 0, 0, 0
1, 1, 1, 1, 0
0, 0, 0, 1 0
~~0, 0, 0, 1, 0~~
0, 0, 0, 1, 1
0, 0, 0, 0, 1}

1 for the path, 0 for blocks.

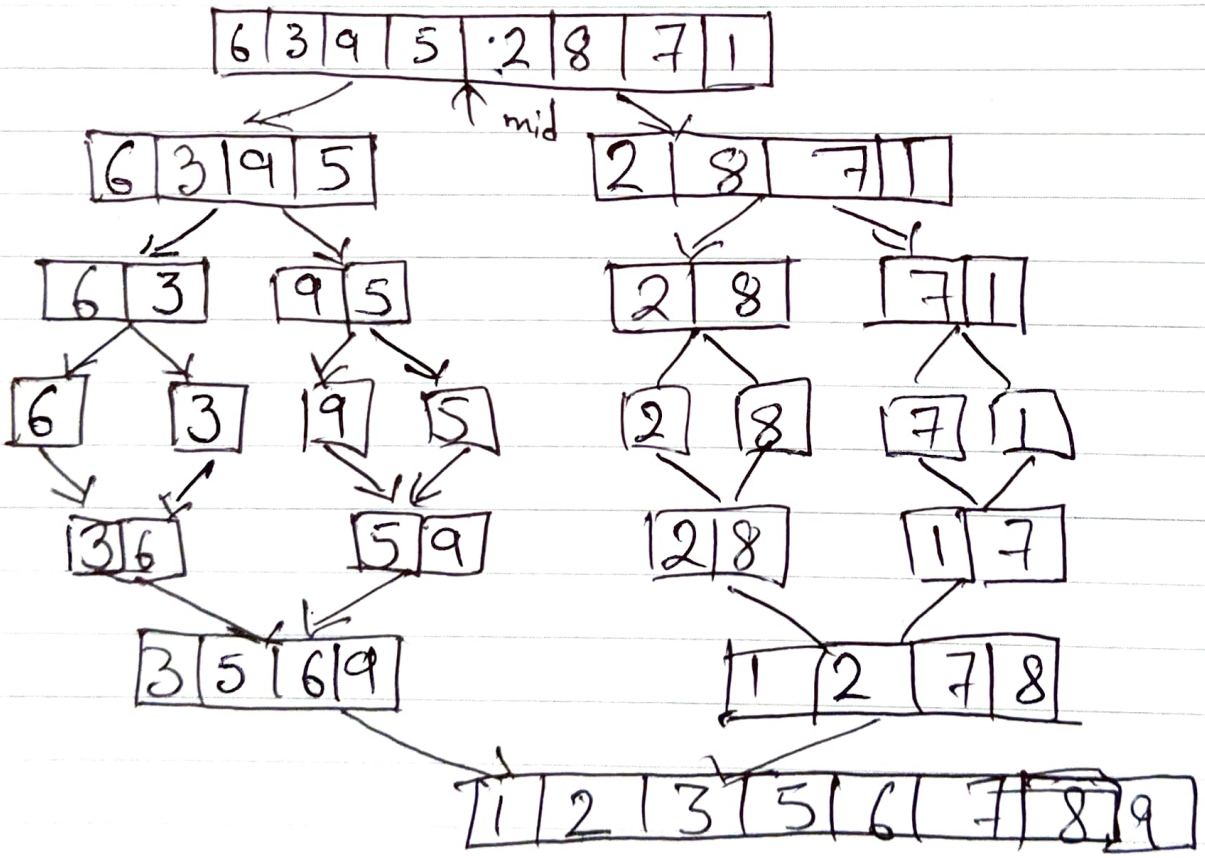


DD/MM/YYYY

Day 15 Merge Sort

A divide and conquer algorithm.

Divide the ~~unsorted~~ given array into single unit recursively and merge them.



careers N' OPTIONS
Careers N Options Services Pvt. Ltd.

Pivot - Pivot is an element of an array or matrix, which selected first by an algorithm.

Day 16 Quick Sort

It is also follows the divide and Conquer algorithimic technique.

The Quick Sort algorithm works on three conditions.

(i) (10) 80 90 60 30 20
(ii) 6 3 5 4 2 1 (9)
(iii) 4 6 7 (10) 16 12 13.

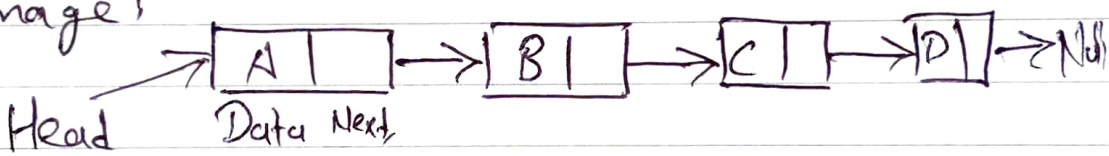
In above example the marked elements are sorted because the first element is the smallest, the second one is the largest and in case of third one it is surely smaller than the right side of the array and large. than the left side of array.

Quick Sort: | 6 | 3 | 9 | 5 | 2 | 8 | 7 | 1 |

1. Choose pivot element (usually last or random)
2. store elements less than pivot in left subarray
   store elements greater than pivot in right subarray
3. Call quicksort recursively on left subarray
   call quicksort recursively on right subarray

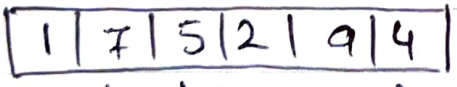
Day 17 Linked List

Linked list is a linear data structure also known as sequential data structure, which are connected together via links. The elements of linked list are not stored at contiguous memory locations. The elements in a linked list are linked using pointers as shown in the below image:



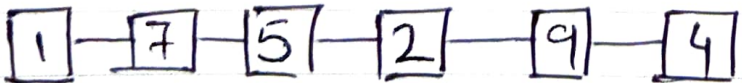
In simple words, a linked list consists of nodes where each node contains a data field and a reference (link) to the next node in the list.

Linked list is very different from array



→ Array

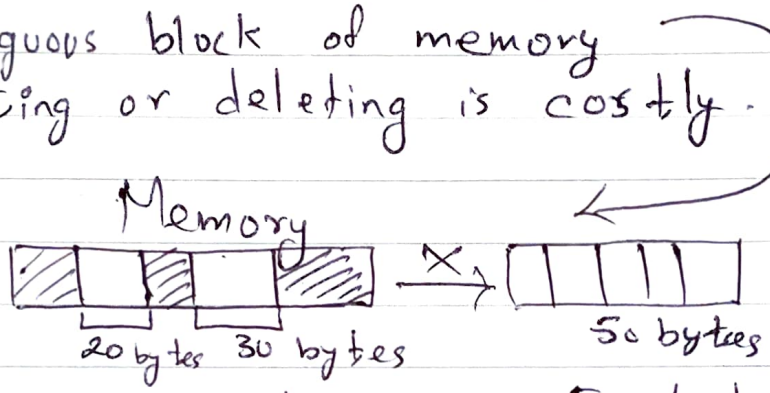
Single block of memory with partitions



→ Linked List

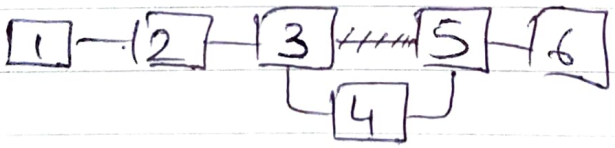
Multiple blocks of memory linked to each other.

* Limitations in arrays
  - Fixed size
  - Contiguous block of memory
  - Inserting or deleting is costly.



for eg: we cannot assign 50 bytes of memory because in arrays we need contiguous block of memory.

* Properties of Linked List
  - Size can be modified
  - Non-contiguous memory
  - Insertion and deletion at any point is easier.



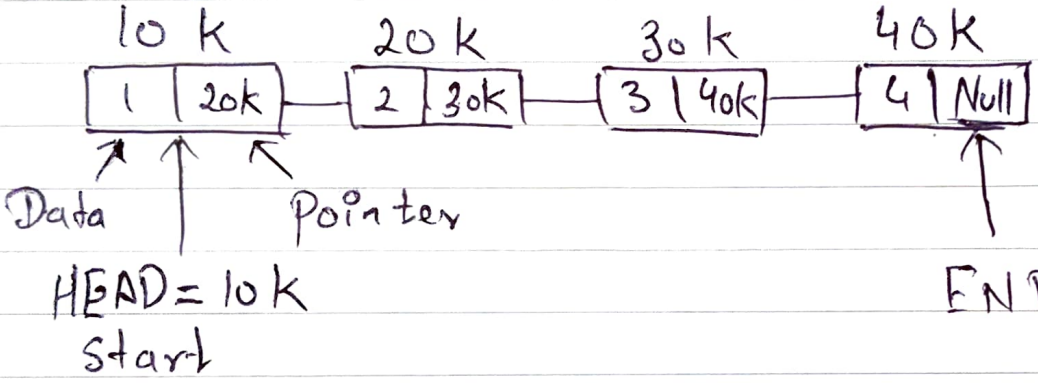
* Structure of Linked list

NODE: | Data | Next |
| 10 | Pointer |

A node is a collection of two-sub elements or parts

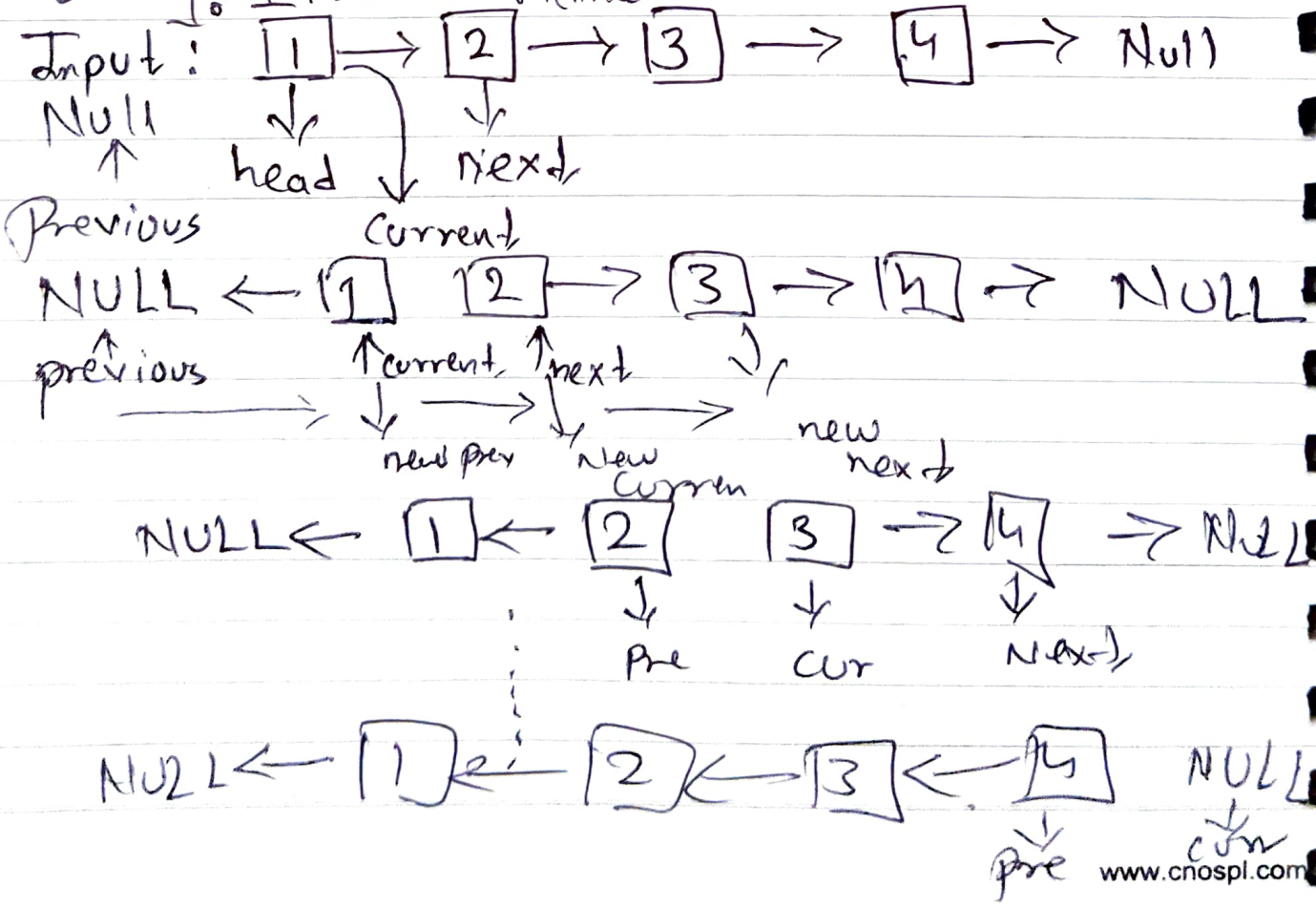
A data part that stores the elements and a next part that stores the link to the next node. A linked list is formed when many such nodes are linked together to form a chain.

Example of a Node | Data | Next |

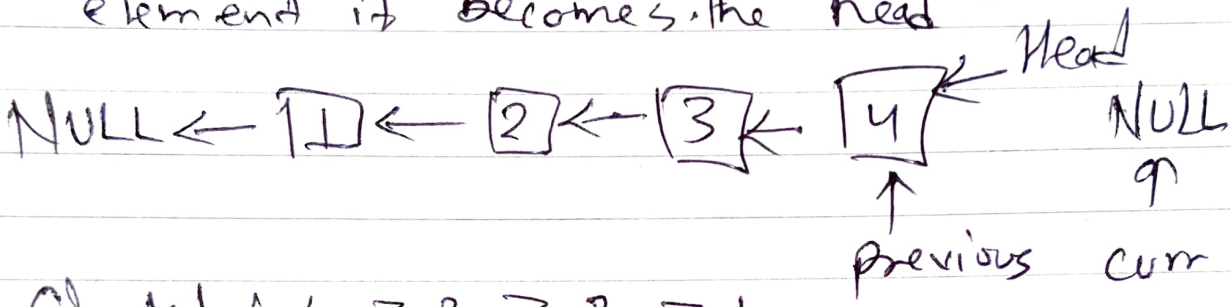


Day 18 Reverse a linked list

1. Iterative Method



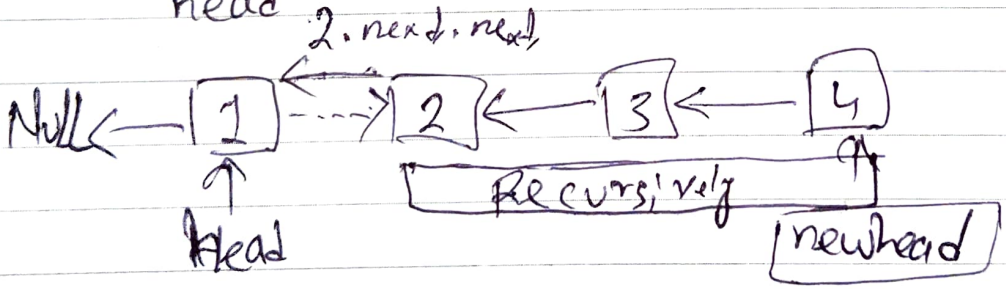
When previous traverse through n-1 element it becomes the head



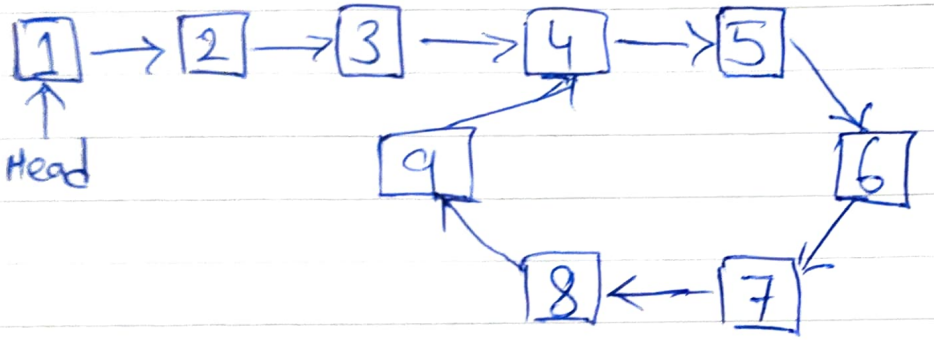
Ouput : 4 → 3 → 2 → 1

2. Recursive Method

head

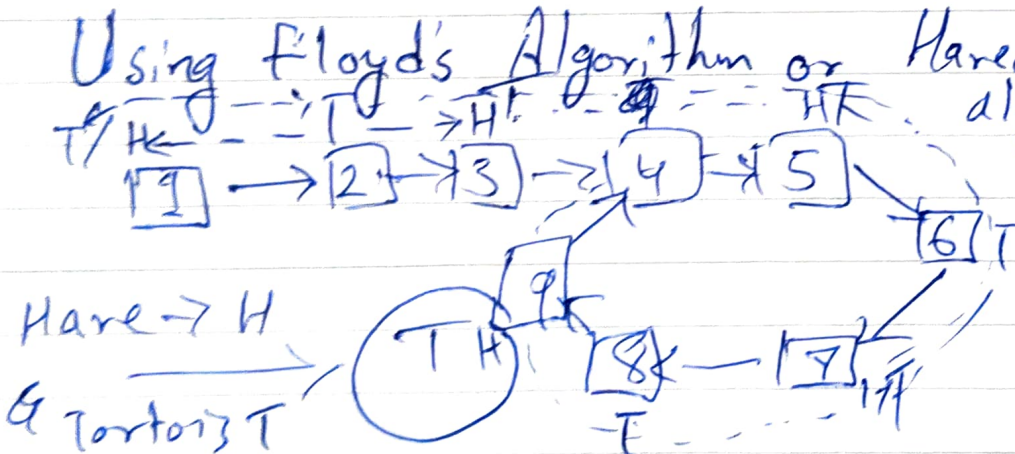


Day 19 Detection and Removal of Cycle in Linked List



How to detect cycle in a linked list?

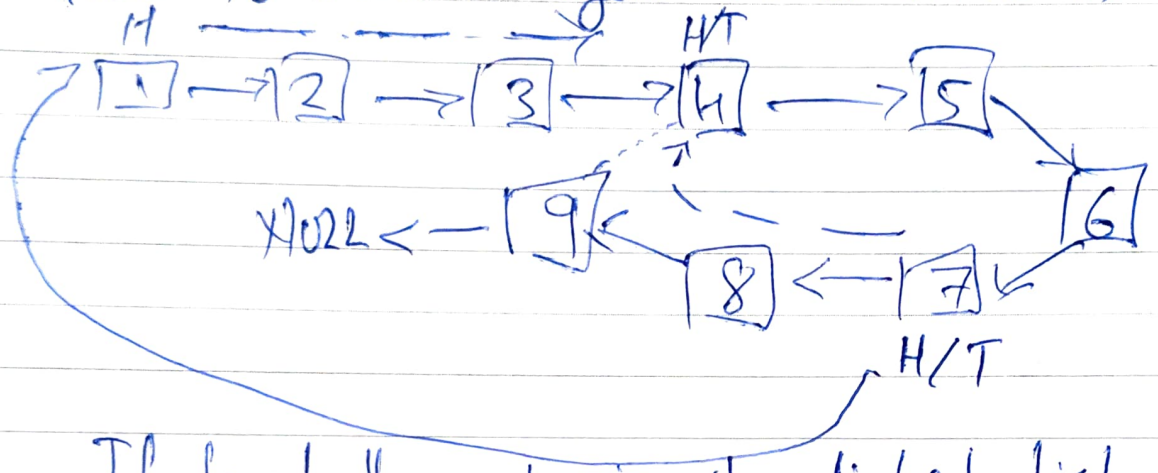
Using Floyd's Algorithm or Hare & Tortoise algorithm



So this algorithm use two pointers the Tortoise is the slow one using one step and the hare is the fast one using two step if the 2 pointers meet at same node it will confirm the cycle in linked list is detected

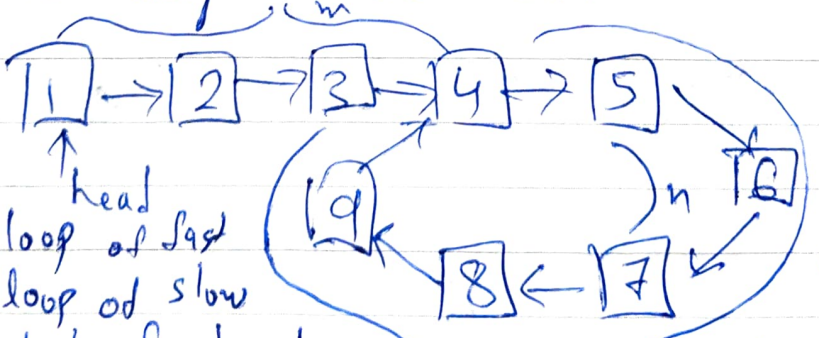
DD/MM/YYYY

How to remove cycle in a linked list



If found the cycle in the linked list from the point where two points meet send back one pointer to head and increment both simultaneously by 1 step and again where ~~they meet will be our starting point~~. there next pointers meet. The second pointer will be assigned to null.

Explanation:



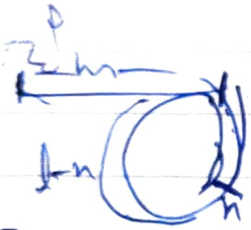
j is the loop of fast
i ... loop of slow

Distance moved by fast ptr = $m + n + l * j$

Distance moved by slowptr = $m + n + l * i$

Distance move by fast ptr = 2x Distance moved by slow ptr

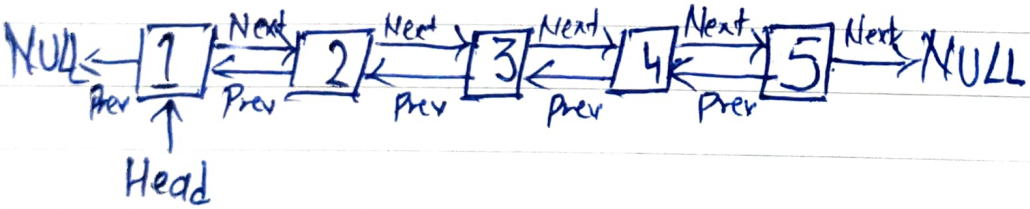
$$m + n + l + j = 2 * (m + n + l + i)$$
$$m + n = l * (j - 2*i)$$
$$m = l * (j - 2*i) - n$$



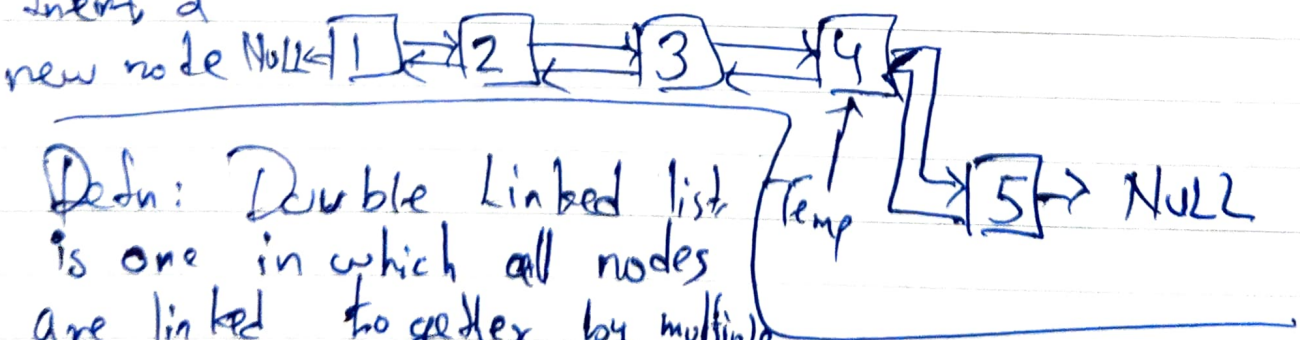
Day 20

Doubly Linked List

Node Structure

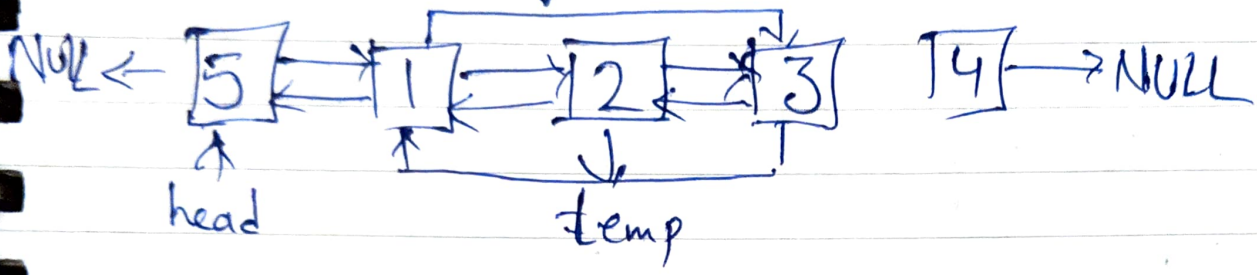


Insert a new node



Defn: Double Linked list is one in which all nodes are linked together by multiple links which help in accessing both the successor and predecessor node. It contains 3 things data, next and previous to create a doubly linked list structure.

Deletion in doubly linked list



DD/MM/YYYY

Day 21

STACK

(Linear Data Structure)

Stores a list of items in which an item can be added to or removed from the list only at one end.

It is based on LIFO mechanism.
i.e Last In First Out.

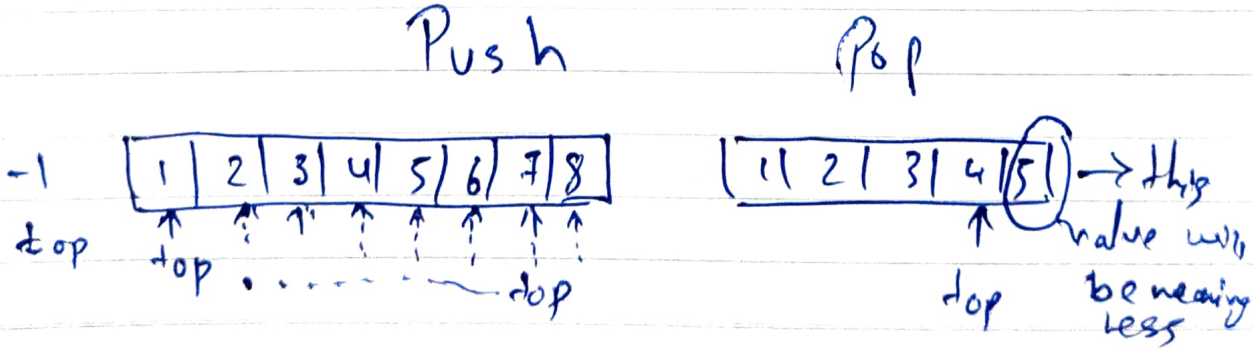
* Operations O(1)

push (x) : It will insert element 'x' on top of Stack.

pop() : It will remove the topmost element.

top() : It will return the top most value of the stack

empty() : It will check if the stack is empty or not.



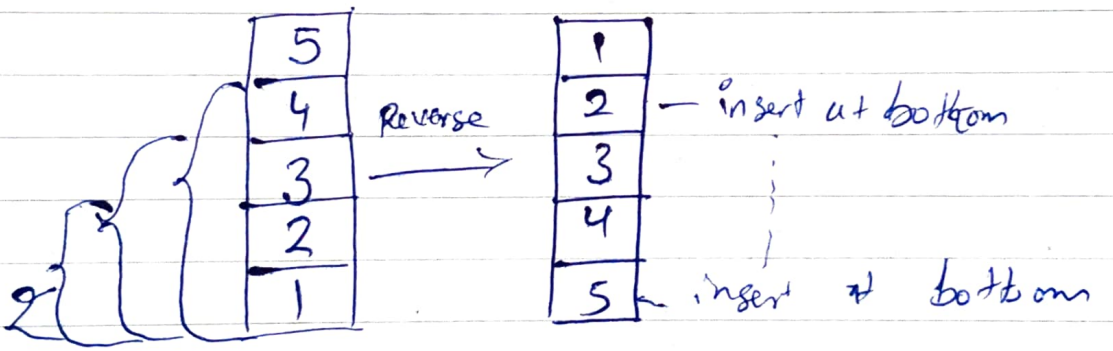
DD/MM/YYYY

Day 22 Reverse a sentence using stack.

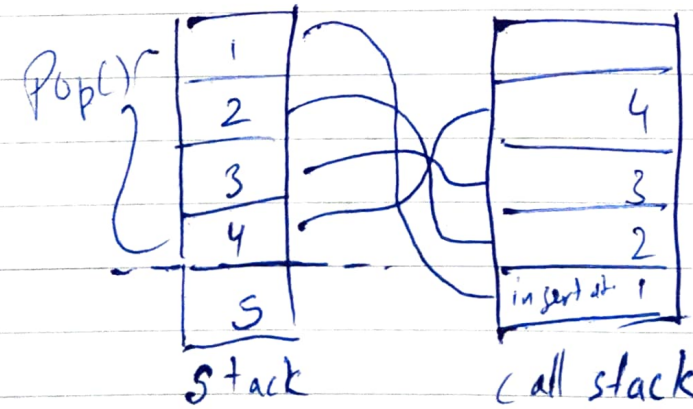
Ex.: Hey, how are you doing?

Output: doing? you are how, Hey,

Reverse a stack



Insert at Bottom



Day 23 Infix, Prefix, Postfix expressions.

(i) Infix expression <operand> <operator> <operand>

2+3
a*c+d
(8/2)-5
} expression

Precedence		
1	()	L → R
2	^	R → L
3	*, /	} L → R
4	+, -	

How to evaluate? 4 * 2 + 3

BODMAS ✓ (4*2)+3 4*(2+3)

(ii) Prefix expression
Polish notation <operator> <operand> <operand>

((4 * 2)+3)
infix

+ * 4 2 3
Prefix

(5^(6/3))
human readable

-5/63
Computer readable

← ← ←
Right to left

(iii) Postfix expression <operand> <operand> <operator>

Reverse polish notation

((4 * 2) + 3) 42 * 3 +

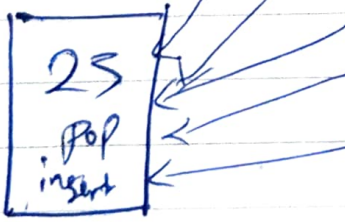
(5 - (6 / 3)) 563 / -

→ → → →

Left to Right

Prefix expression evaluation

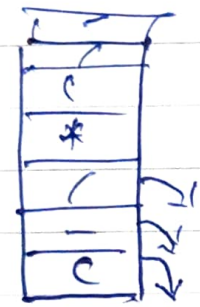
- + 7 * 4 5 + 2 0 ←



Stack

op1	2	op. 4	7	27
	+	*	+	-
op2	0	5	20	2

Day 24 Infix to Postfix



Stack

if operand
print

id '('
push to stack

if ')'
pop from stack and print until '(' is found.

if operator
pop from stack and print until an operator with less precedence is found

(a - b/c) * (a/k-1)
Infix

abc/-ak/f-*
postfix

Infix to Prefix

(a - b/c) * (a/k-1) reverse

(1-k/a) * (c/b-a)

Than same as Infix to postfix

1ka/-cb/a-* reverse

*-a/bc-/akl

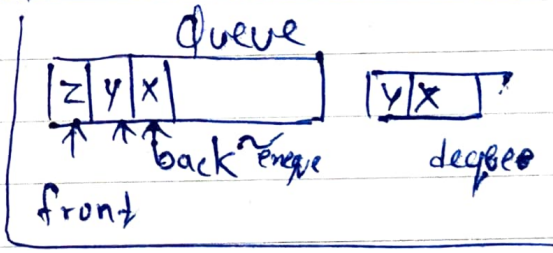
Day 25

Queue
(Linear Data Structure)

Stores a list of items in which an item can be inserted at one end and removed from the other end only

It uses FIFO principles to handle data
FIFO - First In First Out.

* Operations in Queue



(i) enqueue (x):
Similar to push in stack the enqueue operation insert new element on back.

(ii) dequeue () : Similar to pop and applies on front.

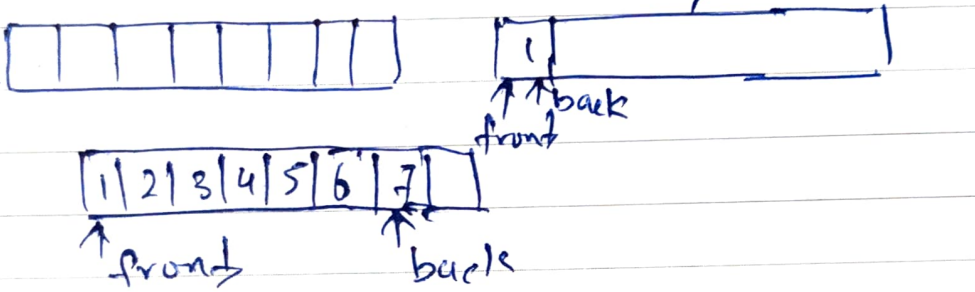
(iii) peek() : It returns the value of front pointer.

(iv) empty(): Returns that the queue is empty or not.

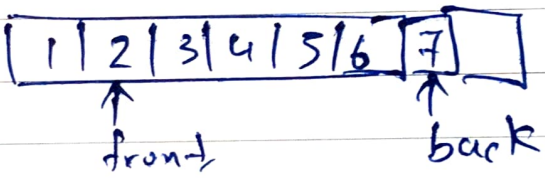
DD/MM/YYYY

Array Implementation:

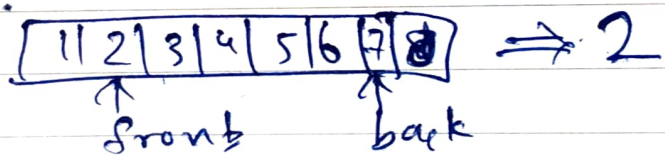
(1) front back



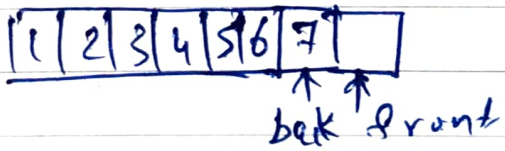
(2) Dequeue()



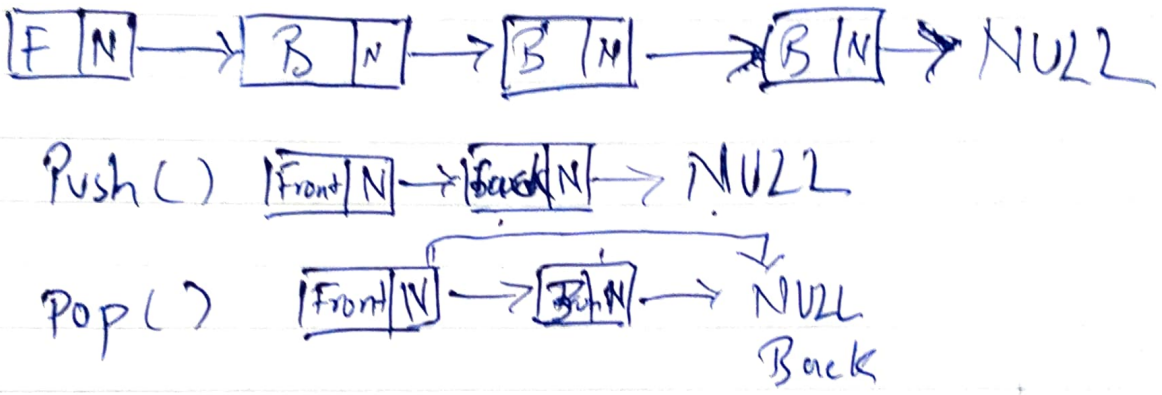
(3) peek()



(4) Empty()



Day 26 Linked List Implementation of Queue.



Day 27 Queue Implementation using Stack

2 Stack approach

stack1 = [ ], stack2 = [ ]

push() stack1.append(x)

but pop() stack1.pop() ⇒ stack2.append()
stack1.pop
stack2.pop()

Recursive Approach → Push is same

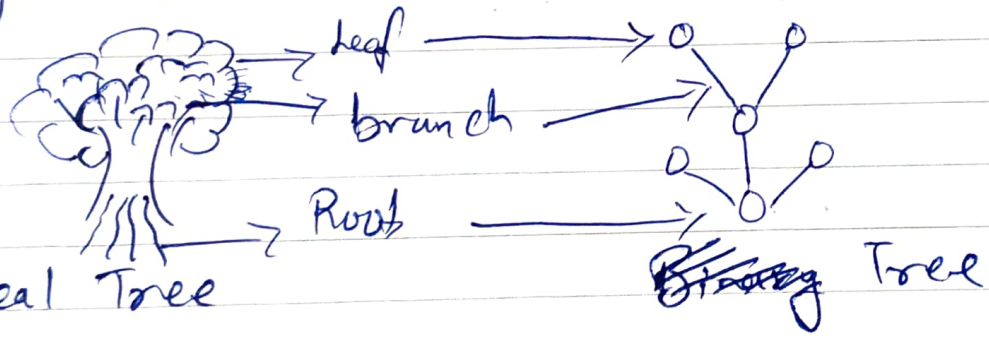
stack [ ]

for pop() if stack is not empty
recursively pop the top and store it temporarily
and then return pop the topmost element

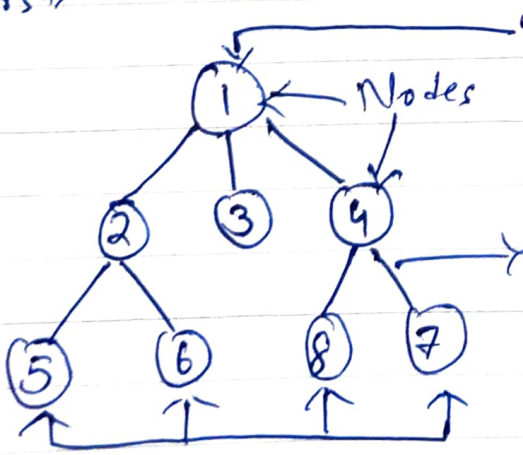
DD/MM/YYYY

Day 28 Binary Tree (Non-Linear Data Structure)

Def: A Binary tree is a tree data structure in which each node has at most two children, which are referred to as the left child and the right child.

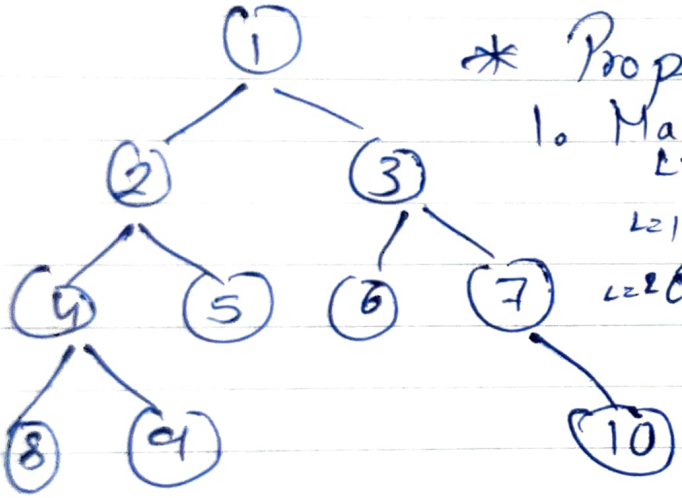


* In Computers the tree is represented in opposite direction. i.e the root is represented first and then the branches.



In trees each node can have unlimited branches or children.

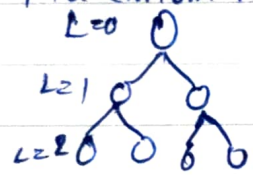
But, in case of Binary Tree the maximum children each node can have is two



Binary Tree

* Properties of Binary Trees

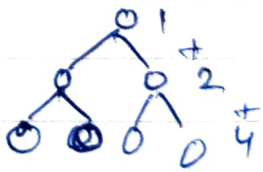
1. Maximum nodes at level $L = 2^L$



$L = 0$, nodes = 1

$L = 1$; nodes = 2

2. Maximum nodes in a tree of height H is $2^H - 1$



Height = Total no. of Levels

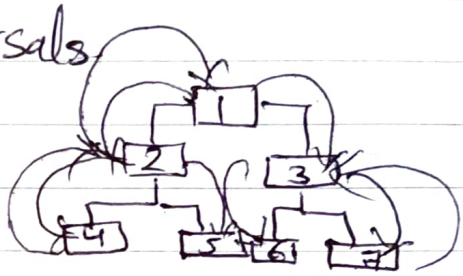
$H = 2^H - 1 = 2^3 - 1$

3. For N nodes, minimum possible height or minimum number of levels are $\log_2(N+1)$

4. A binary tree with L leaves has at least $\log_2(N+1) + 1$ number of levels.

DD/MM/YYYY

Day 29 Binary Tree Traversals



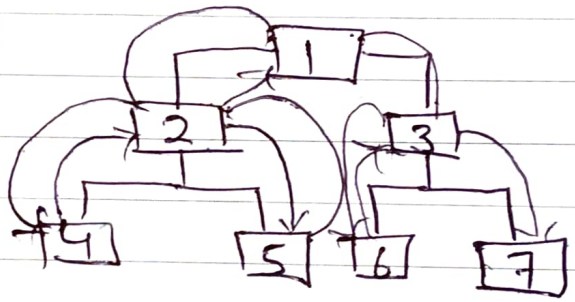
1. Preorder Traversal

- Read the root
- Read the left subtree
- Read the right subtree

1, 2, 4, 5, 3, 6, 7

2. Inorder Traversal

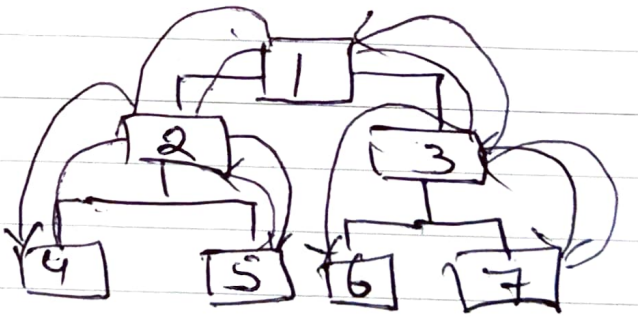
- Left subtree
- Root
- Right sub tree



Inorder: 4, 2, 5, 1, 6, 3, 7

3. Postorder Traversal

- Left subtree
- Right subtree
- Root

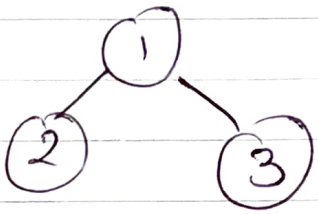


Post order: 4, 5, 2, 6, 7, 3, 1

Day 30 Build tree from preorder & inorder

Preorder
1 2 3

Inorder
2 1 3
Left ↑ right



Algorithm

1. Pick element from preorder & create a node.
2. Increase preorder index.
3. Search for picked element's position in inorder.
4. Call to build left subtree from inorder's 0 to pos-1.
5. Call to build right subtree from inorder pos+1 to n.
6. Return the node.