Hashing & Hash Maps in DSA: Beginner to Advanced Guide

Subtitle: Learn how to efficiently store, retrieve, and manipulate data using hash maps and hashing techniques.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app

Introduction

Hashing is a technique to store and access data in **constant time (O(1))** on average. Hash maps (or dictionaries) are widely used in coding interviews and real-world applications for **frequency counts, lookup, and caching**. This guide explains patterns, examples, and exercises.

Step 1: What is Hashing?

- Map a **key → value** for efficient lookup
- Use a **hash function** to convert key to an index
- Collisions handled via **chaining or open addressing**

Example: Simple JS object as hash map

```
let map = {};

map["apple"] = 5;

map["banana"] = 3;

console.log(map["apple"]); // 5
```

Step 2: Key Hash Map Operations

- **Insert / Update:** map[key] = value
- **Search / Lookup:** value = map[key]
- **Delete:** delete map[key]
- **Check existence:** key in map or map.hasOwnProperty(key)

Step 3: Common Hashing Patterns

1. Frequency Counting

- Count occurrences of elements

```
let arr = [1,2,2,3,1,4];

let freq = {};
```

```
for (let num of arr) freq[num] = (freq[num]||0)+1;

console.log(freq); // { '1': 2, '2': 2, '3': 1, '4': 1 }
```

2. Two Sum / Pair Problems

- Check if complement exists in map

```
let nums = [2,7,11,15], target = 9;

let map = {};

for (let i=0; i<nums.length; i++) {

  if (map[target-nums[i]] !== undefined) console.log([map[target-nums[i]], i]);

  map[nums[i]] = i;

}
```

3. Subarray Sum Problems

- Prefix sums + hash map to find count of subarrays with sum = k

4. Longest Substring with K Distinct Characters

- Sliding window + hash map to store frequencies

Step 4: Advantages of Hashing

- Lookup, insert, delete in **O(1) on average**
- Useful for **frequency, pair, and subarray problems**
- Combines well with **sliding window, two pointers, and dynamic programming**

Step 5: Common Mistakes

- Forgetting to handle **collisions** (for custom hash)
- Using hash map incorrectly in loops → overwriting counts
- Ignoring edge cases like **empty array or zero sum**
- Using hash maps in **ordered problems** without extra care

Step 6: Practice Plan for Hashing

Day 1:

- Frequency count, simple lookup → 5 problems

Day 2:

- Two-sum and pair sum → 5–7 problems

Day 3:

- Subarray sum / prefix sum → 5 problems

Day 4:

- Longest substring / distinct characters → 5 problems

Day 5–6:

- Mixed hashing problems → 10 problems

Mini Exercises

1. Find majority element in an array
2. Count pairs with given difference
3. Longest consecutive sequence in array
4. Subarray with sum zero

Key Takeaways

- Hashing provides **efficient storage and retrieval**
- Hash maps are crucial for **frequency and lookup problems**
- Combine with **sliding window and prefix sums** for complex problems
- Mastering hash maps improves **coding interview problem-solving speed**
- Practice helps **recognize patterns quickly**

Visit haas.dev for more DSA hashing guides, problem sets, and interview preparation resources.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app