Hashing in DSA: Beginner to Advanced Guide

Subtitle: Learn how to use hash maps and sets to solve problems efficiently and improve coding interview performance.

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

Introduction

Hashing is a technique used to store and retrieve data **efficiently**. Hash maps and sets help in solving problems like frequency counting, duplicates, and lookup operations in **constant time**. This guide teaches hashing patterns, implementation, and common problems.

Step 1: What is Hashing?

Hashing maps **keys** to **values** using a hash function.

- **Hash Table** → underlying data structure
- **Hash Map** → key-value storage
- **Hash Set** → stores only unique keys

Example in JS:

```
let map = new Map();

map.set("a", 1);

map.set("b", 2);

console.log(map.get("a")); // 1
```

Step 2: Basic Operations

Operation	Example	Time Complexity
Insert	map.set(key, value)	O(1)
Search	map.has(key) / map.get(key)	O(1)
Delete	map.delete(key)	O(1)
Iterate	for (let [k,v] of map)	O(n)

Step 3: Hashing Patterns

1. Frequency Counting

- Count occurrences of elements in array or string

```
let arr = [1,2,2,3,1];
let freq = new Map();

for (let num of arr) {
  freq.set(num, (freq.get(num) || 0) + 1);
}
console.log(freq);
```

2. Two Sum / Pair Problems

- Use map to check complements in O(n) time

```
function twoSum(nums, target) {
  let map = new Map();
  for (let i = 0; i < nums.length; i++) {
    if (map.has(target - nums[i])) return [map.get(target - nums[i]), i];
    map.set(nums[i], i);
  }
}
```

3. Subarray Problems (Prefix Sum + Hashing)

- Count subarrays with sum = k

```
let count = 0, sum = 0;
let map = new Map();
map.set(0, 1);

for (let num of arr) {
  sum += num;
  if (map.has(sum - k)) count += map.get(sum - k);
  map.set(sum, (map.get(sum) || 0) + 1);
}
```

Step 4: Hash Set Usage

- Store **unique elements**
- Check duplicates quickly

```
let set = new Set([1,2,3,2]);
console.log(set.has(2)); // true
console.log(set); // {1,2,3}
```

Step 5: Important Problems You Must Practice

1. **Two Sum / Three Sum / Four Sum**
2. **Find first non-repeating character**
3. **Subarray sum equals K**
4. **Longest consecutive sequence**

```
let nums = [100,4,200,1,3,2];

let set = new Set(nums);

let longest = 0;



for (let num of set) {

  if (!set.has(num - 1)) {

    let current = num;

    let streak = 1;

    while (set.has(current + 1)) {

      current++;

      streak++;

    }

    longest = Math.max(longest, streak);

  }

}
```

Step 6: Common Mistakes

- Using arrays instead of hash maps → O(n^2) solutions
- Not handling duplicate keys properly
- Forgetting to check **base cases**
- Misusing hash set for value counting

Step 7: Practice Plan for Hashing

Day 1:

- Frequency counting + simple hash map problems → 5 problems

Day 2–3:

- Two sum / pair problems → 5–7 problems

Day 4–5:

- Subarray problems using prefix sum + hashing → 5–7 problems

Day 6:

- Longest consecutive sequence + mixed problems → 5–7 problems

Day 7:

- Mixed hashing problems + revision → 10 problems

Mini Exercises

1. Find duplicates in an array
2. Count occurrences of each word in a string
3. Longest substring without repeating characters using set
4. Intersection of two arrays

Key Takeaways

- Hashing allows **constant-time access and storage**
- Use hash maps for **key-value problems**
- Use hash sets for **uniqueness and lookup**
- Learn common patterns: **frequency count, two-sum, prefix sum**
- Practice to recognize when to **apply hashing efficiently**

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