

# ABC037 解説

平成 28 年 5 月 11 日

## 1 A: 饅頭

この問題では 2 種類の物が買えるが、一定の価格の中で、なるべく多くの個数を買うには、このうち安い方のみを買える限り買うのが最適である。

1 個  $A$  円の饅頭と 1 個  $B$  円の饅頭があり、このうち安い方の値段を  $s$  円とすると、手持ちの  $C$  円で買える饅頭の個数は、 $C/s$  円となる。ただし、 $/$  とはあまりを切り捨てるときの割り算の商を意味する。例えば  $C++$  では、 $C/s$  と書くのと同じである。

## 2 B: 編集

1 次元配列のある範囲に値を代入することでこの問題を解くことができる。

まず、長さ  $N$  の 1 次元配列  $a[]$  を用意する。最初、全ての要素に 0 を代入する (配列を用意しただけで最初に 0 が代入されるプログラミング言語もある)。

$Q$  個の操作を順に処理していく。それぞれの処理においては、配列の  $L_i$  番目から  $R_i$  番目に for 文でループを回すことで、 $T_i$  を代入する。

最後に全ての要素を順に出力する。

## 3 C: 総和

$N - K + 1$  個の連続する長さ  $K$  の部分列の和を for ループで  $K$  個の値を足すことで計算しようとする時、最悪  $O(N^2)$  かかり ( $K = N/2$  の時を考えてみよう)、これは  $N \leq 10^3$  のときは問題ないが、満点を取ろうとすると実行時間制限に間に合わないので、高速化する必要がある。

一つの方法としては、 $S_i = a_1 + a_2 + \dots + a_i$  となる  $S_i$  を計算しておくものがある。これは、 $S_{i+1} = S_i + a_{i+1}$  という式に従って合計  $O(N)$  で求めることができ、 $i$  番目の要素から  $j$  番目の和は  $S_j - S_{i-1}$  に等しいので、一つの連続する範囲について和を  $O(1)$  で計算することができる。

他にも、それぞれの要素がいくつの「長さ  $K$  の連続する部分列」に含まれるかを計算し、それらに各  $a_i$  を掛けて足すことで答えを求めるというものもあるが、これは場合分けが若干難しくなる。

## 4 D: 経路

$f(i, j) := i$  行  $j$  列からスタートする経路の総数とする。求めたい答えは、全てのマスに対する  $f(i, j)$  の和となる。この値は、 $f(i-1, j), f(i+1, j), f(i, j-1), f(i, j+1)$  のうち、その行き先のマスが存在していて、そこに移動できるマスの合計に 1 (このマスから始まり、移動せずに終了する経路に対応する) を加えたものである。

上の  $f(i, j)$  は、再帰的にコードを書くことで計算することができる。このとき、無限に再帰が深くなることはない。なぜなら、移動先のマスに書かれたのは移動前のマスに書かれた値よりも大きく、これらの値の種類数は最大でも  $H * W$  種類しかなく、これより多い回数大きくするという事はできないからである。

しかし、このままでは非常に遅い。ここで、いつ計算しても  $f(i, j)$  の値は不変であるので、一度この値を計算したら配列 ( $dp[i][j]$  とする) にこの答えを記憶しておき、二回目以降  $f(i, j)$  が呼ばれた時はこの値を参照してすぐ return をすることで、高速化をすることが可能である。全体で、計算量は  $O(HW)$  となる。この方法はメモ化再帰と呼ばれており、これは動的計画法の一種である。

またこの問題では入力サイズが大きいので、入力が遅い言語では高速な入力を使うのが望ましい。例えば、C++では cin, cout ではなく scanf, printf を使うべきである。