Date : 28th - 09 - 2020
Morning Session : 9am – 11.00 PM
By ~ Rohan Kumar

Topics: Hashing Part-1

Hashing is an important Data Structure which is designed to use a special function called the Hash function which is used to map a given value with a particular key for faster access of elements. The efficiency of mapping depends on the efficiency of the hash function used.

We use hashing for search, when it comes to search there are two type of search

1) Linear Search
2) Binary Search

Time complexity for Linear Search O(n)
Time complexity for Binary Search O(logn)

```
dict1 = {}
dict1['a'] = 1
dict1['b'] = 2
dict1['c'] = 3
dict1
```

```
{'a': 1, 'b': 2, 'c': 3}
```

```
for i in dict1.keys():
    print(dict1[i])
```

```
1
2
3
```

```
for k,v in dict1.items():
    print(k,v)
```

```
a 1
b 2
c 3
```

```
keys = ['a','b','c']
values = [1,2,3]
hash = dict(zip(keys,values))
hash
```

```
{'a': 1, 'b': 2, 'c': 3}
```

An array is given and we have to find all the symmetric pairs?

```
def findPairs(arr,n):
    hash_ = dict() #empty dict
    for i in range(n):
        first = arr[i][0]
        sec = arr[i][1]
        if sec in hash_.keys() and hash_[sec] == first:
            print(sec,first)
        else:
            hash_[first] = sec

arr = [[0 for i in range(2)] for j in range(5)]
arr[0][0],arr[0][1] = 11,20
arr[1][0],arr[1][1] = 30,40
arr[2][0],arr[2][1] = 5,10
arr[3][0],arr[3][1] = 40,30
arr[4][0],arr[4][1] = 10,5

findPairs(arr,5)
```

Output:
30 40
5 10

Another problem

If 2 arrays are given in both arrays we have to find the elements which are present in array 1 not in 2 array.

```
def missing(arr1,arr2,n,m):
    dict1 = dict()
    for i in range(m):
        dict1[arr2[i]] = 1
    for i in range(n):
        if arr1[i] not in dict1.keys():
            print(arr1[i])

arr1 = [1,2,3,4,5,6]
arr2 = [2,3,1,0,5]
missing(arr1,arr2,6,5)
```

```
4
6
```

We have to find most frequent number

```
def mostFrequent(arr,n):
    dict1 = dict()
    for i in range(n):
        if arr[i] in dict1.keys():
            dict1[arr[i]] += 1
        else:
            dict1[arr[i]] = 1
    max_count = 0
    max_item = -1
    for i in dict1:
        if max_count < dict1[i]:
            max_count = dict1[i]
            max_item = i
    return max_item

arr = [1,3,2,1,4,1]
mostFrequent(arr,len(arr))
```

```
1
```

MCQ 1:

1. Why we use hashing.
    a. to sort efficiently
    b. to search efficiently

Answer: B, To Search Effeciently

MCQ 2:

2. Hashing data structure can be used implemented in python by using
    a. Dictionary
    b. Set

Answer: A, Dictionary

MCQ 3 :

3. Hashing data structure
    a. Uses lists to insert elements.
    b. Uses key,value to store elements

Answer: B, Uses key, value to store Elements.

MCQ 4:

4. Hashing technique takes space complexity of O(n)
    a. True
    b. False

Answer: A, True..

Resources:

https://www.gatevidyalay.com/hashing/

https://www.programiz.com/dsa/hash-table