自動でメモ化を行うライブラリ

noshi91

2021 年 2 月 25 日

1 概要

一定の条件を満たす関数を自動でメモ化してくれる C++ のライブラリを作成したので、紹介します。メモ化再帰の実装を簡単に行うことが出来るようになります。

2 実装

https://github.com/noshi91/blog/blob/master/codes/auto_memoization.cpp

3 使い方

EDPC–C の解答例を示します。

https://atcoder.jp/contests/dp/tasks/dp_c

https://atcoder.jp/contests/dp/submissions/20480799

```
/* ここにライブラリを貼る */
#include <algorithm>
#include <iostream>
#include <vector>

int main() {
  int N;
  std::cin >> N;

  // 0-indexed
  std::vector<std::array<int, 3>> a(N);
  for (int i = 0; i < N; i++) {
    for (int j = 0; j < 3; j++) {
      std::cin >> a[i][j];
    }
  }
```

```
  // dp(i, k) = i 日目に活動 k を行った直後の最大幸福度
  auto dp = memoized([&](auto &&dp, int i, int k) -> int {
    if (i == -1) {
      return 0;
    }

    int ans = 0;
    for (int j = 0; j < 3; j++) {
      if (j != k) {
        ans = std::max(ans, dp(i - 1, j) + a[i][k]);
      }
    }
    return ans;
  });

  std::cout <<
    std::max({dp(N - 1, 0), dp(N - 1, 1), dp(N - 1, 2)}) << "\n";
}
```

auto f = memoized([&](auto &&f, 引数) -> 返り値の型 {関数の実装}); のように書くことで、引数に対して自動的にメモ化する関数 f を作ることが出来ます。呼び出しの際の引数と同じ引数で以前に関数が実行されていた場合、その時の結果を直ちに返します。関数の実装部分でも f を呼び出すことが可能です(再帰)。

4 時間計算量

関数を呼び出した際の引数が今までに無いものだった場合、それをメモの無い呼び出しと呼ぶことにします。メモのない呼び出しは関数の中身が実行されるため、関数自体の時間計算量が掛かります。それに加えて、メモの有無に関わらず、全ての呼び出しに付き $\mathrm{O}(\log(N))$ の時間計算量が掛かります。N は関数の実行が完了した時点でのメモの無い呼び出しの総回数です。

より正確には、引数を operator< で比較するのに掛かる時間計算量を c とすれば、1 回の関数呼び出しに付き $\mathrm{O}(c\log(N))$ 掛かります。従って、引数の数が多い場合や、引数に std::vector などが含まれる場合、遅くなります。

5 注意事項

- メモ化は std::map により行っています。従って、operator<を実装していない引数があると使うことが出来ません。ただし、第一引数の auto &&f は除きます。
- 引数の型は T または const T &の形のものにしてください。T &や T &&だとコンパイルエラー、また

は誤動作の可能性があります。

- 返り値の型が `void` の関数は扱えません。
- 第一引数の `auto &&f` を除き、引数に `auto` を使うことはできません。
- 機能を絞ってあるので、`mutable` 指定されたラムダ式が扱えません。その他にも、使い方の節で説明した用法から外れる場合、上手く動作しない可能性があります。
- 返り値の型を指定しないとコンパイルが通らないことが多いと思います。