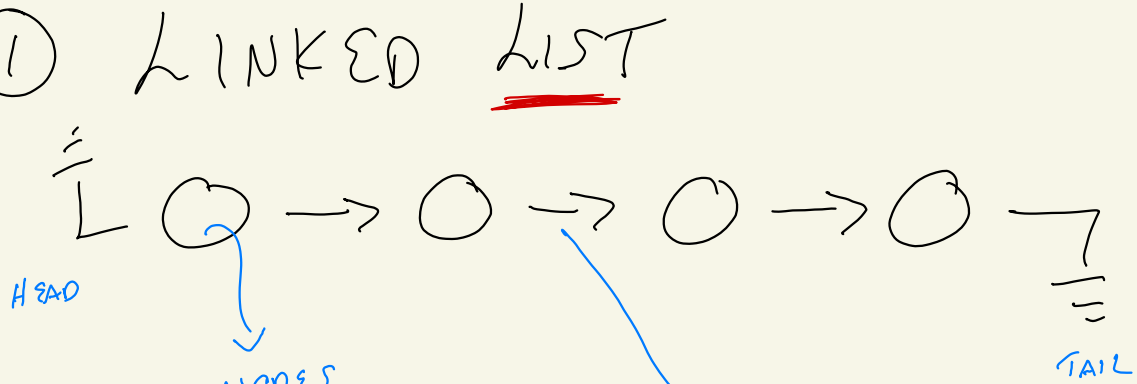
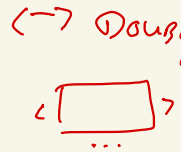
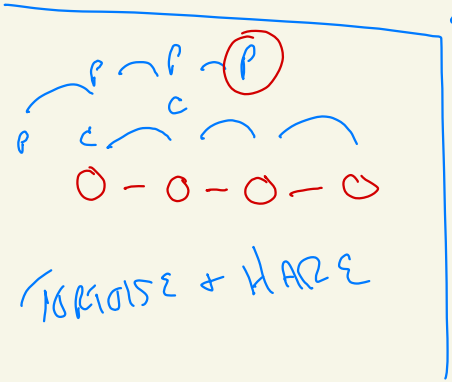
DATA STRUCTURES

① LINKED LIST



→ SINGLY LINKED LIST





STACKS + QUEUES

SAMENESS

- PEEK() - NON-DESTRUCTIVE
- INHERENTLY SELF-DESTRUCTIVE
  - AS YOU TRAVERSE
  - REMOVE
  - AT END:
    EMPTY

"PROCESS"

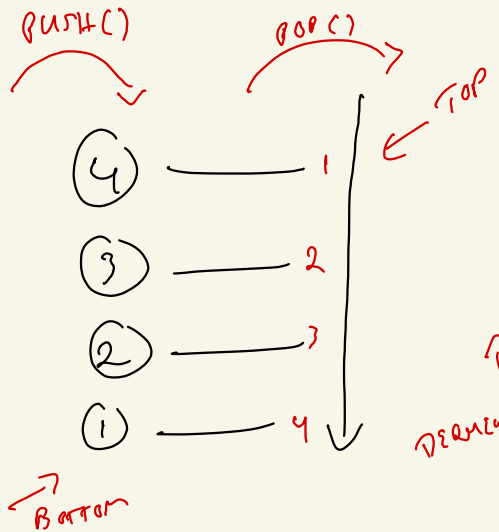
TRAVERSE: WHILE(PEEK())

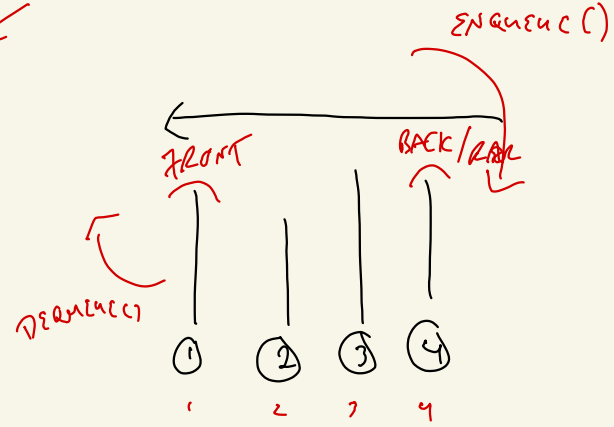
DIFFS

ORDER:

STACK: FILO, LIFO

QUEUE: FIFO, LIFO

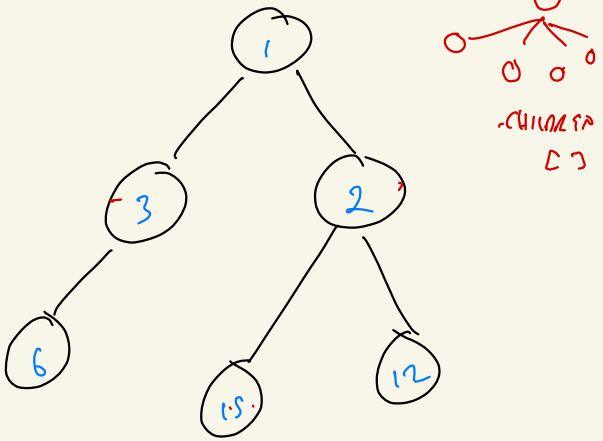




← TREE

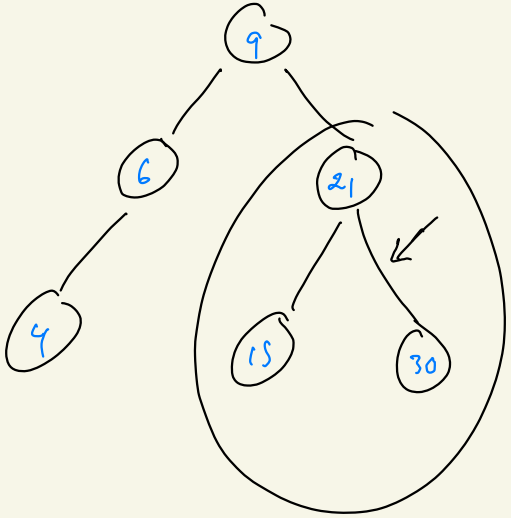
HIERARCHY / ORDER

1 BINARY



2 BST

SEARCH



COMMON

CHILDREN: .LEFT .RIGHT

PARENT: .ROOT

LEAF = NO CHILDREN

EDGE: CONNECTORS

SUB-TREE

BRANCH

DEPTH FIRST
- PRE
- IN
- POST

BREADTH FIRST

BF: 1 3 2 6 15 12

HOW MANY .....

TIMES DOES ALONE RUN

BYTES DO I CONSUME

COPIES DO I MAKE

ENTRIES IN CALL STACK

efficiency

OPERATIONAL COMPLEXITY

TIME = COMPUTE CYCLES

OPS

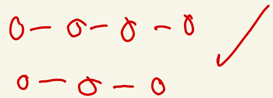
SPACE = MEMORY / FOOTPRINT

BYTES / THINGS / COPIES

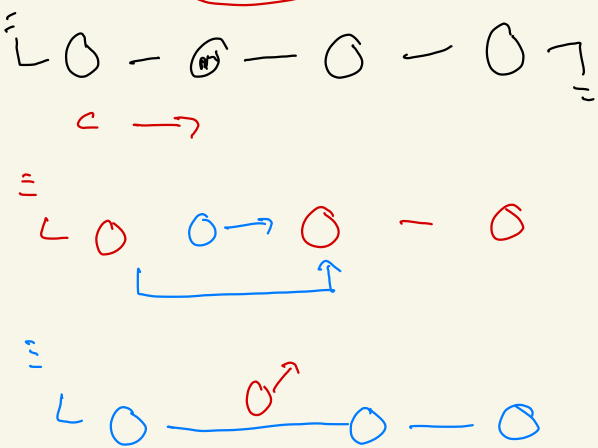
RESTATE QUESTION

Q & A

CLARIFY



VISUAL



WR

o — LOOP

o — LOOK

o — REPOINT

O: T(N^2)

S(1)

CODE

```
f() {

}
```

HASH TABLE / HASH MAP

KEYS VALUES

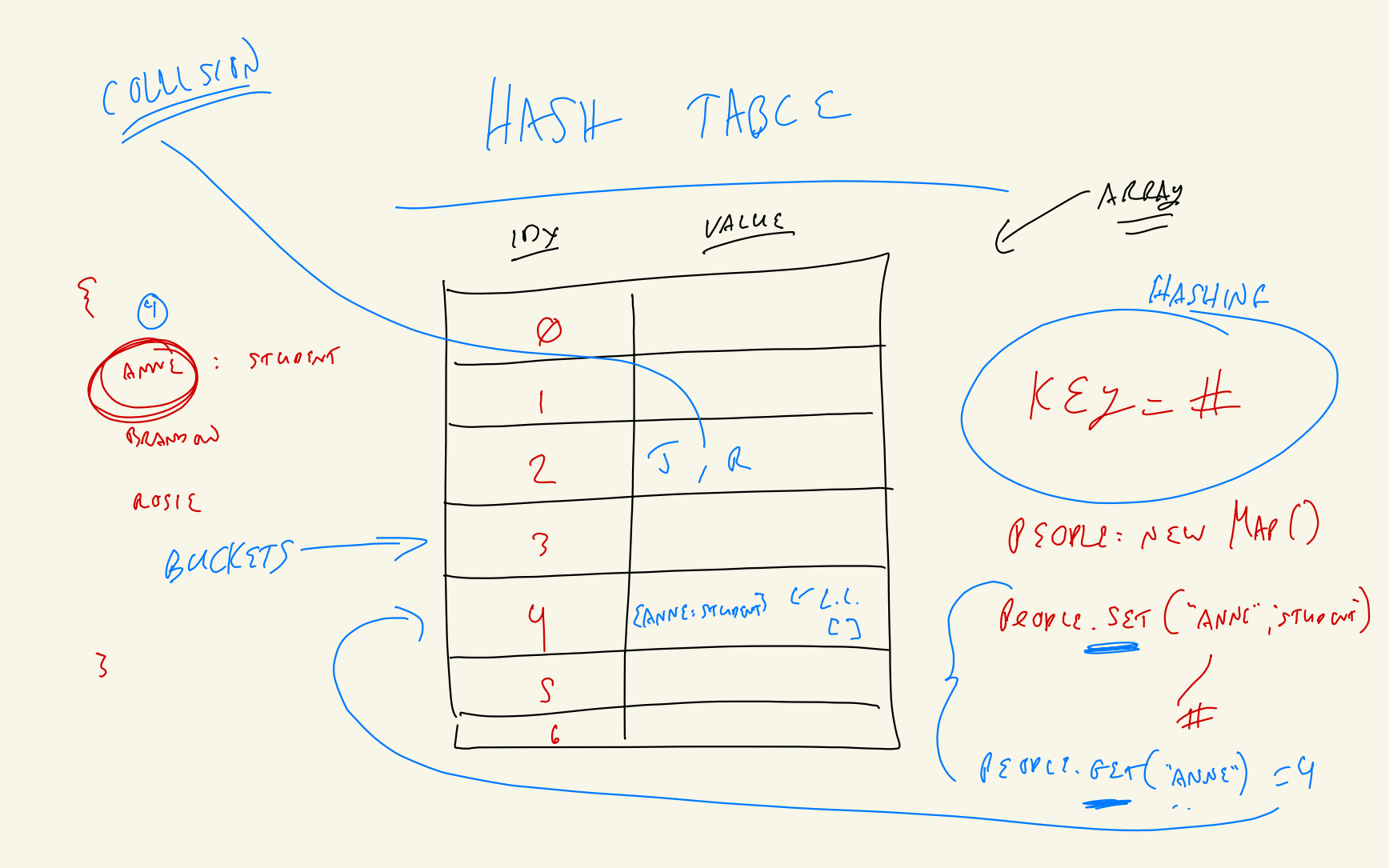
```
{
  ANNE: 'STUDENT',
  BRANDON: "STUDENT",
  ROSIE: "DOG",
}
```

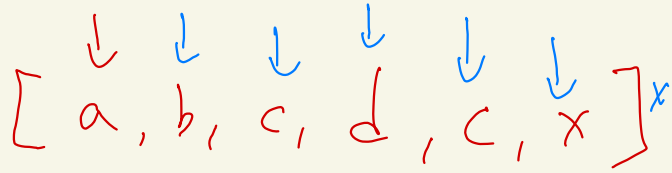
KEY/VALUE STORE

OBJECTS

JSON

FIND A KEY = O(1) TIME


COLLISION
HASH TABLE
ARRAY
IDX
VALUE
0
1
2
3
4
5
6
J, R
[ANNE: STUDENT]
[]
HASHING
KEY = #
PEOPLE: NEW MAP()
PEOPLE.SET("ANNE", STUDENT)
#
PEOPLE.GET("ANNE") = 4
BUCKETS
ANNE : STUDENT
BRIANNA
ROSIE
3



{

a : TRUE,

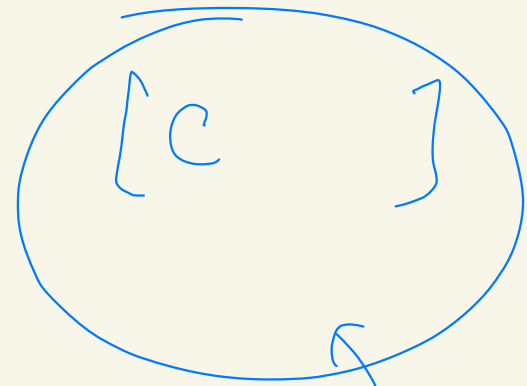
b : TRUE,

c : TRUE

d : TRUE

x : TRUE

}



NEW MAP()

if (MAP.HAS("a")) { }

MAP.SET("A", TRUE);