

*Technical Report*

MAP vs HASHMAP  
C++

2021/01/28

Brycen Co., Ltd

# Lịch sử sửa đổi

|  |  |  |  |
| --- | --- | --- | --- |
| Phiên bản | Ngày | Nội dung sửa đổi | Người viết |
| 1.0 | Jan 28, 2021 | - Phiên bản đầu tiên | Tran Van Quang Huy |
| 1.1 | Jan 30, 2021 | -Viết các khái niệm về Map và HashMap | Tran Van Quang Huy |
| 1.2 | Jan 31,2021 | -Nêu sự khác nhau giữa Map và HashMap  -Kết luận | Tran Van Quang Huy |

# Mục lục

[Lịch sử sửa đổi 2](#_Toc63078833)

[Mục lục 3](#_Toc63078834)

[Danh mục hình ảnh 4](#_Toc63078835)

[Giới thiệu 5](#_Toc63078836)

[1.Map (std::map) là gì ? 6](#_Toc63078837)

[1.1 6](#_Toc63078838)

[1.2 Lớp pair 6](#_Toc63078839)

[2. HashMap (std::unorder\_map) là gì? 6](#_Toc63078840)

[3. Sự khác nhau giữa Map (std::map) và và HashMap (std::unorder\_map) trong C++ 7](#_Toc63078841)

[3.1 Thời gian tìm kiếm một phần tử 8](#_Toc63078842)

[3.2 Thứ tự các phần tử 8](#_Toc63078843)

[3.3 Thời gian chèn và xóa phần tử 8](#_Toc63078844)

[3.4 Sử dụng bộ nhớ 8](#_Toc63078845)

[4. Kết luận 10](#_Toc63078846)

[4.1 Khi nào nên sử dụng Map (std::map) ? 10](#_Toc63078847)

[4..2 Khi nào nên sử dụng HashMap (std::unorder\_map) 10](#_Toc63078848)

[Tài liệu tham khảo 11](#_Toc63078849)

# Danh mục hình ảnh

[Hình 1. Hash Table 6](#_Toc63077579)

[Hình 2. Bảng tóm tắt sự khác nhau giữ Map (std::map) và HashMap (std::unorder\_map) [1]. 7](#_Toc63077580)

[Hình 3. Thời gian tìm kiếm một phần tử [2]. 8](#_Toc63077581)

# Giới thiệu

Tài liệu báo cáo về các cấu trúc dữ liệu Map và HashMap (unorder\_map) trong C++:

1. Map là gì ?
2. HashMap là gì ?
3. Sự khác nhau giữ Map và HashMap trong C++
4. Kết luận

# 1.Map (std::map) là gì ?

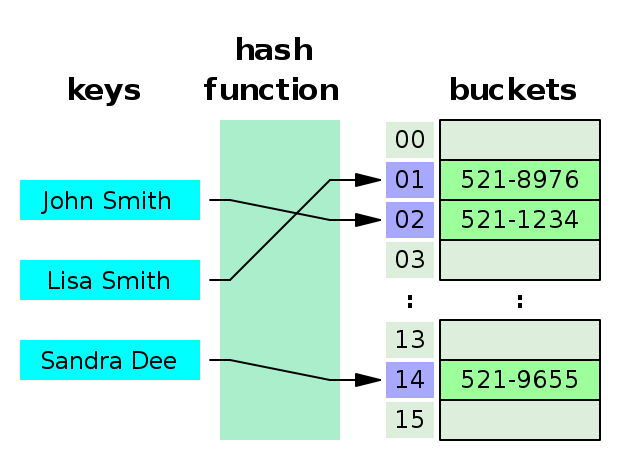
Map là cấu trúc dữ liệu ánh xạ giữa cái được gọi là khóa (key) sang giá trị của khóa đó (gọi là value)

## 1.1

## 1.2 Lớp pair

# 2. HashMap (std::unorder\_map) là gì?

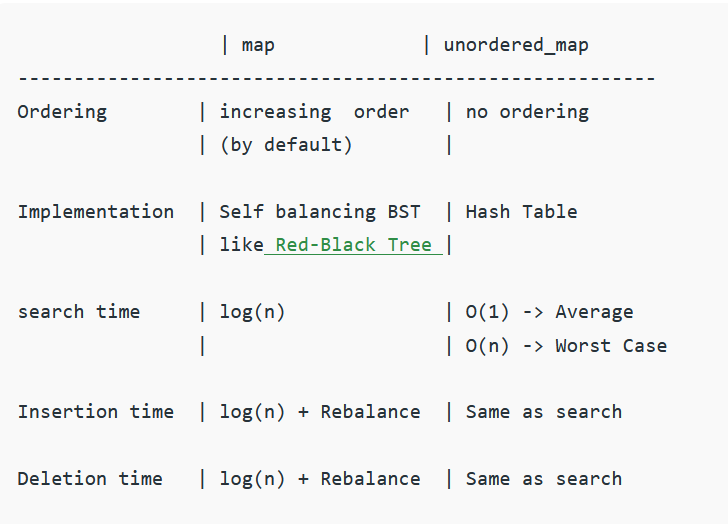
Là Map nhưng được cài đặt dựa trên nguyên lý Hashing



Hình . Hash Table

# 3. Sự khác nhau giữa Map (std::map) và và HashMap (std::unorder\_map) trong C++

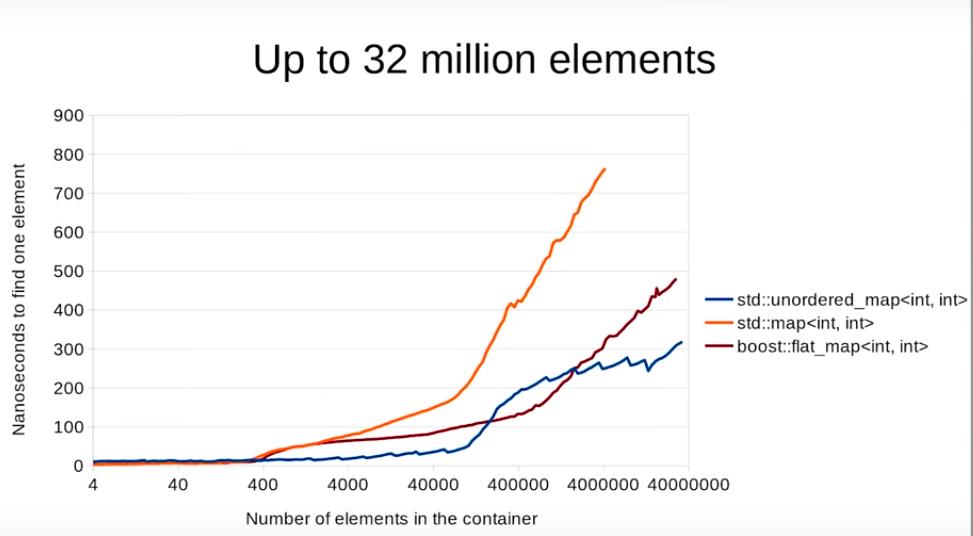
Dưới đây là bảng mô tả nhưng điểm khác nhau giữa 2 kiểu cấu trúc dữ liệu Map (std::map) và HashMap (std::unorder\_map):



Hình . Bảng tóm tắt sự khác nhau giữa Map (std::map) và HashMap (std::unorder\_map) [1].

Sự khác nhau giữa BST (chính xác là balanced BST) và Hash Table là yếu tố chính tạo nên sự khác biệt giữa Map và HashMap khi đưa ra quyết định sử dụng cho những trường hợp cụ thể.

## 3.1 Thời gian tìm kiếm một phần tử



Hình . Thời gian tìm kiếm một phần tử [2].

Khi Map tìm kiếm một phần tử, thì cần phải duyệt qua các phần tử khác và với triển khai cấu trúc BST nên thời gian bỏ ra khi tìm kiếm trên BST là O(log(n)), nhanh hơn nhiều so với cách tìm kiềm tuần tự. Riêng đối HashMap, việc sử dụng Hash Table giúp việc tìm kiếm một phần tử đạt thời gian trung bình là O(1) thông qua key mà không cần phải đi duyệt qua các phần tử khác. Tuy nhiên, nếu Hash Table không được triển khai tốt, hàm Hash không tối ưu thì xuất hiện những vấn đề như: thời gian thực thi hàm Hash, thời gian so sánh các key bị trùng, … dẫn đến trường hợp tệ nhất là cần O(n) thời gian tìm kiếm.

## 3.2 Thứ tự các phần tử

Map (std::map) được triển khai với cấu BST nên các phần tử đã được sắp xếp trong suốt quá trình lưu trữ dữ liệu (chèn hoặc xóa) mà vẫn đảm bảo thứ tự các phần từ. Nên có sự liên kết thông tin giữa các phần tử nhằm không phá cấu trúc của BST. Còn đối với HashMap (std::unorder\_map), đúng như tên gọi, các phần tử không được sắp xếp theo thứ tự.

## 3.3 Thời gian chèn và xóa phần tử

Map (std::map) sử dụng cấu trúc BST nên mỗi lần chèn và xóa phần tử cần mất một lượng thời gian O(log(n)) và sau đó sẽ phải cân bằng lại cây. Còn đối với HashMap (std::unorder\_map) chỉ hash key là có thể truy cập trực tiếp đến phần tử cần tìm nên thời chèn hoặc xóa chỉ mất khoảng thời gian lý tưởng là O(1).

## 3.4 Sử dụng bộ nhớ

Hash Table sẽ có những chỗ trống chưa được lấp đầy vì các key được hash không trả về giá trị liên tục nhau, nên sẽ chiếm bộ nhớ không cần thiết. Còn đối với BST, thì bộ nhớ sử dụng tương ứng với số lượng phần tử. Do đó, lưu trữ dữ liệu bằng Map (std::map) sẽ có lợi về mặt bộ nhớ hơn HashMap (std::unorder\_map)

# 4. Kết luận

## 4.1 Khi nào nên sử dụng Map (std::map) ?

* Muốn dữ liệu của mình được sắp xếp.
* Truy cập dữ liệu của phần tử theo thứ tự.
* Khi muốn sử dụng ít bộ nhớ hơn để lưu trữ dữ liệu lớn.

## 4..2 Khi nào nên sử dụng HashMap (std::unorder\_map)

* Khi muốn tìm kiếm cũng như thêm và xóa nhanh mà không quan tâm đến thứ tự các phần tử.
* Lưu trữ liệu lớn và muốn thời gian tìm kiếm và cập nhật dữ liệu nhanh.

Cả hai cấu trúc dữ liệu trên đều giúp tổ chức và lưu trữ dữ liệu hiệu quả, nên sử dụng một cách hợp lý để đạt được hiệu năng tốt nhất cho từng trường hợp. Đối với dữ liệu nhỏ (dưới một triệu phần tử trở xuống) thì sự khác biệt về tốc độ tìm kiếm thực sự không nhiều nên ta có thể sử dụng Map (std::map) hoặc HashMap (std::unorder\_map). Còn đối với dữ liệu lớn thì ta cần đánh đổi giữa tốc độ tìm kiếm và bộ nhớ, nếu muốn tiết kiệm bộ nhớ thì hãy sử dụng Map (std::map), còn nếu muốn tốc độ thì sử dụng HashMap (std::unorder\_map).

# Tài liệu tham khảo

1. https://www.geeksforgeeks.org/map-vs-unordered\_map-c/
2. <https://www.youtube.com/watch?v=M2fKMP47slQ>
3. <https://vi.wikipedia.org/wiki/C%C3%A2y_AVL>
4. <http://mrpear.com/cplusplus/su-dung-hieu-qua-stdmap-vs-stdunorder_map/>