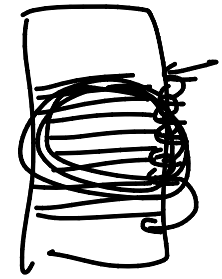
# 8. Hashing - 2, Queues

30 June 2024 14:01

⇒ **open addressing** :-

We place all key value pairs in separate slots.



1000
50 → 50051

$n \rightarrow$ keys

$m \rightarrow$ size of hash table

then $m \geq n$.

## ① Linear probing :-

$\rightarrow h(K) = (h_1(K) + i) \% m$

$= (h_1(K) + i) \% 7$

| | |
|---|---|
| 0 | 28 |
| 1 | 15 |
| 2 | 30 |
| 3 | |
| 4 | ~~48~~ |
| 5 | |
| 6 | 13 |

del → 37

S = 48

$h_1(K) = K$

$h(K) = (K + i) \% 7$

insert → 30, 28, 15, 13, 37, 48

39

$39 \% 7 = 4$

$(39+1) \% 7 = 5$

$(37+0) \% 7 = 2 \rightarrow$ filled

$(37+1) \% 7 = 3 \rightarrow$

$(48+0) \% 7 = 6$

$(48+1) \% 7 = 0$

$(48+2) \% 7 = 1$

$(48+3) \% 7 = 2$

$(48+4) \% 7 = 3$

$(48+5) \% 7 = 4$

$(48+5) \% 7 = 4$

## Conditions to stop while inserting :-

① You find empty / deleted slot.
② You complete one full circle.

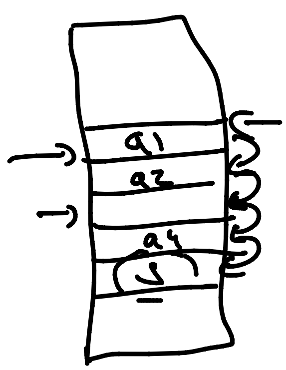
⇒ **Search** :- $S = 48$ $(48+0) \% 7 \rightarrow$

Keep searching by incrementing $i$

$h(K) = (h_1(K) + i) \% m;$

## Stopping criteria :-

① Found the value
② Completed full circle.
③ You found empty slot.




ins → 5
del → 92
find → 5

③ **Delete** :-

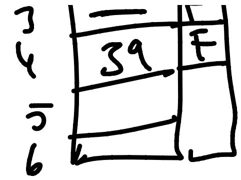
→ Search the value

deletion markers

| | | |
|---|---|---|
| 0 | 29 | |
| 1 | 36 | T |
| 2 | 37 | |
| 3 | — | F |
| 4 | 39 | F |

← 43, F, T

→ search the value




→ If found, make the deletion marker true.

→ Drawback :- Clustering.

$$h(K) = (K+i)\ \%\ 100$$

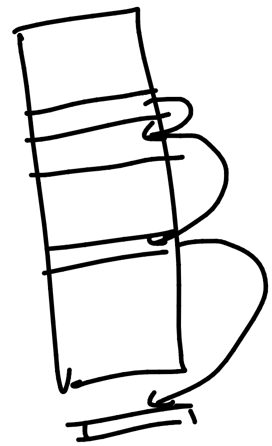
$K\ \%\ 100$ →


② Quadratic Probing :-

$$h(K) = (h_1(K) + i^2)\ \%\ m$$

in → 3, 9, 13, 12, 15, 8, 7,

$$h(K) = (K + i^2)\ \%\ 7$$



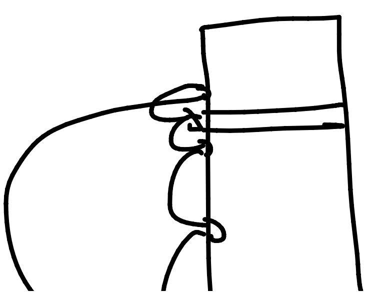
→ It may not always return an empty slot even if it exists.

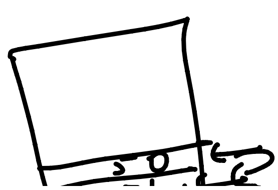
→ $\alpha < 0.5$ & $m$ → prime

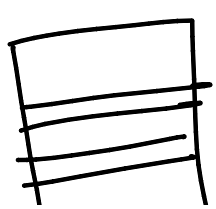
↓

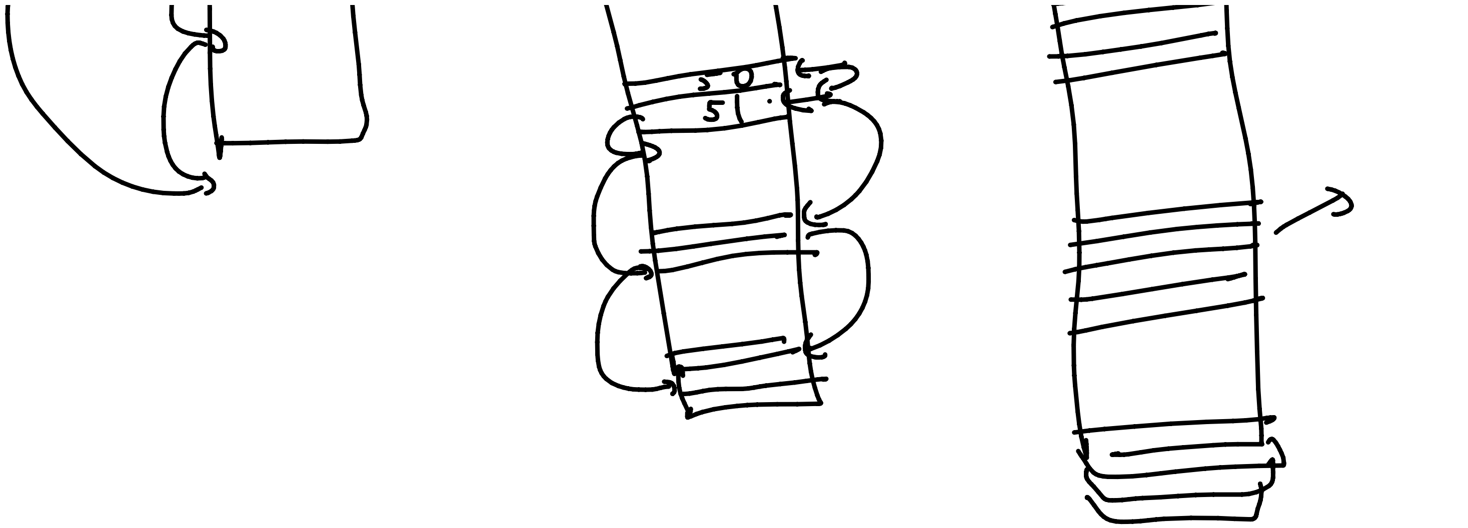
you will always find an empty slot.

⇒ Quadratic clustering.







## 3 Double Hashing

$$h(K) = (h_1(K) + h_2(K)\, i) \% m \rightarrow 7$$

$$\Rightarrow h_1(K) = K \% 10$$

$$h_2(K) = 6 - (K \% 6) \rightarrow \text{range} = \{1, 6\}$$

$K \% 6 = 0, 1, 2, 3, 4, 5$

$6 - K \% 6 = 6, 5, 4, 3, 2, 1$

$m = 7$

$i \rightarrow 32, 47, 33, 19, 7, 3, 22$

| | |
|---|---|
| 0 | 47 |
| 1 | 7 |
| 2 | 32 |
| 3 | 33 |
| 4 | |
| 5 | 9 |
| 6 | |

$h(K) = (2+0) \% 7 = 2$

$h(K) = (7+0) \% 7 = 0$

$h(K) = (3+0) \% 7 = 3$

$h(K) = (9+0) \% 7 = 2$

$= (9 + 1(6-3)) \% 7$

$= 5$

$h(K) = (7+0) \% 7 = 0$

$= (7 + 1(6-1)) \% 7$

$= 5$

$h(K) = (7 + 2(5)) \% 7$

$= 3$

$h(K) = (7 + 3(5)) \% 7$

$= 1$

$(7 + 4(5)) \% 7 = 6$

$(7 + 6(5)) \% 7 = 2$

$(7 + 5(5)) \% 7 = 4$

$(7 + 7(5)) \% 7 = 0$

$(7+5(5))\% 7 = 4$ / $(7+7(5))\% 7 = 0$ $h(K) = (7+5(5))\%\ldots$
$= 4$

→ Here, $h_2(K) \neq 0$,

→ If $h_2(K)$ & $m$ are co-prime, you will always find an empty slot if exists.

⇒ **Analysis** :-

Assumption → uniform distribution.

$75\%$ → 4 iterations (1 out of every 4 slot is empty)

$90\%$ → 10 iterations

$99\%$ → 100

$$\text{No of iterations} = \frac{1}{1-\alpha}$$

As $\alpha \to 1$,
# iterations $\to \infty$.

---

⇒ **Queues** :-

→ Inspired by real life queues.

3, 5, 7, 9, 12 → 5, 7, 9, 12, 18

→ 3, 5, 7, 9, 12,

→ 3, 5, 7, 9, 12,

queue → FIFO principle
first in first out.

front → <- rear   open at 2 ends.

insert → rear → enqueue
remove → front → dequeue

O(1)
→ empty(), size(), front() → normal queue
↓
return value at front end of queue.
→ q.push(30)
→ q.pop()

→ q → empty
push(10)
push(20)
push(30)
pop()
front()

pop()
→ size()
→ empty()
→ push(50)
→ front()

| | | |
|---|---|---|
| 10 | 20 | 30 |

↓

| | |
|---|---|
| 20 | 30 |

| | |
|---|---|
| 30 | 50 |

O/P → 20, 1, F, 30

---

Stack using 2 queues:-

push(1)
push(2) ✓
push(3) ✓

q1 → 1
q2 ← q1
q1 → 2, 1

q2 → 1
q1 → —

q2 → 2 1

q1 → 3

q1 → 3, 2, 1 → pop() → 3

4 q1 → 3, 2, 1

q2 → 3, 2, 1

q1 → 4

q1 → 5, 4, 3, 2, 1

pop() → O(1)

push() → O(n)

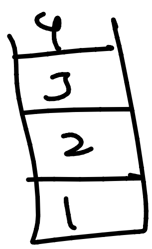
H.W → pop() → O(n)

push() → O(1)

---

q → 1, 2, 3, 4

q → 4, 3, 2, 1



1, 2, 3, 4

j = 1 → 2, 3, 4

q = 4, 3, 2, 1

j = 2 → 3, 4

q = 4, 3, 2

j = 3 → 4

q = 4, 3

j = 4

q = 4

q → 1, 2, 3, 4, 5, 6   K = 3

q → 1, 2, 3, 4, 5, 6    k = 3

q → 4, 5, 6

| |
|---|
| 3 |
| 2 |
| 1 |

q → 4, 5, 6, 3, 2, 1

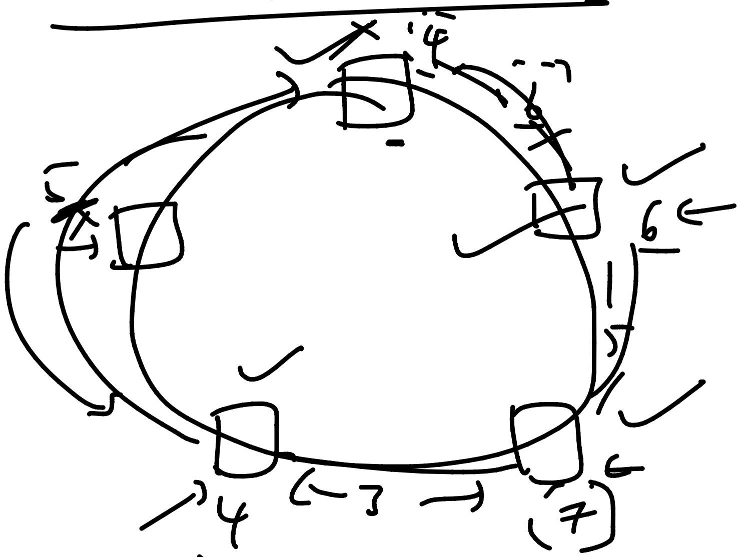
5, 6, 3, 2, 1, 4

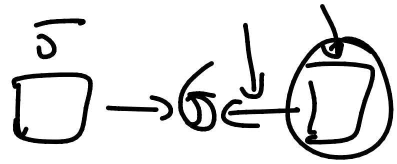
6, 3, 2, 1, 4, 5

3, 2, 1, 4, 5, 6

---

## Q/ Circular Tour




1 unit → 1 unit

$(6-5)+(7-3)+(4-5)$
$+(4-6)$

(8, 3)  (2, 4)

1 → 2 → 3 → 4 → 5 → 6 → 7

$p[1]-d[1] \geq 0$

$\underbrace{p[1]-d[1]+p[2]-d[2] \geq 0}$

$p[1]+p[2]+p[3]-d[1]-d[2]-d[3] \geq 0$

$p[i]-d[i] = x_i$

$x_1 \geq 0$

$x_1 + x_2 \geq 0$

$x_1 + x_2 + x_3 \geq 0$

$x_1 + x_2 + x_3 + x_4 \geq 0$

$x_1 + x_2 + x_3 + x_4 \geq 0$

$x_1 + x_2 \geq 0$

$x_1 + x_2 + x_3 + x_4 \geq 0$

$x_1 + x_2 + x_3 + x_4 + x_5 \geq 0$

$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq 0$

$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 < 0$

$x_2 + x_3 \text{ — } + x_4 = -11 < 0$

q → 0, 1, 2, 3, 4, 5

q → 6, 7, 8, 9, 0, 1, 2, 3, 4, 5

q → 6, 7

q → 8, 9, 0, 1, 2   curr < 0

---

## Deque :- Double ended queue

Here, you can insert as well as pop out elements at both ends.

| | |
|---|---|
| q.push_back(10) | q.size() → 3 |
| q.push_back(20) | q.front() → 5 |
| q.push_front(5) | q.back() → 20 |
| | q.pop_back() → |

q.pop-back() →
q.pop-front()

5 | 10 | 20

→ 5 | 10 → 10

1, 2, 7, 4, 5, 16   K=3

for(i=0 ; i<n ; i++)
{
for(j=i ; j<=i+K-1 ; j++)
{ max }
}

K-size subarrays?

n, 1 → n
2 → n-1
K → n-K+1

K(n-K+1) →

50

l   r

?

ans → 50

0, 1, 2, 3, 4, 5

max=0
for(i=0 ; i<n ; i++)
{
max= max(max, arr[i]);

}

⇒ → 1, 3, 2, 5, 1, 9, 7, 6, 10, 4
i=0 ... j ... 8

=) → $i=0$ 3, 2, 5, 1, 9, 7, 6, 10, 4

$K=4$

5, 9, 9, 9, 10, 10

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ~~3~~ | ~~2~~ | ~~5~~ | ~~1~~ | ~~9~~ | ~~7~~ | ~~6~~ | 10 | 4 |

3, 2, 1, 1, 0

5, 4

5, 1, 0, 0, 0

9, 7, 6, 3, 2

5, 4, 3, 2, 1

$K = 4, 4$

| | | | | |
|---|---|---|---|---|
| ~~5~~ | 4 | 3 | 2 | 1 |

decreasing

$i-K$

$S$ $i$

$i-K+1$

$i-S+1 = K$

$S = i-K+1$

7, 3, 2, 1, 6

$K=4$

| | | | |
|---|---|---|---|
| 7 | 3 | 2 | 1 |

→

| | |
|---|---|
| 7 | 6 |

$< 4$

$i=3$

1, 2, 3, 4, 5, 6