累積和は 0-indexed と半開区間で理解できる

noshi91

2020 年 5 月 11 日

1　概要

0-indexed と半開区間は競技プログラミングにおいてメジャーな概念である。累積和は 1-indexed でないと考えづらいという話を耳にするため、0-indexed 半開の枠組みで十分合理的に説明できることを示す。また、接頭辞ではなく接尾辞の和を考えるバリエーションも示す。

2　累積和とは

長さ n の列 a を考える。a の要素は一般に群を考えることが出来るが、単に整数とする。

長さ $n+1$ の数列 s を以下のように定義する。

$$s_r = \sum_{i \in [0,r)} a_i$$

すると任意の区間 $[l,r) \subseteq [0,n)$ に対して

$$\sum_{i \in [l,r)} a_i = s_r - s_l$$

が成立するため、a の区間和が $O(1)$ で計算可能である。また、$0 \leq r < n$ に対して

$$s_{r+1} = s_r + a_r$$

が成立するため、s も $O(n)$ で計算可能。

3　接尾辞の累積和

a を破壊的に変更して (in-place に) 累積和を構成したい場合も頻出である。基本的には以下のようなコードを用いたいが、これは正しく計算されない。

```
for (int i = 0; i < n; ++i) {
    a[i + 1] += a[i];
}
```

もしこのアルゴリズムを用いるなら、a_i は a[i + 1] に入っている必要がある。

そのような場合、s の定義を

$$s_l = \sum_{i \in [l,n)} a_i$$

とするとよい。普通の累積和が接頭辞の累積和であるのと比べると、これは接尾辞の累積和である。このとき

$$\sum_{i \in [l,r)} a_i = s_l - s_r$$

が成立するため、同様に $O(1)$ で計算できる。

a[i] $= a_i$、a[n] $= 0$ とすると、以下のアルゴリズムで in-place に累積和を計算可能である。

```
for (int i = n; i > 0; --i) {
    a[i - 1] += a[i];
}
```

4 実装例

接頭辞の累積和: `https://github.com/noshi91/blog/blob/master/codes/cumsum_prefix.cpp`
接尾辞の累積和: `https://github.com/noshi91/blog/blob/master/codes/cumsum_suffix.cpp`