Day27: Introduction To Data Structures

- what are Data Structures?
- Why we need DS?
- Why we study time and space complexity
- Types
  - Linear
  - Non-linear

1) what are Data Structure
=> organization of data in a structured way.

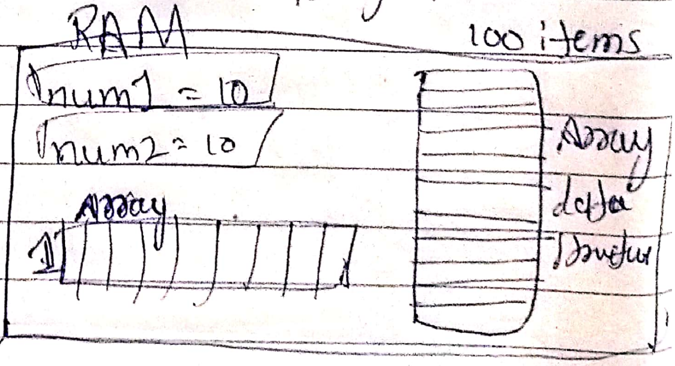
CRUD -> Create, Read, Update, Delete.

RAM   DB -> files.

Memory -> DS

2) why we need DS.
3) why we study time and space complexity
4) Type

Linear	Non-linear
-> arrays	-> Trees
-> Linked list	-> Heap
-> Array List	-> Map
-> Stack	-> ~~Tree~~ Trie
-> Queue	-> Graph
-> Dictionary	
-> Sets	
-> Tuple	



problem: document has 1000 words count the words how many times occured

solution: dictionary ds

Why Learn Data Structures?

Linear Datastructure
→ store data in single block sequentially one after another

Array

9	5	10	100	1	16	7
0	1	2	3	4	5	6

Index
↳ 4 byte

0	1	2	3	4	5	6
4	4	4	4	4	4	4

→ byte

Address 100 104 105 106 107

O(n)

Time Complexity :-

Time complexity is way to describe how long an algorithm (a step-by-step method to solve a problem) takes to run.

Know Time complexity of code

1) Identify Basic Operations
2) count the basic operations
3) Express the count as a function
4) Simplify the Function
5) Using Big O Notation

Constant O(1)

Array Numbers

Index	Value
0	1
1	2
2	3
3	4
4	5
→ 5	6

retrieve value at 5th index.

Numbers[5]   constant time   O(1)

Logarithmic O(log n)

Binary linear search x7 Best Case linear search → worst case

1	2	3	4	5	6	7	8	9	10

key = 9

binary search   divide & conquer

1	2	~~3~~	4	5	6	7	8	9	10

key = 9   middle = 5   9 > 5

middle = 8   key 9 > 8

middle = 9   key = 9

time O(log n)

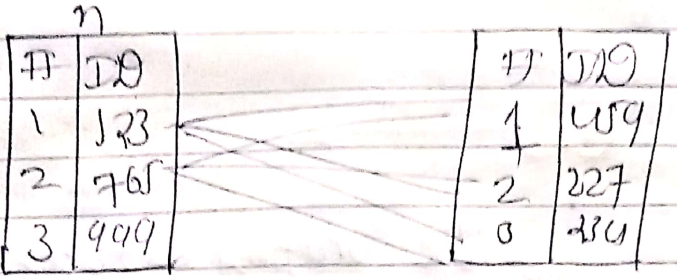
Linear O(n) : linear search - worst case

14	23	24	48	523	61	72	8	93	150

key = 150

O(n)

Quadratic O(n^2)

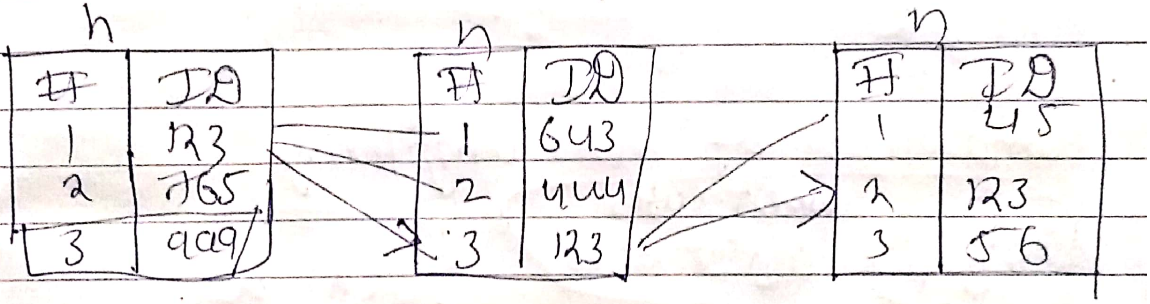


#	ID
1	123
2	765
3	909

#	ID
1	459
2	227
0	234

~~n*~~ n*n O(n^2)

Cubic O(n^3)



#	ID
1	123
2	765
3	909

#	ID
1	643
2	444
3	123

#	ID
1	45
2	123
3	56

n x n x n

O(n^3)

Exponential ~~O n^~~ $O(2^n)$ $O(n^n)$

1	2	3	4	5	6	7	8	9	10

All possible subsets

2 - 4

2 - 6

Factorial $O(n!)$

Generates all possible seating arrangments for a list of friends ID's

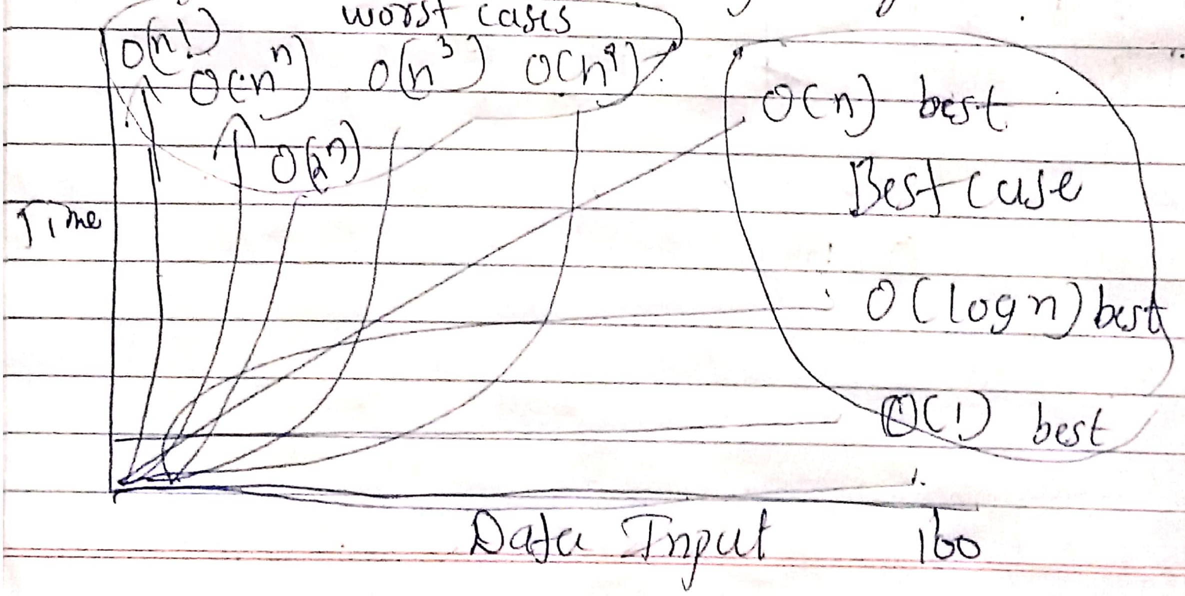
1	2	3	4	5	6	7	8	9	10

123

231

132

comparison of time complexity



why learn DS:

Linear DS
-> store data sequentially one after another.

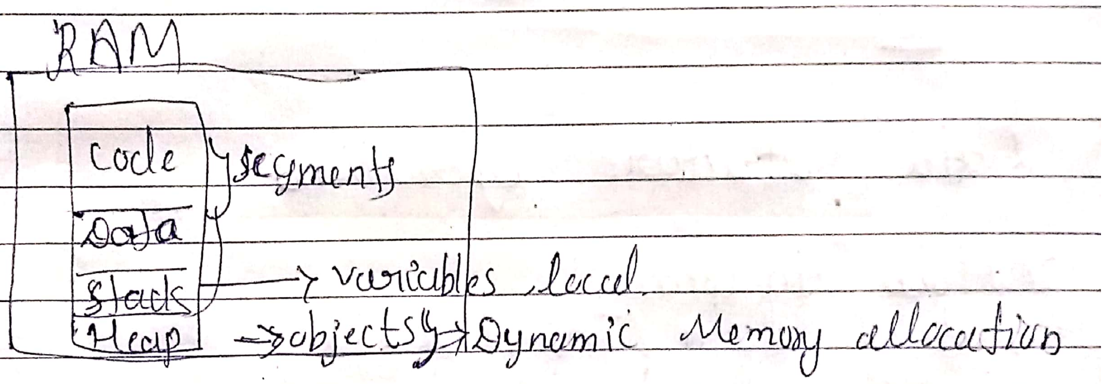
Array List -> supports primitive datatypes

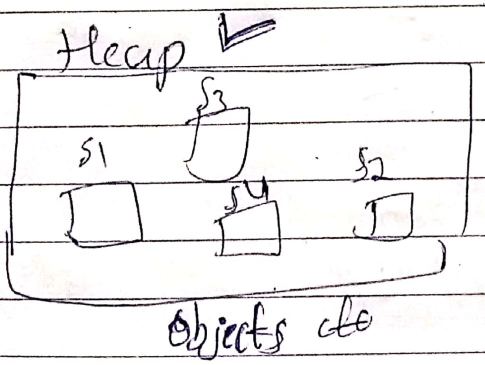
Single Linked List :

disadvantage of array -> cons

|1|2|3|4|X| (ArrayIndex out of bound) Error

Dynamic Memory Allocation



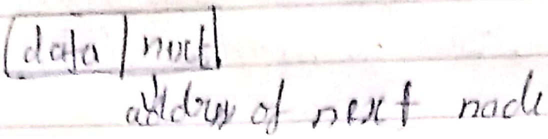


If no space in Heap through Error out of Memory Exception

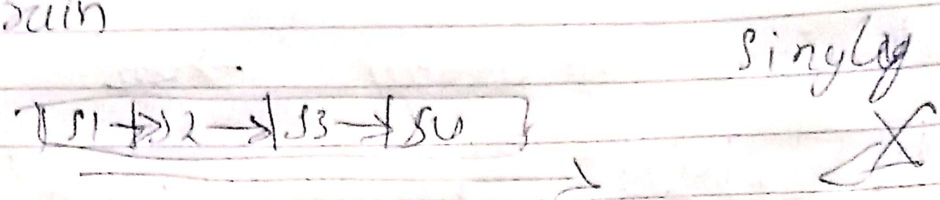
Garbage Collector : clean memory after complete task

Linked list

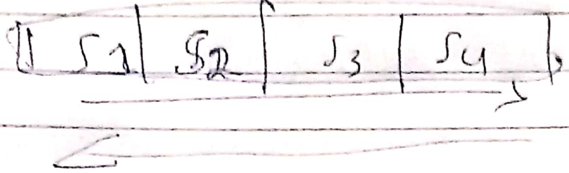
each node has
an object
each node has
address



Train



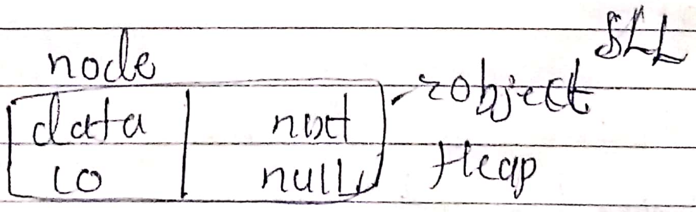
Doubly Linked list

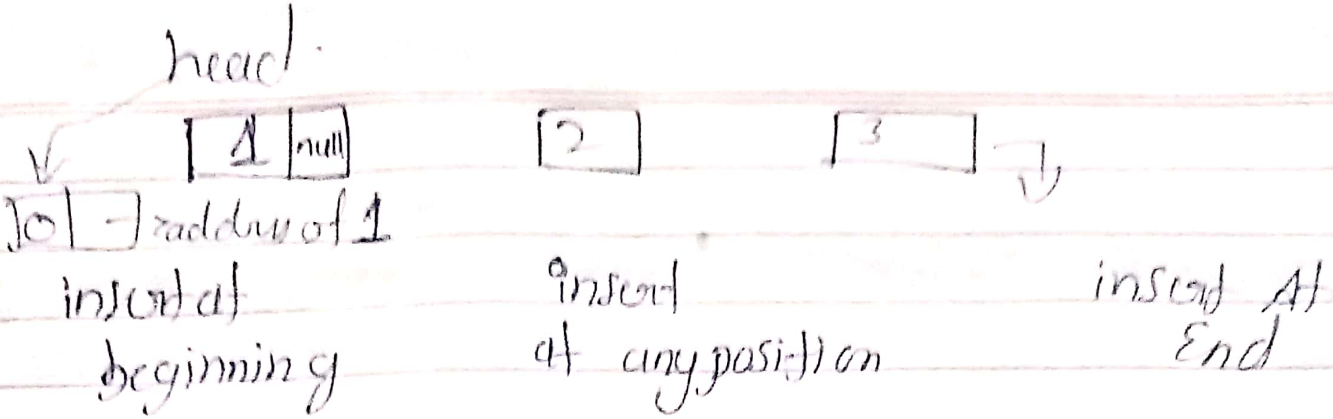


Data Structure Operations

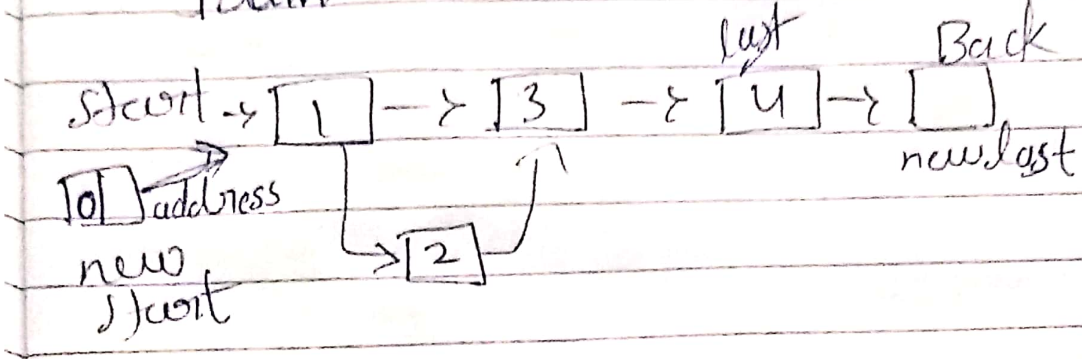
- How to create
- Add data (Insert data
- Delete data
- Search data
- print lists data
- update data

create a Linked list





Train



Node → User define datatype

int data

Node next

}