LeetCode: A Forge for Sharper Coders

LeetCode offers a vast collection of coding challenges categorized by difficulty, topic, and company tags. It provides a valuable environment to:

* Develop Algorithmic Thinking: Practice applying efficient algorithms to solve diverse problems.
* Master Data Structures: Gain proficiency in utilizing various data structures for optimal solutions.
* Challenge Yourself: Tackle problems of increasing complexity, pushing your coding boundaries.

Problem 1: Longest Substring Without Repeating Characters (Medium)

Problem Description:

Given a string, find the length of the longest substring that consists of unique characters (no repeated characters).

Approach:

Leverage a sliding window technique with two pointers: left and right. Utilize a set to keep track of characters encountered within the current window.

1. Initialize left and right pointers to 0.
2. Maintain a unordered\_set named charSet to track unique characters within the current window.
3. While right pointer moves through the string:
   * If the character at right is not in the charSet, add it and update the maximum window length (maxLength) if needed.
   * Otherwise, remove the character at left from the charSet and move the left pointer forward.
4. Return the maxLength found.

C++ Solution:

C++

#include <iostream>

#include <unordered\_set>

#include <string>

using namespace std;

int lengthOfLongestSubstring(string s) {

if (s.empty()) {

return 0;

}

int left = 0;

int right = 0;

int maxLength = 0;

unordered\_set<char> charSet;

while (right < s.length()) {

char ch = s[right];

if (charSet.count(ch) == 0) {

charSet.insert(ch);

maxLength = max(maxLength, right - left + 1);

right++;

} else {

charSet.erase(s[left]);

left++;

}

}

return maxLength;

}

int main() {

string str = "abcabcbb";

int length = lengthOfLongestSubstring(str);

cout << "Length of longest substring without repeating characters: " << length << endl;

return 0;

}

Explanation:

1. We include necessary libraries for string manipulation (string) and unordered sets (unordered\_set).
2. The lengthOfLongestSubstring function takes a string s as input and returns the length of the longest unique character substring.
3. We handle empty strings by returning 0.
4. We initialize left and right pointers to 0, maxLength to 0, and an unordered\_set charSet to store unique characters.
5. We iterate through the string using a while loop with right pointer incrementing.
6. We check if the current character (ch) is not present in the charSet. If not, we add it and update maxLength if the current window length surpasses the previous maximum.
7. If a duplicate character is encountered, we remove the character at left from the charSet and move left forward to shrink the window.
8. Finally, the maxLength is returned, representing the length of the longest unique character substring.

Problem 2: Subsets (Medium)

Problem Description:

Given a set of distinct integers, generate all possible subsets (including the empty set).

Approach:

Utilize backtracking to recursively explore all possible combinations of elements.

1. Define a helper function generateSubsets that takes the current subset and the remaining elements as arguments.
2. Add the current subset to the result list.
3. Iterate through the remaining elements:
   * Create a new subset by including the current element.
   * Recursively call generateSubsets with the new subset and the remaining elements (excluding the current element).
4. Return the result list containing all generated subsets.

C++ Solution:

C++

#include <iostream>

#include <vector>

using namespace std;

void generateSubsets(vector<int>& subset, vector