DSA Notes: Hashing in C++

1. Basics of Hashing

Hashing is a technique to map data to a fixed-size value (hash code), which is used to index into a hash table.

Hash Function:

- Example: h(key) = key % size
- Should minimize collisions

Hash Table:

- Stores key-value pairs for fast operations
- C++: unordered_map, unordered_set

Collision Handling:

- Chaining: store multiple elements using list/vector
- Open Addressing: linear probing, quadratic probing, etc.

Load Factor:

- Load Factor = entries / table size

Time Complexities:

Operation	Avg Case	Worst Case
Insert	O(1)	O(n)
Search	O(1)	O(n)
Delete	O(1)	O(n)

Applications:

- Frequency count
- Remove duplicates
- Caching (e.g., LRU)
- Dictionary implementation

2. unordered_map and unordered_set in C++

```
unordered_map<int, int> mp; // Key -> Value mapping
unordered_set<int> st; // Unique elements
```

Example: Count Frequencies

```
vector<int> nums = {1, 2, 2, 3};
unordered_map<int, int> freq;
for (int x : nums) freq[x]++;

// freq = {1:1, 2:2, 3:1}
```

3. LeetCode Problems with Explanation

Problem: Two Sum (LeetCode #1)

Given array nums and target, return indices of two numbers adding to target.

Approach:

- Use unordered_map to store (number -> index)
- For each num, check if (target - num) exists

Code:

```
unordered_map<int, int> mp;
for (int i = 0; i < nums.size(); i++) {
    int diff = target - nums[i];
    if (mp.count(diff)) return {mp[diff], i};
    mp[nums[i]] = i;
}
```

Problem: Intersection of Two Arrays (LeetCode #349)

Return intersection of nums1 and nums2

DSA Notes: Hashing in C++

Approach:

- Insert nums1 into unordered_set
- Check if nums2[i] exists in set

Code:

```
unordered_set<int> s(nums1.begin(), nums1.end());
unordered_set<int> res;
for (int x : nums2)
    if (s.count(x)) res.insert(x);
```

Problem: Longest Consecutive Sequence (LeetCode #128)

Find length of the longest sequence of consecutive integers.

Approach:

- Insert all elements in unordered_set
- For each number, if num-1 not in set -> start of sequence

Code:

```
unordered_set<int> s(nums.begin(), nums.end());
int longest = 0;
for (int num : s) {
    if (!s.count(num - 1)) {
        int current = num;
        int streak = 1;
        while (s.count(current + 1)) {
            current++;
            streak++;
        }
        longest = max(longest, streak);
    }
}
```