GATE

CRASH COURSE

Data Science & AI

Subject

Data Structure & Algorithms
Hash Tables and Queues
Lec No. - 02

By – Satya Sir

Last Class

Quick Recap

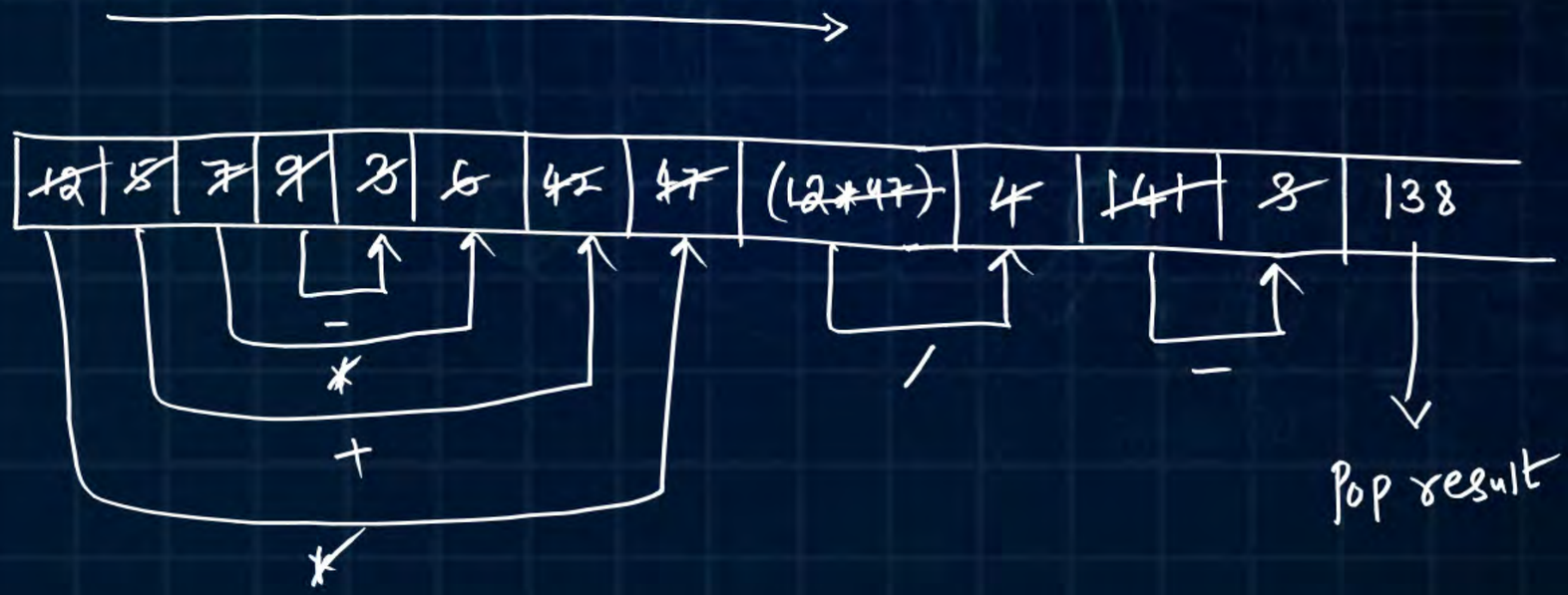
1. ~~Homework Question Solution~~
2. Data Structures, Classification
3. Stack, Operations
4. Applications Of Stack
5. Examples

Topics *to be covered*

1. Homework Questions Solution
2. Types Of Queues, Operations
3. Simple Queue, Circular Queue
4. Deque, Priority Queue
5. Hashing, Collision Resolution Techniques

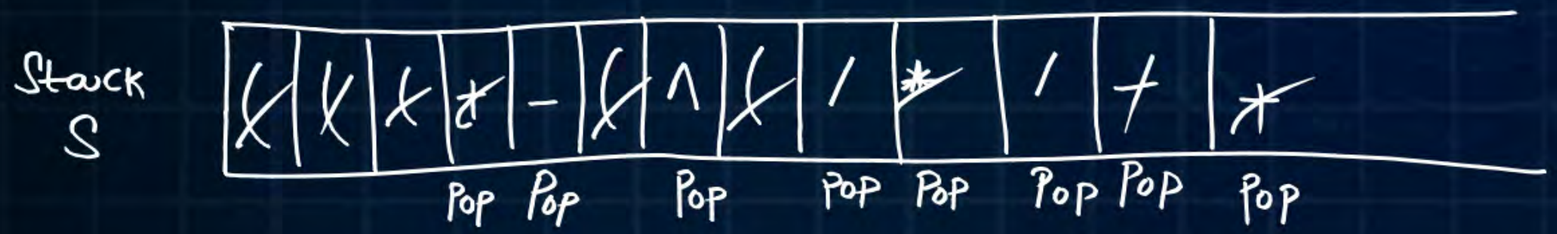
Homework Question - 1

Evaluate Postfix Expression: 12 5 7 9 3 - * + * 4 / 3 -



Homework Question - 2

Convert Infix Expression **(((P+Q)-(R^(S/T*U)/V)-W)+X)** to Postfix Expression



Y: PQ+RST/U*^V/-W-X+

Queue DS

- A Linear DS, in which insertion / Deletion of Elements Performed from different ends of the list, is known as Queue DS.

- Insertion End : Rear End Deletion End : front End.

- In Queue Implementation, Insertion operation == Enqueue( ) operation
  Deletion Operation == Dequeue( ) Operation

Types Of Queues

4 Types of Queues :

1) Simple Queue 2) Circular Queue 3) Double Ended Queue 4) Priority Queue

- In Simple Queue, Circular Queue : Insertion End : rear End,
  Deletion End : front End.

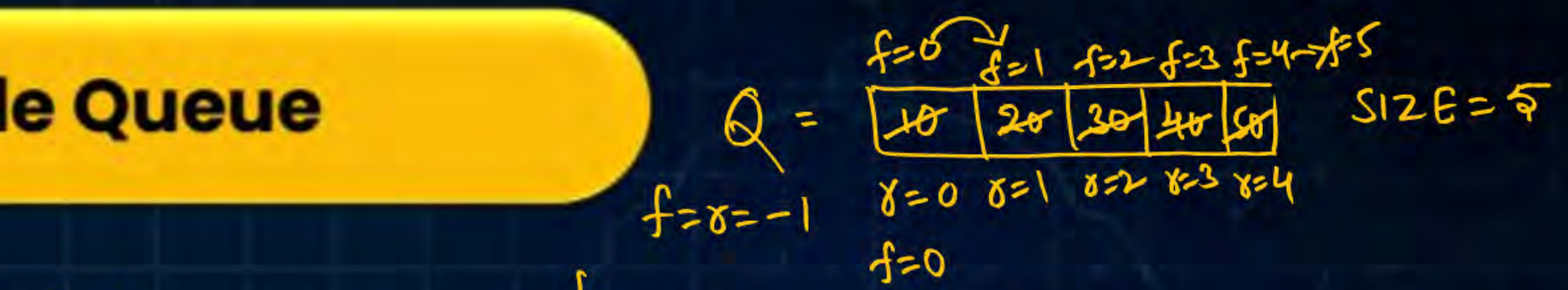
⇒ First-In-First-out (FIFO) Lists.

- In Double Ended QUEue (DEQUE), Insertion : front, rear End
  Deletion : front, rear End

- Deque Can be Used as both FIFO list, LIFO lists.

- In Priority Queue, Deletion is not Performed in the insertion Sequence but based on Priority.

Simple Queue




```
def Enqueue(Q, Value):
    if r == SIZE - 1:
        Print('Queue is Full') #overflow
        return
    r = r+1
    Q[r] = Value
    if f == -1:
        f = f+1
```

Time Complexity : O(1)

```
def Dequeue(Q):
    if f == -1:
        Print('Queue is Empty') #underflow
        return
    deleted_Element = Q[f]
    f = f+1
    if f == r+1:
        f = r = -1
```

Time Complexity : O(1)

Circular Queue

Queue, Q: | ~~10~~ | ~~20~~ | 30 | 40 | 50 |

r=0 r=1 r=4

SIZE = 5

f = r = -1 f=0 f=2

enqueue(Q, 10) r=0 f=0

enqueue(Q, 20) r=1

enqueue(Q, 30) r=2

enqueue(Q, 40) r=3

enqueue(Q, 50) r=4

dequeue(Q) f=1

dequeue(Q) f=2

enqueue(Q, 60):

r == SIZE-1 ?

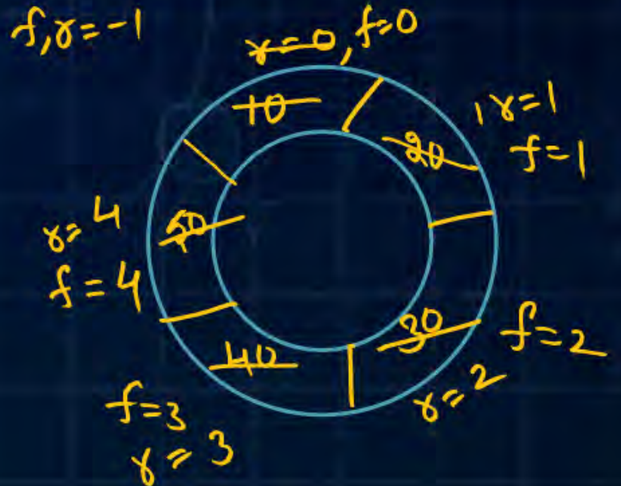
4 == 5-1 ? True

Print 'overflow'

Once, Simple Queue is Full, until it is cleared Completely, further insertion Not Possible. Hence Circular Queue is used.

Circular Queue : It is like Simple Queue only, but rear and front updates in Circular (or) cyclic manner.

$$f = (f+1) \,\%\, SIZE, \quad r = (r+1) \,\%\, SIZE$$



Full : f == (r+1) % SIZE

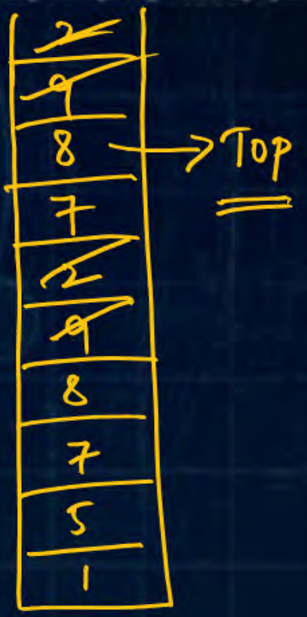
Empty : f == r or f == -1

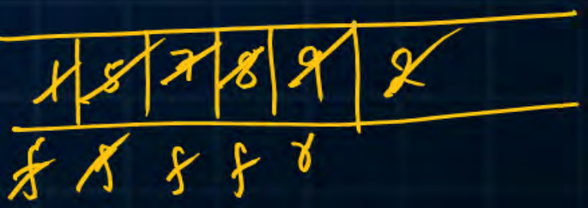
Question

#Q. Consider a sequence a of elements a0 = 1, a1 = 5, a2 = 7, a3 = 8, a4 = 9, and a5 = 2. The following operations are performed on a stack S and a queue Q, both of which are initially empty.

I: push the elements of a from a0 to a5 in that order into S.
II: enqueue the elements of a from a0 to a5 in that order into Q.
III: pop an element from S.
IV: dequeue an element from Q.
V: pop an element from S.
VI: dequeue an element from Q.
VII: dequeue an element from Q and push the same element into S
VIII: Repeat operation VII three times.
IX: pop an element from S.
X: pop an element from S.

The top element of S after executing the above operations is 8.





Double Ended Queue (DeQue)

- In Deque, Insertion, deletion can be Performed from both Ends.

- It can be developed in 2 ways:

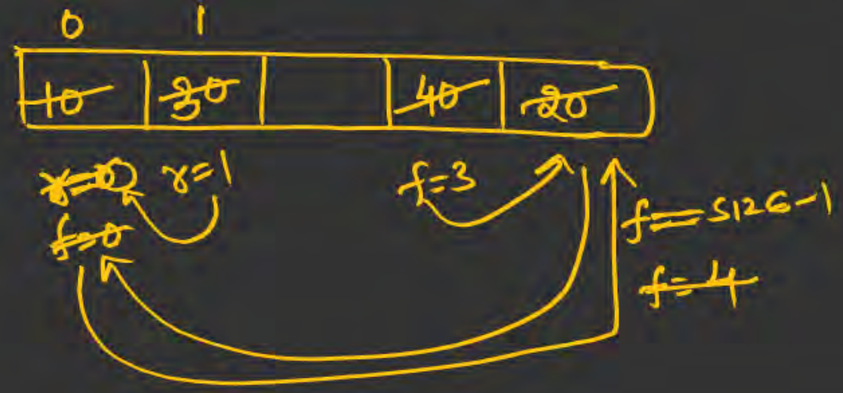
  1) Input-restricted Deque : Insertion : rear End only, Deletion : rear, front Ends

  2) Output-restricted Deque : Insertion : rear, front Ends, Deletion : front End only.

- In DEQUE, Empty : $f == r$ or $f == -1$

  Full : $f == (r+1) \% size$

Let Deque size = 5, Initially Empty ⇒ f=-1, r=-1



Enqueue_rear(10) : r=0, f=0

Enqueue_front(20) : r=0, f=4

Enqueue_rear(30) : r=1, f=4

Enqueue_front(40) : r=1, f=3

dequeue_rear( ) : r=0, f=3

dequeue_front( ) : r=0, f=4

dequeue_front( ) : r=0, f=0

dequeue_front() | dequeue_rear( ) f=r=-1

	rear	front
Insertion	rear = (rear+1) % size	f = f-1 if f == 0 ⇒ f = size-1
Deletion	rear = rear-1 if rear == 0 rear = size-1	front = (front+1) % size

Question

#Q. . Suppose you are given an implementation of a queue of integers. The operations that can be performed on the queue are:

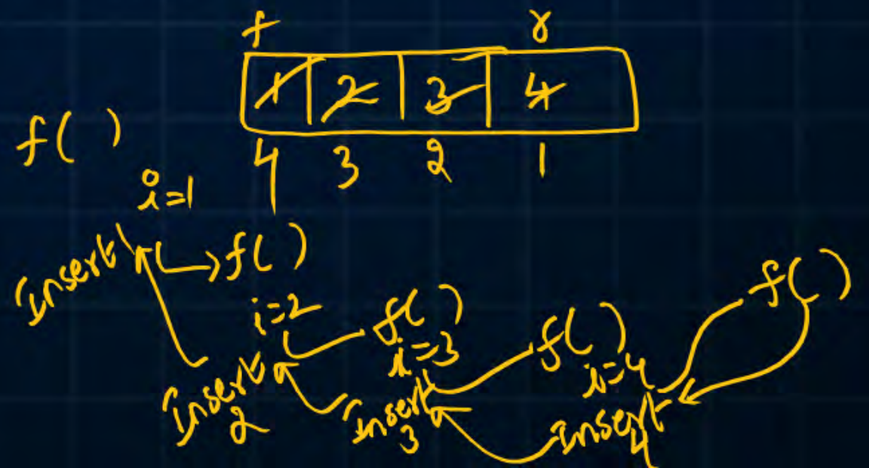
i. isEmpty (Q) — returns true if the queue is empty, false otherwise.
ii. delete (Q) — deletes the element at the front of the queue and returns its value.
iii. insert (Q, i) — inserts the integer i at the rear of the queue.

Consider the following function:

```
from collections import deque
def f(Q):
if not is_empty(Q):
        i = delete(Q)
        f(Q)
        insert(Q, i)
def is_empty(Q):    return len(Q) == 0
def delete(Q):    return Q.popleft()
def insert(Q, i):    Q.append(i)
```

What operation is performed by the above function f ?

(A) Leaves the queue Q unchanged
(B) Reverses the order of the elements in the queue Q
(C) Deletes the element at the front of the queue Q and inserts it at the rear keeping the other elements in the same order
(D) Empties the queue Q



Question

Home-Work-1

#Q. The fundamental operations in a double-ended queue D are:
insertFirst(e) – Insert a new element e at the beginning of D.
insertLast(e) – Insert a new element e at the end of D.
removeFirst() – Remove and return the first element of D.
removeLast() – Remove and return the last element of D.
In an empty double-ended queue, the following operations are performed:
insertFirst(10)
insertLast(32)
a ←removeFirst()
insertLast(28)
insertLast(17)
a ←removeFirst()
a ← removeLast()
The value of a is _____.

Hash Tables, Hashing

Hashing : The Process of mapping key to value. Value is the slot (or) bucket number in memory.

- The memory region/area that stores keys based on Hashing is said to be Hash Table.
- To Map, Keys to Values, Hash Function (Pre-defined) is used.

- While Hashing, if more than one key, maps to the same value, then it is said to be COLLISION.
  - To resolve Collission, Collission Resolution Techniques are Used.

Collision Resolution Techniques

- Open-Hashing
  (OR)
  Seperate chaining Method
- Closed-Hashing
  (or)
  Open-addressing
  - Linear Probing
  - Quadratic Probing
  - Double-Hashing

Collision Resolution Techniques

Open-Hashing | Chaining Method

- Linked lists are used to maintain all keys that maps to same slot.

Ex: $h(k) = k \% 7$

keys = 23, 30, 27, 34, 9, 16, 21

$h(23) = 23 \% 7 = 2$

$h(30) = 30 \% 7 = 2$

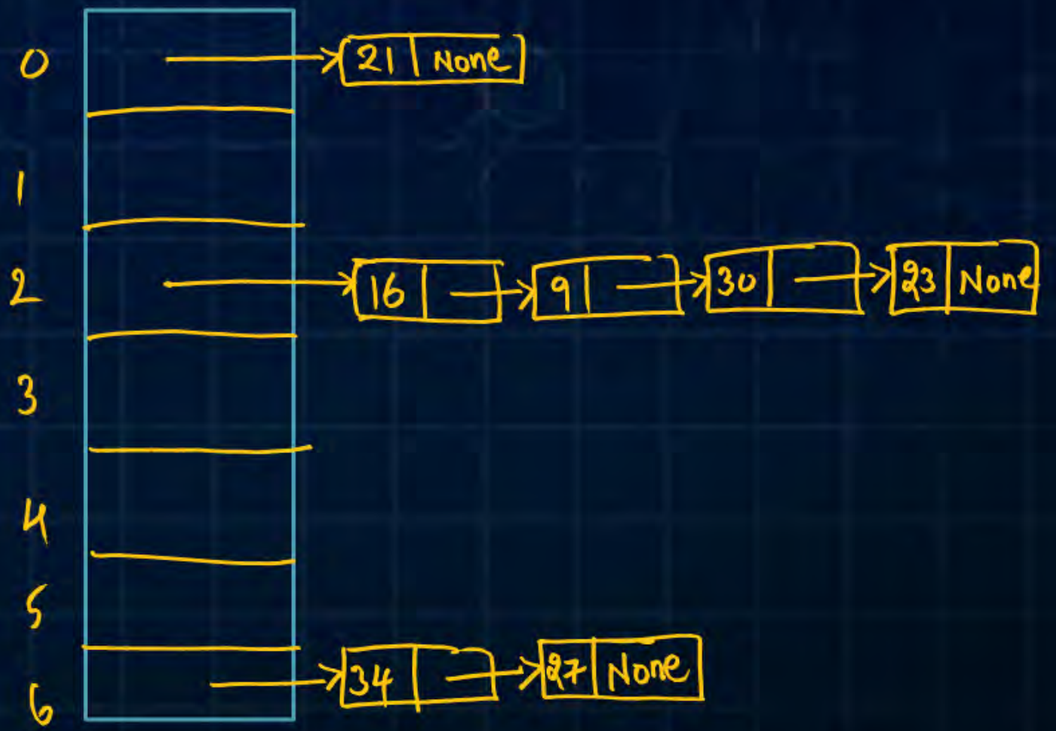
$h(27) = 27 \% 7 = 6$

$h(34) = 34 \% 7 = 6$

$h(9) = 9 \% 7 = 2$

$h(16) = 16 \% 7 = 2$

$h(21) = 21 \% 7 = 0$



Open-addressing Techniques

$i = 0, 1, 2, 3 - - -$ for each key.

When collission occurs for any key on certain 'i' value, Then only $i = i+1$.

Linear Probing : $h(K, i) = (K + i) \,\%\, N$

Quadratic Probing : $h(K, i) = (h'(K) + i^2) \,\%\, N$ $\boxed{h'(K) = K \,\%\, N}$

Double Hashing : $h(K, i) = [h_1(K) + i * h_2(K)] \,\%\, N$

$h_1(K) = K \,\%\, N$ $\quad h_2(K) = K \,\%\, N'$

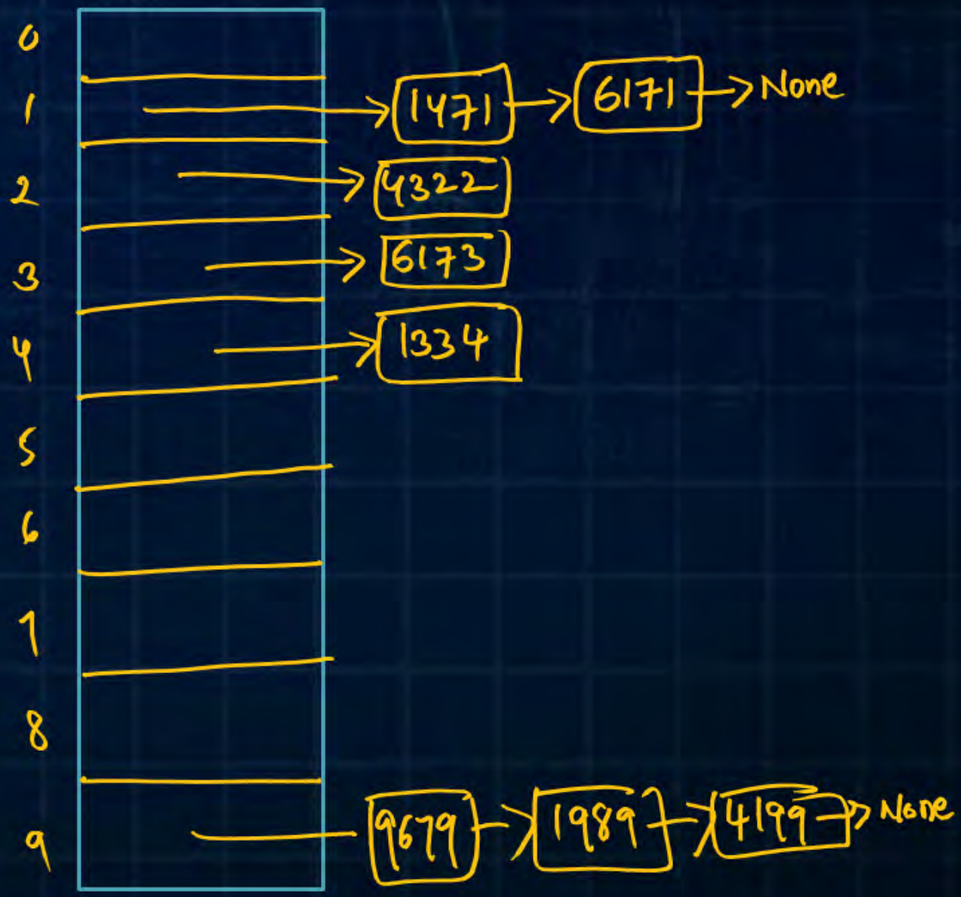
$\boxed{N' < N}$

Question

#Q. Given the following input (4322, 1334, 1471, 9679, 1989, 6171, 6173, 4199) and the hash function x mod 10, which of the following statements are true? (Chaining method)

1. 9679, 1989, 4199 hash to the same value
2. 1471, 6171 hash to the same value
3. All elements hash to the same value
4. Each element hashes to a different value

(A) 1 only
(B) 2 only
(C) 1 and 2 only
(D) 3 or 4



Question

#Q. A hash table contains 10 buckets and uses linear probing to resolve collisions. The key values are integers and the hash function used is key % 10. If the values 43, 165, 62, 123, 142 are inserted in the table, in what location would the key value 142 be inserted?

(A) 2
(B) 3
(C) 4
(D) 6

$h(k,i) = (k+i) \% N$

$h(43,0) = (43+0) \% 10$

$= 43 \% 10 = 3$

$h(165,0) = (165+0) \% 10 = 5$

$h(62,0) = (62+0) \% 10 = 2$

$h(123,0) = (123+0) \% 10 = 3$ Collission

$h(123,1) = (123+1) \% 10 = 124 \% 10 = 4$

0	
1	
2	62
3	43
4	123
5	165
6	142
7	
8	
9	

$h(142,0) = (142+0) \% 10$
$= 2$ Collission

$h(142,1) = (142+1) \% 10$
$= 143 \% 10 = 3$ Coll

$h(142,2) = (142+2) \% 10$
$= 144 \% 10 = 4$ Coll

$h(142,3) = 145 \% 10 = 5$ Coll

$h(142,4) = 146 \% 10 = 6$

Question

Home-work-2

#Q. Consider a double hashing scheme in which the primary hash function is h1(k) = k mod 21, and the secondary hash function is h2(k) = 1+(k mod 19). Assume that the table size is 21. Then the address returned by probe 1 in the probe sequence (assume that the probe sequence begins at probe 0) for key value k = 70 is _____ .

A. 0 B. 1 C. 2 D. 4

Question

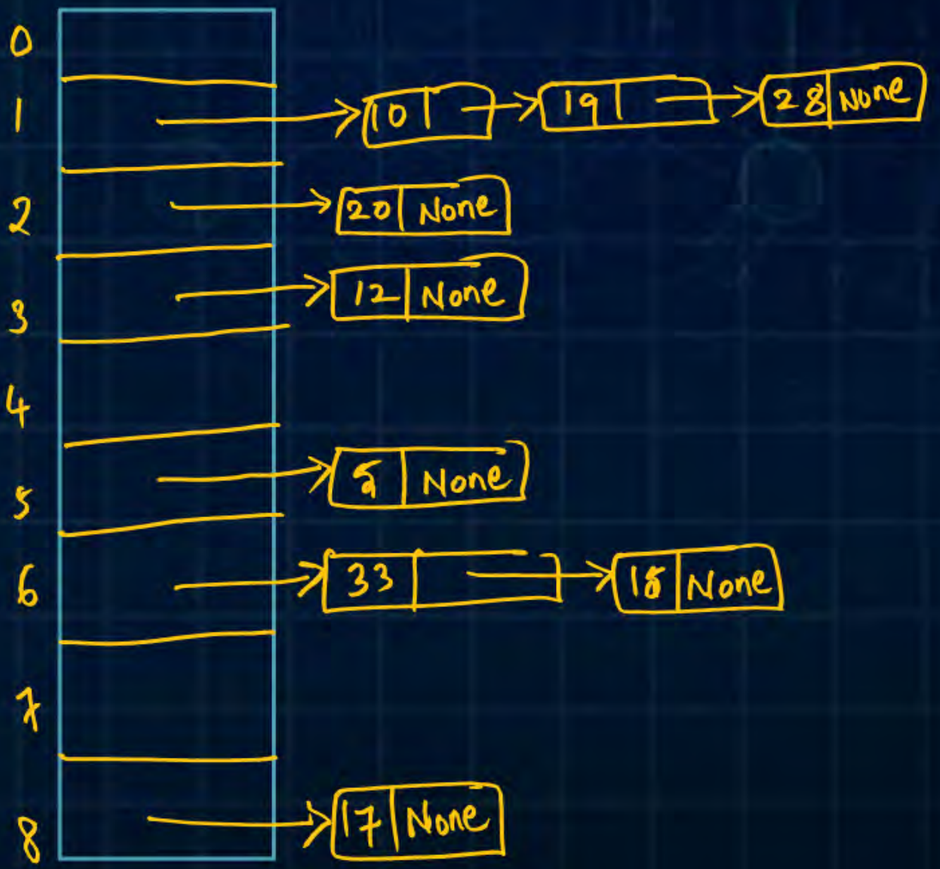
#Q. Consider a hash table with 9 slots. The hash function is $h(k) = k \bmod 9$. The collisions are resolved by chaining. The following 9 keys are inserted in the order: 5, 28, 19, 15, 20, 33, 12, 17, 10. The maximum, minimum, and average chain lengths in the hash table, respectively, are

(A) 3, 0, and 1
(B) 3, 3, and 3
(C) 4, 0, and 1
(D) 3, 0, and 2

$h(5) = 5 \% 9 = 5$
$h(28) = 28 \% 9 = 1$
$h(19) = 1$
$h(15) = 6$
$h(20) = 2$
$h(33) = 6$
$h(12) = 3$
$h(17) = 8$
$h(10) = 10 \% 9 = 1$

Chain Length

0	0
3	1
1	2
1	3
0	4
1	5
2	6
0	7
1	8



minimum = 0
Maximum = 3
Average = 1

$\frac{(0+3+1+1+0+1+2+0+1)}{9} = \frac{9}{9} = 1$

Post Your Homework Answers / Queries / Doubts @

t.me/ satyasirpw

Summary

- Queues
- Hash Tables


Thank
THANK
Keep Hustling!