C++ STL Topics Frequently Asked in Interviews

In interviews, questions about the C++ Standard Template Library (STL) often focus on the conceptual understanding of STL
containers, algorithms, iterators, and functions, as well as practical problem-solving skills. Here's a breakdown of
commonly asked STL-related topics:

Containers

1. Vector (std::vector)
   - Internal implementation: dynamic arrays, resizing mechanism.
   - Operations: insertion, deletion, access, resizing complexity.
   - Comparison with std::array and std::deque.

2. Set and Multiset (std::set, std::multiset)
   - Implementation: Red-Black Tree.
   - Time complexity of insert, erase, and find.
   - Difference between std::set and std::unordered_set.

3. Map and Multimap (std::map, std::multimap)
   - Implementation: Red-Black Tree.
   - Key-value pair storage and iteration order.
   - Difference from std::unordered_map.

4. Unordered Containers (std::unordered_map, std::unordered_set)
   - Implementation: Hash tables.

- Hash collisions and resolution strategies (e.g., chaining, open addressing).
- Complexity: average-case vs worst-case performance.

5. Deque (std::deque)
- Implementation: segmented memory structure.
- Difference from std::vector (efficient push/pop at both ends).

6. Priority Queue (std::priority_queue)
- Implementation: Binary heap.
- Use of std::vector as the underlying container.

Iterators

- Types of iterators: input, output, forward, bidirectional, and random access.
- Iterator invalidation rules for different containers after insert/erase operations.
- Custom iterators: how to implement and use them.

Algorithms

- Frequently used algorithms from <algorithm>:
  - Sorting: std::sort, std::stable_sort.
  - Searching: std::binary_search, std::lower_bound, std::upper_bound.
  - Modifying sequences: std::reverse, std::rotate, std::remove, std::unique.
  - Heap operations: std::make_heap, std::push_heap, std::pop_heap, std::sort_heap.

- Custom comparators in algorithms:
  - Writing lambdas or function objects for sorting.

- Efficiency considerations:

- Time complexity of common algorithms like std::sort.

Other Key Areas

1. String (std::string)

- Comparison with std::vector<char>.
- Common functions like substr, find, replace.
- Performance concerns with concatenation.

2. Smart Pointers (std::shared_ptr, std::unique_ptr, std::weak_ptr)

- Understanding of ownership models and memory management.

3. Function Objects and Lambdas

- Writing custom functors and lambda expressions.
- Passing them to STL algorithms.

4. Concurrency

- std::thread, std::mutex, and other components from <thread> and <atomic> libraries.

Sample Questions

1. Vector vs. List:

- When would you use std::vector vs. std::list? Explain the trade-offs.

2. Custom Comparators:

- Write a program to sort a list of pairs based on the second element.

3. Set and Multiset:

- Given an array, find the k largest elements using an STL container.

4. Map and Unordered Map:

- How would you find the frequency of elements in an array using an STL map?

5. Algorithm Analysis:

- Explain the difference between std::sort and std::stable_sort.

6. Iterator Invalidation:

- Explain what happens to iterators of a std::vector when elements are inserted.

To succeed in interviews, focus on conceptual clarity, use cases, and hands-on practice with STL through platforms like

LeetCode, Codeforces, or HackerRank.