Day 14
23-02-2024

Hashing (Basics)

* Hashing is a process That maps data of any size (input) to a fixed-size value, known as a hash code, using a function called Hash function.

* The primary purpose of Hashing is to enable fast data retrieval, particularly in hash Tables where the Hashcode determines the index at which data is Stored or found

* This Technique is widely used in various Computer science applications, including Database indexing, caching and ensuring data integrety.

Key points about Hashing:

1. Hash function: Transforms any size input into a fixed-size byte string, uniquely identifying the data.

2. Efficiency: ensures quick computation, Storage, and retrieval of data

3. fixed-size output: produces a constant output size for any input, ensuring predictable storage.

4. Collision Handling: manages identical outputs for different inputs using methods like chaining and open addressing

5 use cases: utilized in data retrieval, encryption, compression and integrity checks

6. **Deterministic**: Generates consistent Hashcodes for the same input every time.

7 **Sensitivity**: minor input changes result in significantly different Hash codes.

8 **Security**: Designed to be Secure and collision resistant for cryptographic purposes.

9. **Hash Tables**: efficiently stores key-value pairs, with the hashcode dictating storage location.

Example problem count frequency in a range:

input n=6 arr = [1,3,1,9,2,7]

output: [2,1,1,0,0,0]

Table shows Thier counts

number	counts
1	2
2	1
3	1
4	0
5	0
6	0

Approach To Question:

Step 1: initialize a frequency array with n elements all initially set to 0

```
int[] frequency = New int[n];
```

Step 2: iterate over each element in the input array.

```
for (int i=0; i< arr.length; i++){
    int element = arr[i];
```

Step 3 check if element is within the range 1 to n

```
    if (element >=1 && element <=n){

    // increment the frequency count of the element

    // Subtract 1 To match the elements's value's
    To index position in the frequency array.

    frequency [element - 1]++;
```

Step 4: return the frequency

Same approach using HASHMAP:

```
    // initialize a Hashmap to Store the frequency of each element
    MAP<Integer, Integer> frequencyMap = new HashMap<>();

    // iterate over each element in the input array.
      for (int i=0; i<arr.length; i++) {
            int num = arr[i]; // access the element at index i
    // check if the element is within range 1 to n
        if (num>=1 && num<=n) {
         // update the frequency count in the map
        frequencyMap.put (num, frequencyMap.getOrDefault (num, 0) +1)
        }
      }
```

```
//initialize the frequency array to return the result.

int[] frequency = new int [n];
// populate the frequency array using the map

for (int i =1; i<=n; i++){
    frequency[i-1] = frequencymap.getOrDefault(i,0);
}
return frequency
```