ACPC Day 3
C : mod reap 解説

原案、解説 : N_hara
Tester : N_hara, monkukui, pitsu, rsk0315

問題概要

N 個の正整数 $A_1, \dots, A_N$ から 0 個以上の値を選びその和を S とします。

また関数 $f(S)$ を

$$f(S) = \left\lfloor \frac{S}{K} \right\rfloor - (S \ (mod\ K))$$

と定義します。(※1)

このとき、$f(S)$ の最大値を求めてください。

$1 \leq N \leq 5 \times 10^4$

$1 \leq K \leq 100$

$1 \leq A_i \leq K$

(※1) $\lfloor a \rfloor$ は a を超えない最大の整数としています。

解答方針

基本方針 : DP（動的計画法）

（知らない人は、「DP 競技プログラミング」などで検索してみましょう。）

この問題では、取りうる S の値が分かればよいので、例えば

$dp[i][j] \equiv A_1, \ldots, A_i$ から 0 個以上の値を選んで和を j にできるか(true or false)

というような DP が考えられます。

（後述しますが、実はこの DP ではうまくいきません。理由を考えてみましょう。）

解答方針（つづき）

前ページの定義に従うと、このときの更新式は

$dp[i][j+A_i] = dp[i-1][j+A_i] \mid dp[i-1][j]$

(| は論理和をとる演算子)

となります。

解答方針(つづき)

このときの計算量は $\mathcal{O}(N\sum_i A_i)$ となります。

(※ $0 \leq S \leq \sum_i A_i$ より)

$\sum_i A_i \leq NK$ であることを考慮すると、

$N\sum_i A_i \leq N^2K \leq 2.5\times 10^{11}$

となり、実行時間制限内に解くのは厳しいです。

実行時間制限内に解くには、計算量を落とす必要があります。

想定解

ここで、$f(S)$ の式を見てみると以下のようになっています。

$$f(S) = \left\lfloor \frac{S}{K} \right\rfloor - (S\ (mod\ K))$$

すると、以下の式が成り立つことが分かります。

$$f(S+K) - f(S)$$

$$= \left\{ \left\lfloor \frac{S+K}{K} \right\rfloor - \big((S+K)(mod\ K)\big) \right\} - \left\{ \left\lfloor \frac{S}{K} \right\rfloor - \big(S\ (mod\ K)\big) \right\}$$

$$= \left\{ \left\lfloor \frac{S}{K} \right\rfloor + 1 - \big(S(mod\ K)\big) \right\} - \left\{ \left\lfloor \frac{S}{K} \right\rfloor - \big(S\ (mod\ K)\big) \right\}$$

$$= 1 \qquad \therefore f(S+K) > f(S)$$

想定解（つづき）

先程の式より

$f(S) < f(S+K) < f(S+2K) < \cdots$

となることが分かります。

このことから、取りうる S の値を集合 $0, 1, \ldots, K-1$ の K 個の集合に分けた

（集合 i $(0 \le i \le K-1)$ には $S \pmod K = i$ なる S の値が含まれる）とき、

各集合内で $f(S)$ の最大値を取る可能性があるような S の値は高々 1 個（各集合内で最大の S ）しかないことが分かります。

想定解（つづき）

これを踏まえて先程の DP の更新式を改良すると、

$dp[i][j] \equiv A_1, \ldots, A_i$ から 0 個以上の値を選んでできる和 T_i の中で、$T_i = j \ (mod\ K)$ となるもののうち最大の値 T に対する $\left\lfloor \frac{T}{K} \right\rfloor$（存在しない場合は -1 ）

と定義でき、更新式は

$dp[i][(j + A_i)(mod\ K)] = \max(dp[i-1][(j + A_i)\ mod\ K]\ , dp[i][j] + \left\lfloor \frac{j+A_i}{K} \right\rfloor)$

となります。

このとき計算量は $\mathcal{O}(NK)$ となり、実行時間制限内に解くことができました。

おまけ

この問題を作っていた当初は

$1 \leq N \leq 2 \times 10^3$

$1 \leq K \leq 2 \times 10^3$

$1 \leq A_i \leq K$

という制約で問題を作成していましたが、実は「bitset高速化」と呼ばれる方法を使うことで

最初に紹介した計算量 $\mathcal{O}(N \sum_i A_i)$ の解法でも正解となる場合があるため、制約を変更しました。

おまけとして、この「bitset高速化」についての説明をします。

bitset高速化

いくつかの言語には bitset と呼ばれるクラスが定義されており、

これにより複数ビットのビット列が表現できます（10^5 ビットほどの長さでも使えます）。

この問題では、例えば以下のように bitset 高速化を行います。

ビット列 B の長さを $(\sum_i A_i + 1)$ とし、Bのjビット目が$dp[i][j]$に対応するようにします。

（$dp[i][j] \equiv A_1, \dots, A_i$から 0 個以上の値を選んで和を j にできるかどうか、と定義していました）

そして、更新のときは B と、B を A_i ビットだけシフトしたビット列で論理和をとり、得られるビット列を新しい B とするとこれで更新が行えます。

bitset高速化（つづき）

先ほどの更新操作は、ちょうど最初の方針で示した更新式に対応しています。

（参考）$dp[i][j+A_i] = dp[i-1][j+A_i] \mid dp[i-1][j]$

Bitset高速化を用いた場合、計算量のオーダーは最初の方針と同じ $\mathcal{O}(N \sum_i A_i)$ ですが

64 倍程度高速になるため当初の制約では間に合う場合があります。

（参考）

$N = 2000, K = 2000$のとき $\frac{1}{64} N^2 K \approx 1.3\times10^8$

$N = 5\times10^4, K = 100$のとき $\frac{1}{64} N^2 K \approx 3.9\times10^9$