SMART

kapp

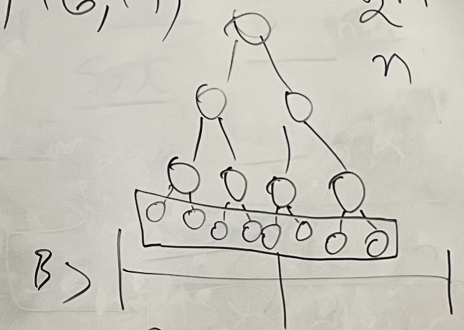
Trees

- Log N
- Traversing
  - BFS, DFS

Terms

- Height, balanced, leaf.

[2, 10, 12, 5, 16, 17]



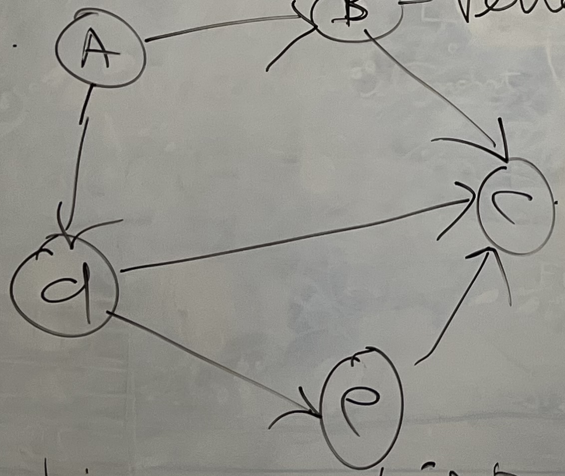
2^{h-1}

n

DS

- Array ✓
- Linked list ✓
- Trees
- Heaps / Priority Queue
- Tries
- Graph
- Stack ✓
- Queue ✓ FIFO O(1) Pop (Q=deque())
- Hashmap
- Hashset

Graph

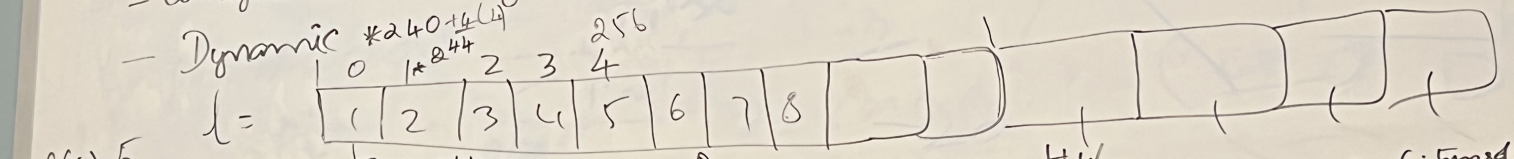


- O(V) — To see if Vertex is there or not
- Vertex - O(V+E) - Traversal (not

{ A: [B, D]
B: [C]
C:
D: [E]
E: [F] }

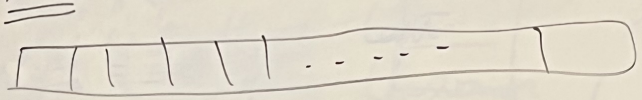
- Adjacency List
- Matrix
- ,)

- List (Python)
  - Collection elements, not necessarily of same Datatype.
  - Contiguous memory (in the memory the reference are stored continous)
  - Dynamic



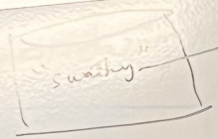
l(4)=5

X 2 — Growing factor



1.27 Python

- Insertion O(1) - append
- Insertion at given index - O(n)
- Deletion O(n)
- updating O(1)
- Look up O(1) - (Given its index / not element)



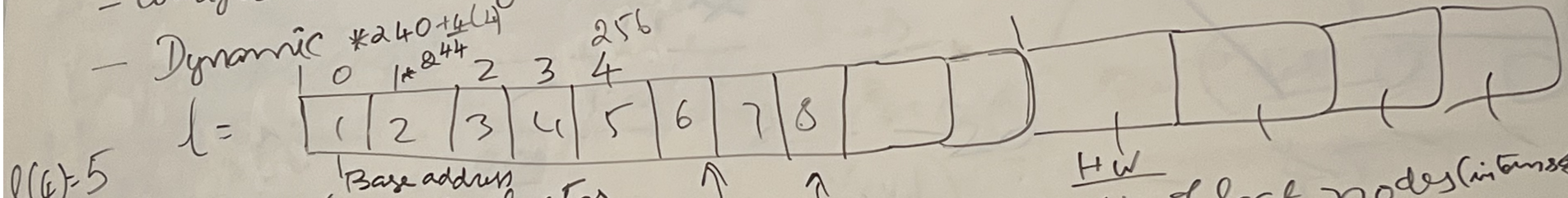
HW

3. No. of leaf nodes (in terms of N) in BT
1. Does the Array shrink in Size when we delete a lot of elements?
2. Can we ALWAYS use List as Stack?
4. How is Tree represented in ML?

t = [1, "Swathy", 3, "Hi"]

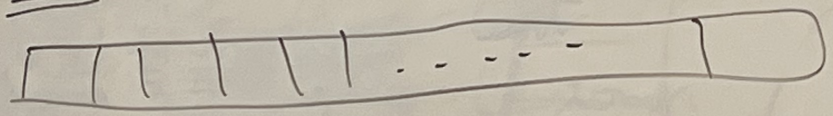
t[1] =

- List (Python)
  - Collection elements, not necessarily of same Datatype.
  - Contiguous memory (in the memory the reference are stored continou
  - Dynamic



l(4)=5

X2 — Growing factor



1.27 Python

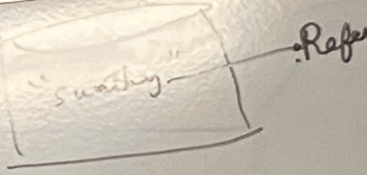
HW

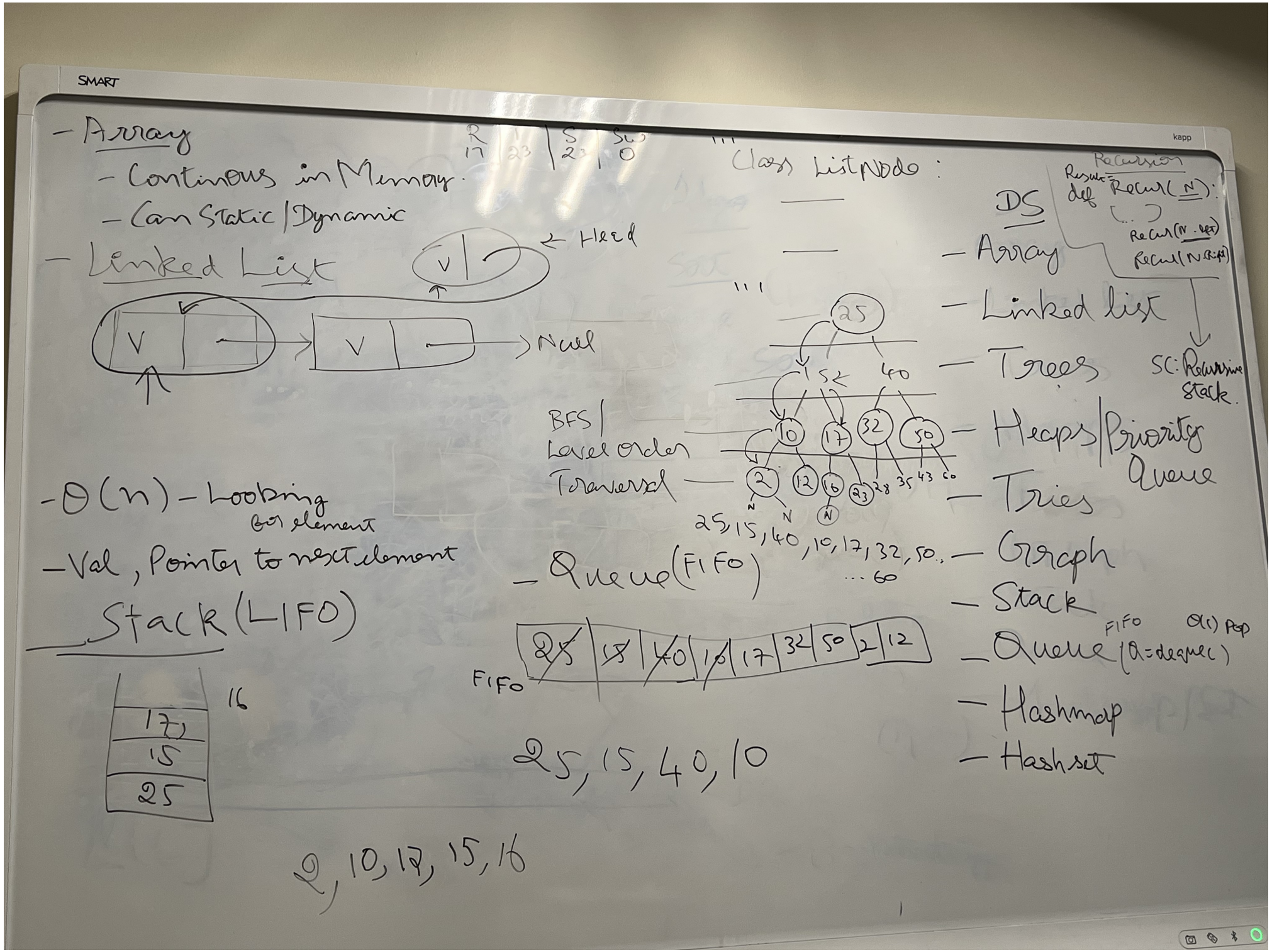
3. No. of leaf nodes (in terms of N) in
1. Does the Array shrink in Size when we delete a lot of elements?
2. Can we ALWAYS use List as Sta

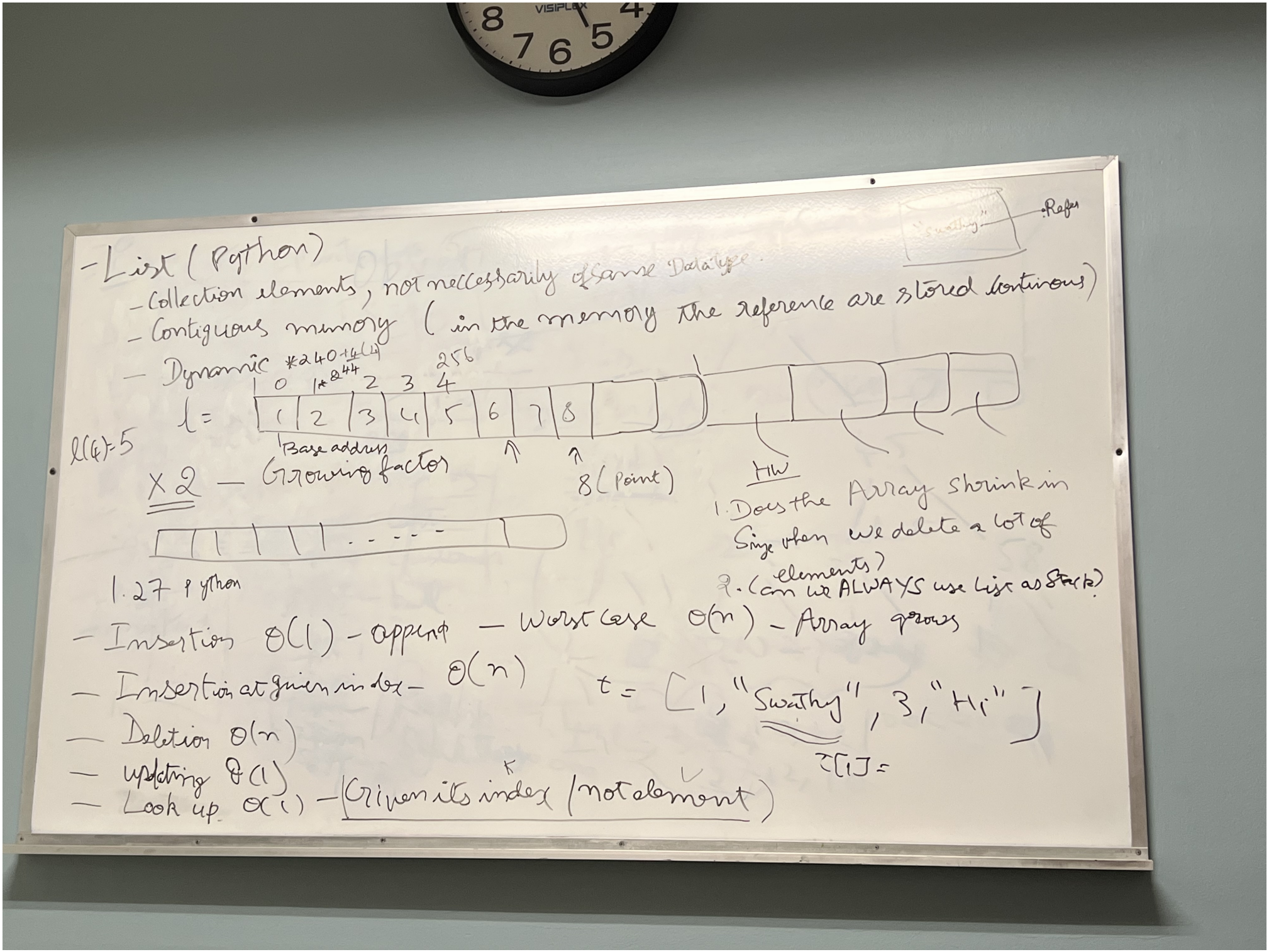
- Insertion O(1) - append — Worst case O(n) - Array grows
- Insertion at given index - O(n)
- Deletion O(n)
- updating O(1)
- Look up O(1) - (Given its index / not element)

t = [1, "Swathy", 3, "Hi"]

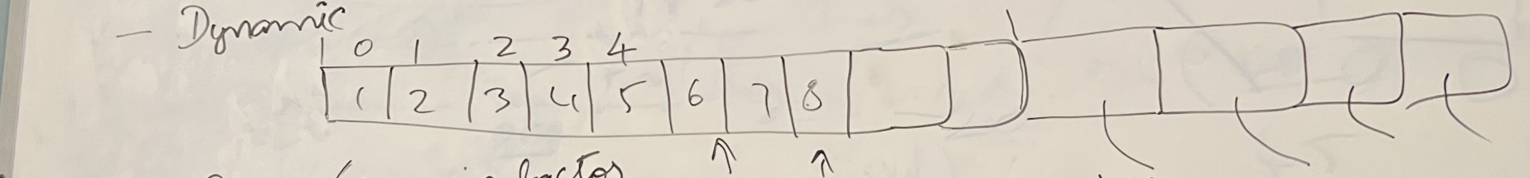
t[1] =




SMART
kapp
- Array
- Continous in Memory
- Can Static / Dynamic
- Linked List
Head
V
V
V
Null
-O(n) - Looping
Get element
- Val, Pointer to next element
Stack (LIFO)
16
17
15
25
2, 10, 17, 15, 16
R 17 | 23 | S 23 | SC 0
Class ListNode:
DS
- Array
- Linked list
- Trees
- Heaps / Priority Queue
- Tries
- Graph
- Stack
- Queue
- Hashmap
- Hashset
Recursion
Result = def Recur(N):
...
Recur(N.left)
Recur(N.right)
SC: Recursive stack.
BFS / Level order Traversal
25
15
40
10
17
32
50
2
12
16
23
28
35
43
60
25, 15, 40, 10, 17, 32, 50, ... 60
- Queue (FIFO)
FIFO
25 | 15 | 40 | 10 | 17 | 32 | 50 | 2 | 12
25, 15, 40, 10
FIFO O(1) Pop
(Q = deque( )


- List (Python)
- Collection elements, not necessarily of same Datatype
- Contiguous memory (in the memory the reference are stored continous)
- Dynamic
*240+4(4)
*244
256
0 1 2 3 4
1 2 3 4 5 6 7 8
l =
l(4)=5
Base address
X 2 — Growing factor
8 (Point)
"Swathy"
Refer
HW
1. Does the Array shrink in Size when we delete a lot of elements?
2. Can we ALWAYS use List as Stack?
1.27 Python
- Insertion O(1) - append — Worst case O(n) - Array grows
- Insertion at given index - O(n)
- Deletion O(n)
- updating O(1)
- Look up O(1) - (Given its index / not element)
t = [1, "Swathy", 3, "Hi"]
t[1] =

- List (Python)
  - Collection elements, not necessarily of same Datatype
  - Contiguous memory (in the memory the reference are stored continous)
  - Dynamic

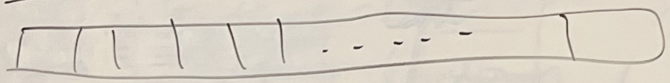


X2 — Growing factor

8 (Point)

HW

1. Does the Array shrink in Size when we delete a lot of elements?



1.27 Python

- Insertion O(1) - append — Worst case O(n) - Array grows
- — Insertion at given index — O(n)
- — Deletion O(n)
- — updating O(1)
- — Look up O(1) — Given its index (not element)