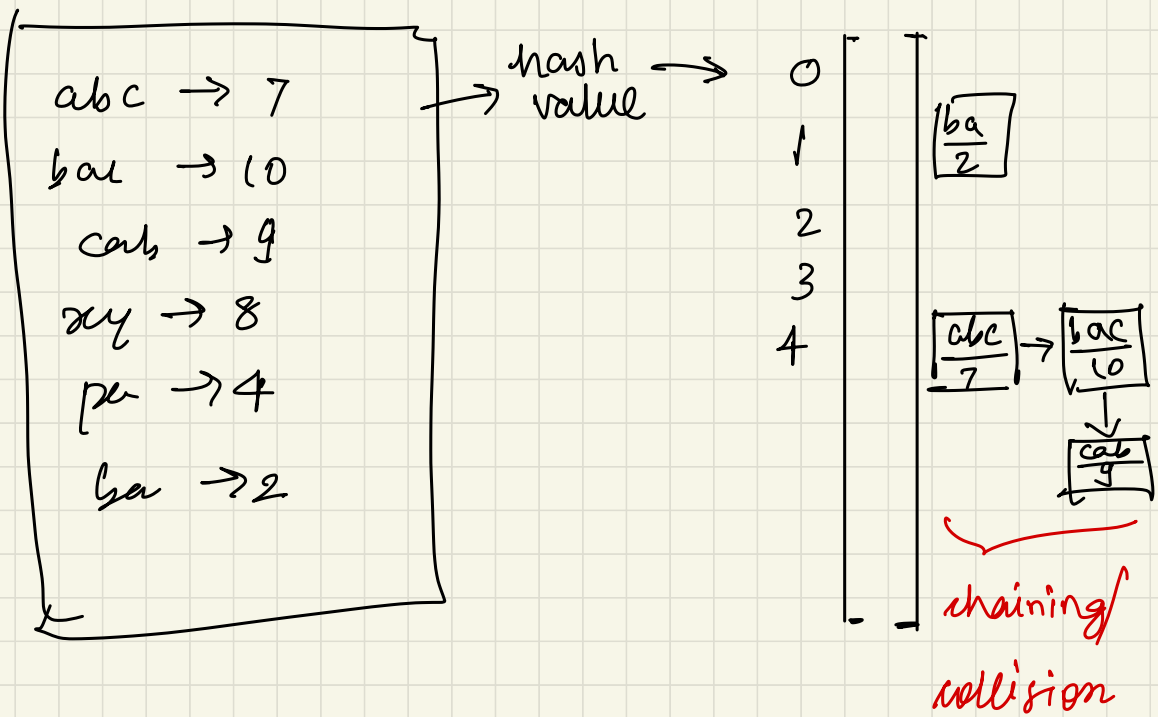
Lecture 14 : Hashing - 1

Hash maps : maps any keys to values

→ space-efficient solution to store K-V pair
→ fast access of data
→ check if a key is present
→ Insert K-V pairs
→ Update K-V pairs



Hash map should be able to

- map the values
- indexable based on keys
- get in O(1) time

(51) Given an array, you have Q queries. Return count of elements in array for each query

[1 1 10^9 0 10^9 0]

Q = [1 → [ ] int = 2

0 →

10^9]

Brute force

$O(N * Q)$

Using dictionary

```
freq = {}
for val in arr:
    if val in freq:
        freq[val] += 1
    else:
        freq[val] = 1
for query in Q:
    if query in freq:
        return freq[query]
    else:
        return 0
```

(52) Given an array, check if there exists a subarray with sum = 0

Brute Force

```
n = len(arr)
for i in range(n):
    for j in range(i, n):
        sum = 0
        for k in range(i, j+1):
            sum += arr[k]
        if sum == 0:
            return True
return False
```

$O(N^3)$

Using Prefix sum

```
n = len(arr)        <- calculate prefix sum
for i in range(n):
    for j in range(i, n):
        if ps[j] - ps[i-1] == 0
            return True
return False
```

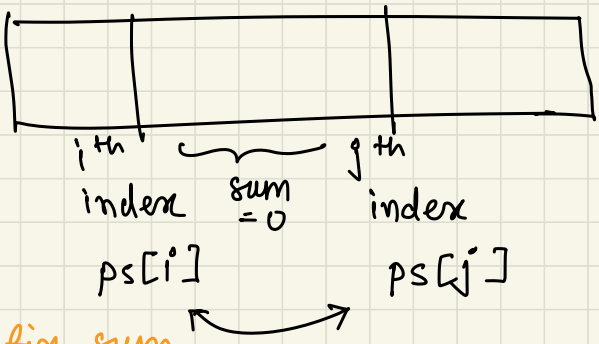
TC: $O(N^2)$ SC: $O(N)$

Using carry forward

```
n = len(arr)
for i in range(n):
    sum = 0
    for j in range(i, n):
        sum += arr[j]
        if sum == 0:
            return True
return False
```

TC: $O(N^2)$ SC: $O(1)$

Trick



Check if a prefix sum is repeated or not

```
sum = 0
freq-ps = {}
for i in range(n):
    sum += arr[i]
    if sum in freq-ps:
        return True
    freq-ps[sum] = 1
return False
```

TC: O(N)

SC: O(N)

easy code →

```
N = len(A)
ps = [0]*N
ps[0] = A[0]
for i in range(1,N):
    ps[i] = ps[i-1] + A[i]
if (0 in ps) or
   (len(set(ps)) != N):
    return 1
else:
    return 0
```

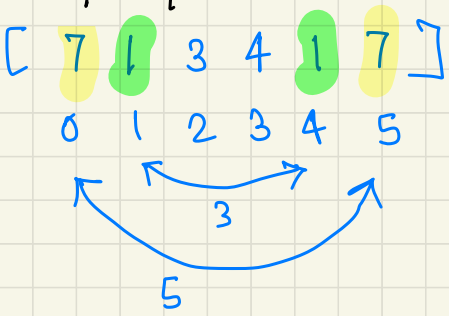
(53) Given an array A, find the first repeating element.

[10 5 3 4 3 5 6]

return the first occurence of repeating element otherwise return -1

```
N = len(A)
freq = {}
for i in A:
    if i in range(N):
        if A[i] in freq:
            freq[A[i]] += 1
        else:
            freq[A[i]] = 1
for e in A:
    if freq[e] > 1
        return e
return -1
```

(54) Given an array of N elements, call a pair of distinct indices in that array special if elements at those indices are equal. Find a special pair s.t. the distance b/w pair is minimum




```
min-dist = sys.maxsize
freq = {}

idx = 0
for i in A:

    if i not in freq.keys():
        freq[i] = idx

    else:
        min-dist = min(min-dist, idx - freq[i])
        freq[i] = idx

    idx += 1

if min-dist == sys.maxsize:
    return -1

else:
    return min-dist
```

(55) Largest Continuous Sequence Zero Sum

[1 2 -2 4 -4] ⟹ [2 -2 4 -4]

```
n = len(A)
mydict = {}
max = -sys.maxsize -1

ps = [0]*n   ps[0] = A[0]
for i in range(n):
    ps[i] = ps[i-1] + A[i]


for i in range(n):

    if (ps[i] in mydict) or (ps[i]==0):

        if (ps[i]==0):
            dist = i+1
            if dist > max:
                max = dist
                start = -1
                end = i
        else:
            dist = i - mydict[ps[i]]
            if dist > max:
                max = dist
                start= mydict[ps[i]]
                end =i
    else:
        mydict[ps[i]]=i
if max != -sys.maxsize-1:
    return A[start+1:end+1]
return []
```

(56) Given a string, check if the letters can be rearranged to create a palindrome

```
freq = {}
oddc = 0
s = set()

for i in range(n):
    if A[i] in freq:
        freq[A[i]] += 1
    else:
        freq[A[i]] = 1

for i in range(n):
    if freq[A[i]] % 2 == 0:
        continue
    elif A[i] in s:
        continue
    else:
        s.add(A[i])
        oddc += 1
if oddc > 1:
    return 0
else:
    return 1
```

(57) Given an array A, find if its COLORFUL or not.

3245 → 3, 2, 4, 5, 32, 24, 45, 324, 245

all of product of digits are different

```
A = str(A)
n = len(A)

check = set()

for i in range(n):
    prod = 1

    for j in range(i, n):
        prod = prod * int(A[j])

        if prod in check:
            return 0
        else:
            check.add(prod)
return 1
```