modint 構造体の紹介

noshi91

2020 年 3 月 30 日

1　概要

modint の利点や実装例を示す。

2　modint とは

競技プログラミングにおいて、値を 1000000007 (あるいは他の素数) で割った余りを求める問題は頻出です。これらの問題ではアルゴリズムの過程で繰り返し剰余を取りますが、modint は普通の整数型などと同じ感覚で扱うだけで自動的に剰余を取る構造体です。

3　使用例

単純ですが、以下の問題を解いてみることにします。
a, b, c, d が与えられます。$(a * b + c - d) \bmod M$ (M は定数) を出力してください。

- modint なし

```
long long a, b, c, d;
cin >> a >> b >> c >> d;
cout << ((a * b % M + c) % M - d + M) % M;
```

- modint 使用

```
modint<M> a, b, c, d;
cin >> a.v >> b.v >> c.v >> d.v;
cout << (a * b + c - d).v;
```

エクサウィザーズ 2019 を modint を使用してコーディングしました。

- D Modulo Operations
  https://atcoder.jp/contests/exawizards2019/submissions/11366384
- E Black or White
  https://atcoder.jp/contests/exawizards2019/submissions/11367016

4　利点

- 可読性の向上
  % M などにコードのロジック部分が覆い隠されなくなり、バグの低減などが見込めます。
- 速度の向上
  加減算は条件分岐を用いることで剰余計算を行わない高速化が可能です。modint なら実装の内部でそのような高速化を自由に掛けることが出来ます。コード本体に直接高速化を入れると、可読性の著しい低下などが懸念されます。
- mod 取り忘れがなくなる
  mod を取り忘れて値が正しい領域を外れることがなくなります。
- 分数がデバッグしやすくなる
  using mint = modint¡M¿; などとしておけば、double に書き換えることでそのまま分数計算などのデバッグが可能です (場合によっては少しだけ入出力を書き換えるかもしれませんが)。
- 他のライブラリに適用しやすい
  例えば累乗を計算するとき、modpow(a, b, M) といった関数を用意しなくとも、template を用いて一般化した pow 関数に modint をそのまま適用することで使用可能になります。他にも行列ライブラリなどに適用可能です。

5　欠点

- ライブラリがコード全体に占める割合が増える
  コンパクトにまとめたい場合、使用しない operator などは冗長になります。
- アルゴリズムの性質に依存した高速化が行えない
  64bit を超えないギリギリまで剰余を取らずにおく、0 にならないことが分かっているので符号反転で条件分岐を行わない、などの切り詰めた高速化は不可能になります。

6　実装例

自由にご使用ください

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

利便性のため、v は public 指定しています (本来は private にするのが良いと思います)。記事に掲載するにあたって、機能はかなり絞りました。足りないと思った部分は是非満足が行くようにカスタマイズしてください。

7　追加機能例

- 各種演算子 (operator++, operator–, operator-(単項), operator== 等)
- 逆元を返す関数

- std::cin, std::cout (あるいは自前の入出力) への対応
- 負数に対応したコンストラクタ
- 内部を 32 bit 整数で保持する
- pow を組み込む
- 0 除算が起きたときに assert や例外の送出などを行う
- 法が大きすぎるときに static_assert を行う、あるいは __int128 を使う

8　実行時に法が決まるとき

法が入力で与えられる場合などは、上記の modint を使用することはできません。そのような状況で使用できる実装は以下のようになります。

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

定数に関する最適化や諸々の恩恵を受けられなくなりますので、使用できる限りは先に示した方の実装を使用した方がよいと考えています。

9　終わりに

提案や異論は広く募集しています。