CeTu - technical assessment

C++ Software engineer

# Hashmap

1. Describe what is the hashmap / hash table data structure.  
   Give an example for when the structure is useful.

**Answer:**

A hashmap is a data structure that stores key-value pairs, allowing for fast retrieval, insertion, and deletion of values based on their keys. It uses a hash function to map keys to indices in an internal array, enabling near constant time complexity, O(1), for these operations in average(amortized for insert).

A hashmap is ideal when you need fast lookup, insertion, and deletion operations for key-value pairs. I always use a hashmap when working with key-value data unless I specifically need the elements to be ordered. In such cases, I prefer using std::map or similar data structures that maintain order.

Examples:

1. Suppose you're analyzing text and need to count how many times each word appears. A hashmap is ideal for storing words as keys and their frequencies as values.
2. A hashmap is great for storing settings as key-value pairs, where the key is the setting name, and the value is the setting's value.
3. Explain what is the complexity of each method (e.g O(1), O(n), etc).

**Answer:**

**erase:**

Average case: O(1)

Worst case: O(n), where n is the number of elements

Worst case occurs when there are many collisions and elements are chained

**insert:**

Amortized case: O(1)

Worst case: O(n)

The amortized O(1) takes into account occasional rehashing operations

Rehashing itself is O(n) but happens so infrequently that when distributed across all insertions, the amortized cost per operation remains constant

**lookup:**

Average case: O(1)

Worst case: O(n)

Worst case happens when all elements hash to the same bucket

**Notes:**

The O(1) average case is achieved with a good hash function and appropriate table size. Load factor management and rehashing are crucial for maintaining efficiency. The O(n) worst case is rare in practice with proper implementation. The term "amortized" is particularly relevant for insert operations due to the occasional rehashing. For both erase and lookup, we typically discuss average and worst case rather than amortized complexity as they don't trigger structural changes to the table

# 

# Past Projects

Please describe one major project you had, what issues did you encounter, how did you approach solving those, and how did you analyze the issue. You can obfuscate parts that might be covered by NDA, and refer to different names or terms.

**Answer:**

### **Task Description 1:**

In a scenario involving text recognition on a document and saving it as a PDF with a text layer, add the ability to overlay recognized text onto the original image.

### **Image Processing Workflow:**

Preprocessing -> Text Recognition -> Export to PDF.

In the current version, the image is preprocessed at the very beginning, and the subsequent stages receive a non-original image (it may undergo multiple transformations and differ significantly from the original). Preprocessing is essential for high-quality text recognition and cannot be disabled. All libraries written in C++.

### **Challenges and Actions:**

1. **Text Rendering Library Enhancements**
   * There was a need to add functionality to the text rendering library to apply various transformations in a short timeframe.
   * Recognized text had to be adjusted to align with the original image, not the preprocessed one.
2. **PDF Export Enhancements**
   * It was necessary to quickly study the PDF export library and add the ability to export the original image instead of the preprocessed one.
3. **Parallel Image Processing**
   * Added the capability to pass the original image from the Preprocessing stage to the Export stage alongside the preprocessed image.
   * This required studying the architecture of both our company's desktop products and cloud infrastructure (where separate pipeline stages were implemented as microservices, with data transmitted over the network).
4. **Lack of Expertise on Legacy Libraries**
   * There was no one to seek help from regarding the rendering and export libraries, as the developers who wrote them had already left the company.

### **Outcome:**

In approximately three months, I went from analyzing the task and having no prior knowledge of some of the libraries to completing the task and presenting the results to the stakeholders.

### **Task Description 2:**

Developed a new technology for detecting and analyzing tables in payment documents. Was responsible for the infrastructure (a C++ library integrated into the payment document processing pipeline) written in C++. Over approximately six months, progressed from studying scientific publications to delivering the first release version.