



競プロのための標準 C++



311

- 01 <string>
- 02 <vector>
- 03 <numeric>
- 04 <unordered_set>
- 05 <algorithm>
- 06 <tuple>
- 07 <ios>, <iomanip>
- 08 Union-Find
- 09 重み付き Union-Find
- 10 最小全域木 / 最大全域木
- 11 トポロジカルソート
- 12 最長増加部分列 (LIS)
- 13 Binary Indexed Tree (BIT)
- 14 ベルマンフォード法
- 15 ワーシャルフロイド法
- 16 ダイクストラ法
- 17 C++ 標準入出力の高速化**
- 18 本書の執筆を応援する
- 19 謝辞

Chapter 17 無料公開

C++ 標準入出力の高速化



Ryo Suzuki

2023.01.10に更新

このチャプターの目次



- 1. C++ 標準入出力の高速化のテンプレート

C++ の標準入出力を高速化する方法の説明です。

- C 言語の入出力関数 (`printf()`, `scanf()` など) を使用していない
- 複数のスレッドから C++ 入出力ストリームを使用していない

以上を満たすプログラムであれば、入出力を行う前に `std::cin.tie(0)->sync_with_stdio(0);` を呼ぶことで、安全に標準入出力のオーバーヘッドを削減し、プログラムの実行時間を短縮できます。

- 参考:
 - [<ios>, <iomanip> | 3.1 C 言語の入出力ストリームとの同期を無効にする](#)



競プロのための標準 C++

311

- 01 <string>
- 02 <vector>
- 03 <numeric>
- 04 <unordered_set>
- 05 <algorithm>
- 06 <tuple>
- 07 <ios>, <iomanip>
- 08 Union-Find
- 09 重み付き Union-Find
- 10 最小全域木 / 最大全域木
- 11 トポロジカルソート
- 12 最長増加部分列 (LIS)
- 13 Binary Indexed Tree (BIT)
- 14 ベルマンフォード法
- 15 ワーシャルフロイド法
- 16 ダイクストラ法
- 17 C++ 標準入出力の高速化**
- 18 本書の執筆を応援する
- 19 謝辞

1. C++ 標準入出力の高速化のテンプレート

```
#include <iostream>

int main()
{
    std::cin.tie(0)->sync_with_stdio();
}
```

▼ これは次のコードと同じです。

```
#include <iostream>

int main()
{
    std::cin.tie(nullptr);
    std::ios_base::sync_with_stdio(false);
}
```

PREV

[← ダイクストラ法](#)

NEXT

[本書の執筆を応援する →](#) GitHubで編集を提案