One good offer can change your life,

WHY WE NEED HASHING

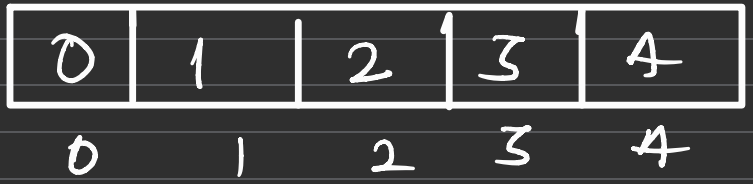
arr:

$x = 3$

O(1)

O(N)

arr



$x = 3$

$x = 1$

1008768908 1008768909 1008768910

WHAT IS HASHING

Hashing is a technique that efficiently stores and retrieve the data in a way that allow for the quick access.

We can perform the operation like insert, delete & update in O(1) time.

In hashing, we mainly convert large number into smaller number & we use the smaller number as a index to store the large number.

This can be done using the hash function

$$h(x) = x \bmod N$$

X = key which need to insert in the table
N = size of the hash table.

Keys = 3 9 14 25 36 2 1 47 18 20

N = 10

$h(x) = x \% N$

$3 \% 10 = 3$

$9 \% 10 = 9$

$14 \% 10 = 4$

$25 \% 10 = 5$

$36 \% 10 = 6$

$2 \% 10 = 2$

$1 \% 10 = 1$

$47 \% 10 = 7$

$18 \% 10 = 8$

$20 \% 10 = 0$

$x = 36$

Key: 3 9 14 25 36 2 74 46 18 20

N = 10

$h(x) = x \% N$

$3 \% 10 = 3$

$9 \% 10 = 9$

$14 \% 10 = 4$

$25 \% 10 = 5$

$36 \% 10 = 6$

$2 \% 10 = 2$

$74 \% 10 = 4$

$46 \% 10 = 6$

$18 \% 10 = 8$

$20 \% 10 = 0$

Collision

COLLISION TECHNIQUES

When two or more numbers get the same hash value, it called as collision.

This can be solved using two methods

- Separate chaining
- Open Addressing
  - linear probing
  - Quadratic probig
  - Double Hashing

Separate Chaining

Key = 3 9 14 25 74 2 36 18 46 20

N = 10

$h(x) = x \% N$

$3 \% 10 = 3$

$9 \% 10 = 9$

$14 \% 10 = 4$

$25 \% 10 = 5$

$74 \% 10 = 4$

$2 \% 10 = 2$

$36 \% 10 = 6$

$18 \% 10 = 8$

$46 \% 10 = 6$

$20 \% 10 = 0$

Linear probing

Key = 3 9 14 25 36 2 74 46 24 35

$$h(x) = (x + i) \% N \quad (i = 0 \ldots\ldots N-1)$$

$(3+0) \% 10 = 3$

$(9+0) \% 10 = 9$

$(14+0) \% 10 = 4$

$(25+0) \% 10 = 5$

$(36+0) \% 10 = 6$

$(2+0) \% 10 = 2$

$(74+0) \% 11 = 4$

$(74+1) \% 10 = 5$

$(74+2) \% 10 = 6$

$(74+3) \% 10 = 7$ ✗

$(46+0) \% 10 = 6$

$(46+1) \% 10 = 7$

$(46+2) \% 10 = 8$

9	9
46	8
74	7
36	6
25	5
14	4
3	3
2	2
35	1
24	0

$(24+0)\ \%10 = 4$

$(24+1)\ \%10 = 5$

$(24+2)\ \%10 = 6$

$(24+3)\ \%10 = 7$

$(24+4)\ \%10 = 8$

$(24+5)\ \%10 = 9$

$(24+6)\ \%10 = 0$

$(35+0)\ \%10 = 5$

$(36+6)\ \%10 = 1$

Quadratic probing

N = 10

Key: 3 9 14 25 36 2 74 46 18 20

$$h(x) = (x + i^2) \% N \quad (i = 0 \ldots (N-1))$$

$(3+0^2) \% 10 = 3$

$(9+0^2) \% 10 = 9$

$(14+0^2) \% 10 = 4$

$(25+0^2) \% 10 = 5$

$(36+0^2) \% 10 = 6$

$(2+0^2) \% 10 = 2$

$(74+0^2) \% 10 = 4$

$(74+1^2) \% 10 = 5$

$(74+2^2) \% 10 = 8$

$(46+0^2) = 6$

$(46+1^2) = 7$

$(18+0^2) \% 10 = 8$

$(18+1^2) \% 10 = 9$

$(18+2^2) \% 10 = 2$

$(18+3^2) \% 10 = 7$

$(18+4^2) \% 10 = 4$

$(18+5^2) \% 10 = 3$

$(18+6^2) \% 10 = 4$

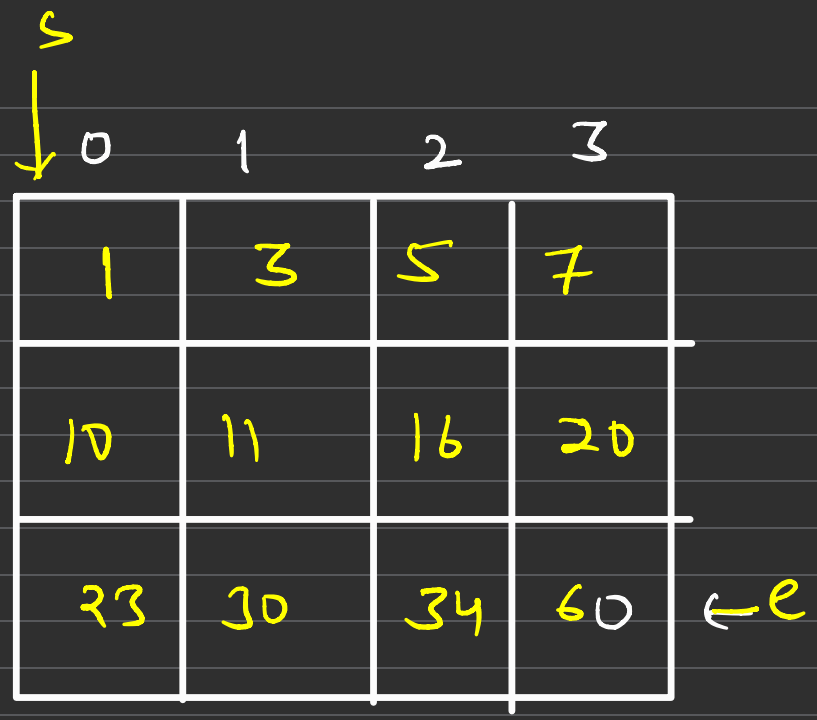
$(18+7^2) \% 10 = 7$

$(18+8^2) \% 10 = 2$

$(18+9^2) \% 10 = 9$

$(18+10^2) \% 12$

$(18+11^2) \% 10$



$s = 0, \quad e = (n \times m) - 1$

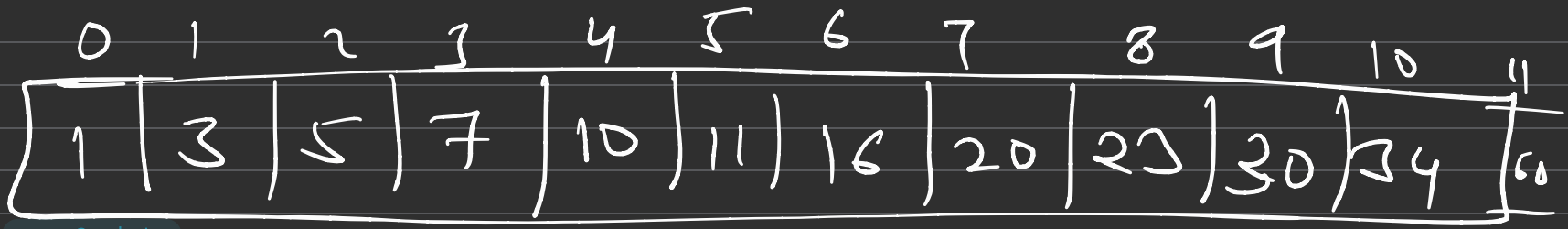
$mid = (s + e) / 2$

$row = mid / M$

$col = mid \% M$

$row = (5 / 4) = 1$

$col = (5 \% 4) = 1$



```cpp
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& mat, int target) {
        int n = mat.size();
        int m = mat[0].size();

        int start = 0, end = (n * m) - 1;
        while(start <= end){
            int mid = (start + end) / 2;
            int row = mid / m;
            int col = mid % m;

            if (mat[row][col] == target){
                return true;
            }

            if (mat[row][col] < target){
                start = mid + 1;
            }else{
                end = mid - 1;
            }
        }

        return false;
    }
};
```