AGC 037 解説

yutaka1999

2019 年 08 月 17 日

A: Dividing a String

解法 1

文字列 S に対して、求める答えを $f(S)$ で表すことにします。まず次の性質を示します。

$$f(s_1s_2\dots s_k) \leq f(s_1s_2\dots s_ks_{k+1})$$

Proof. $S = s_1s_2\dots s_k, K = f(s_1s_2\dots s_k)$ として、S の K 個の分割で条件をみたすものを $S = S_1S_2...S_K$ と置きます。ここで、各 s_i は一つの文字を、各 S_i は一つの文字列を表していますが、s_i によってその文字一つのみからなる文字列を表すこともあるので注意してください。$S_k \neq s_{k+1}$ の場合は $s_1s_2\dots s_{k+1} = S_1S_2...S_Ks_{k+1}$ が条件をみたす分割であるから確かに $K \leq f(s_1s_2\dots s_ks_{k+1})$ が成り立ちます。$S_k = s_{k+1}$ の場合を考えましょう。$K = 1$ の場合は明らかなので、$K \geq 2$ とします。このとき $S_{K-1} \neq S_Ks_{k+1}$ ならば $s_1s_2\dots s_{k+1} = S_1S_2...S_{K-1}(S_Ks_{k+1})$ という分割により $K \leq f(s_1s_2\dots s_ks_{k+1})$ となります。一方で $S_{K-1} = S_Ks_{k+1}$ の場合は $s_{k-2} = s_{k-1} = s_k = s_{k+1}$ であるから、$s_1s_2\dots s_{k+1} = S_1S_2...S_{K-2}(s_{k-2})(s_{k-1}s_ks_{k+1})$ もしくは $s_1s_2\dots s_{k+1} = S_1S_2...S_{K-2}(s_{k-2}s_{k-1}s_k)(s_{k+1})$ のいずれかの分割が条件をみたします。よってこの場合も $K \leq f(s_1s_2\dots s_ks_{k+1})$ であるから、示されました。 □

では $f(S)$ を実際に求めていきましょう。$f(i) = f(s_1s_2\dots s_i)$ とします。$i \leq 2$ に対しては容易に求まるので $i \geq 3$ のときを考えます。

まず、$s_i \neq s_{i-1}$ の場合は先の性質により $f(i) = f(i-1) + 1$ と分かります。$s_i = s_{i-1}$ の場合を考えましょう。$s_1s_2\dots s_i$ の条件をみたす分割を $S_1S_2...S_I$ とおくと $S_{I-1} = S_I = s_i$ となることはないので、$|S_{I-1}| + |S_I| \geq 3$ となります。これより $S_1S_2...S_{I-2} = s_1s_2\dots s_j$ $(j \leq i-3)$ と分かるから $I - 2 \leq f(j) \leq f(i-3)$ です。よって $I \leq f(i-3) + 2$ であるから $f(i) \leq f(i-3) + 2$ です。また $f(i) \geq f(i-3) + 2$ であることは、$s_1s_2\dots s_{i-3}$ の条件をみたす $f(i-3)$ 個への分割を取り、$s_{i-2}s_{i-1}s_i$ をうまく 2 分割して先の分割と連結すると、$s_1s_2\dots s_i$ の $f(i-3)+2$ 個への分割が取れることから分かります。よって、この場合は $f(i) = f(i-3) + 2$ です。

以上より漸化式が求まったので、$f(i)$ がすべて求まりこの問題が解けます。計算量は $O(N)$ です。

解法 2

上の解法からも分かるように、実は条件をみたす分割のうち分割個数が最大となるようなものを考えると、各 S_i の長さは高々定数（実際には 2）で抑えられます。この事実を用いると、動的計画法により $O(N)$ で答えを求められます。

こちらの解法の方がコンテスト中には思いつきやすいかと思います。

B: RGB Balls

$\sum_j (c_j - a_j)$ の最小値をまず求めましょう。R, G, B それぞれの色について、その色を持つボールの番号を小さい順に $r_1 < r_2 < \ldots < r_N, g_1 < g_2 < \ldots < g_N, b_1 < b_2 < \ldots < b_N$ として、$m_i = \min(r_i, g_i, b_i), M_i = \max(r_i, g_i, b_i)$ としましょう。このとき $\min \sum_j (c_j - a_j) = \sum_i (M_i - m_i)$ であることが示せます。まずこれを示しましょう。

まず、人に適当に順序を与えて、$a_1 < a_2 < \ldots < a_N$ となるようにします。このとき各 i に対して、番号が a_i より小さいボールがどの人に配られているかを考えてみましょう。すると $i \leq j$ のとき $a_i \leq a_j < b_j < c_j$ であるので、番号が a_i より小さいボールは人 1 から人 $(i-1)$ のいずれかに配られていることが分かります。特に番号が a_i より小さいボールには各色のボールは高々 $i-1$ 個しかないことが分かり、$a_i \leq r_i, g_i, b_i$ つまり $a_i \leq m_i$ と分かります。よって、$\sum_j -a_j \geq \sum_i -m_i$ が常に成立すると分かります。同様にして、人に適当な順序を再び与えて、$c_1 < c_2 < \ldots < c_N$ となるようにすれば、$c_i \geq M_i$ と分かり $\sum_j c_j \geq \sum_i M_i$ が成り立ちます。よって、確かに $\sum_j (c_j - a_j) \geq \sum_i (M_i - m_i)$ となりますが、この等号が成立するようにボールを分配することができるのは明らかです。（人 i にはボール r_i, g_i, b_i を配ればよいです。）以上により $\sum_j (c_j - a_j)$ の最小値が求まりました。

では $\sum_j (c_j - a_j)$ が最小になる配り方の個数はどのようにして求めればよいでしょうか。$\sum_j (c_j - a_j) \geq \sum_i (M_i - m_i)$ を示す過程で、$a_i \leq m_i$ や $c_i \leq M_i$（この 2 つの式における人の順序は異なることに注意して下さい）といった不等式を用いましたが、これらがすべて等号成立することが $\sum_j (c_j - a_j) \geq \sum_i (M_i - m_i)$ の等号成立条件です。つまり、$A = \{m_i | 1 \leq i \leq N\}, C = \{M_i | 1 \leq i \leq N\}, B = \{1, 2, \ldots, 3N\} \backslash (A \cup C)$ としたときに、以下の条件をみたすようにボールを配る方法を数えればよいです。

- j 番目の人が受け取ったボールの番号を小さい順に $a_j < b_j < c_j$ とする。
- このとき $a_j \in A, b_j \in B, c_j \in C$ となる。

これは B に属する各ボールに対して、それより番号の小さい A に属するボールとそれより番号の大きい C に属するボールを割り当てる方法を求めればよいので、簡単に数えることができます。よって、この問題は $O(N)$ で解くことができました。ただし、人の区別をするので、答えに $N!$ をかけるのを忘れないよう気を付けてください。

C: Numbers on a Circle

操作を逆方向に考えてみましょう。すると以下のような問題になります。

言い換え

i 番目の数は今 B_i です。そこで以下の操作を繰り返して i 番目の数を A_i としたいです。最小で何回の操作が必要でしょうか。

- $1 \leqq i \leqq N$ なる整数 i を一つ選ぶ。
- $i-1, i, i+1$ 番目の数をそれぞれ a, b, c としたとき、i 番目の数を $b-a-c$ に置き換える。

上の問題と同様に、$1 \leqq i \leqq N$ なる整数 i を一つ固定して、$i-1, i, i+1$ 番目の数をそれぞれ a, b, c とします。ここで与えられる数がすべて正であるので、操作を行っている間もすべての数が正である必要があることに注目します。このとき次のことが分かります。

- i 番目の数を変更するためには $b > a + c$ が必要である。
- その条件が成立している間、i 番目以外の数に対してどのように操作を行ったとしても a, c の値は変化しない。
- よって、i 番目の数を変更する必要があり、かつ $b > a + c$ をみたしているならば、他の操作を行う前に i 番目の数を変更する操作を行ってよい。
- つまり、$b > a + c$ かつ変更する必要のある i を探し、その i に対して操作を行うことを繰り返してよい。

特に条件をみたすように操作を繰り返したとき、かかる操作回数は常に一定であることが分かります。ただし、操作回数自体はとても大きくなる可能性があるので、愚直にシミュレーションするのでは間に合いません。そこで、以下のような工夫が必要です。

- $b > a + c$ かつ変更する必要のある i を探し、その i に対して b が A_i と一致するまでもしくは $b \leq a + c$ となるまで i **に対して繰り返し**操作を行う。

このように行うことにするとシミュレーションにかかるステップ数が十分少なくなります。というのも、b が A_i と一致するように操作を行うのは合計で高々 N 回であり、$b \leq a + c$ となるまで操作を行うのは、行うごとに b の値が半分以下になるため、合計で高々 $N \log M_B$ 回（M_B は B_i の最大値）であるからです。（ただし、$x \geq y$ に対して x を y で割ったあまりが $\frac{x}{2}$ 以下であることを用いました。）よって、後は先程記した操作をシミュレーションする方法を考えればよいです。一例として以下のように行うのがよいでしょう。

- 優先度付きキュー等を用いて、変更する必要がある数の内、最大の数を取る。
- その数に対して先程の操作を行う。

以上によりこの問題が解けました。計算量は $O(N \log N \log M_B)$ となります。

D: Sorting a Grid

可能な手順が3段階あるので、このような問題ではまず条件を整理することが重要です。手順3を終えた時点でみたしているべき条件が与えられているので、手順$k(k \geq 1)$を終えた時点でみたしているべき条件というのをkが大きい方から順に整理していきましょう。すると以下のようになります。

- 手順2を終えた時点で、上からi行目は$M \times (i-1)+1$〜$M \times (i-1)+M$からなる。
- 手順1を終えた時点で、各列は以下のように構成されている。
  - 1以上M以下の数が1つ
  - $M+1$以上$2M$以下の数が1つ
  - ...
  - $MN-M+1$以上MN以下の数が1つ

条件は整理できたので、実際に手順1をどのように行えばよいかを考えましょう。ここで以下のような二部グラフを考えましょう。

- $2N$頂点NM辺からなり、各頂点は$S_1, S_2, \ldots, S_N, T_1, T_2, \ldots, T_N$とラベル付けされている。
- 1からNMの各数jについて、jが属する行をi行目とする。
- このとき、jに対応する辺e_jが頂点S_iと頂点$T_{[\frac{j+M-1}{M}]}$を繋いでいる。

手順1を終えた時に、各列に属する数に対応する辺が完全マッチングをなすことが条件です。つまり、先の二部グラフをM個の完全マッチングに分割することができればよいのです。ここで、先の二部グラフの次数はすべて等しくMであることに注目しましょう。実は、ホールの定理を用いるとすべての次数が等しい二部グラフは常に完全マッチングを持つことが比較的容易に示せます。この事実を用いると、適当な完全マッチングを見つけ、それを除いたグラフで完全マッチングを見つける、ということを繰り返すことで手順1の行い方が分かります。計算量は一つのマッチングを見つけるのに$O(NM\sqrt{N})$であるので$O(NM^2\sqrt{N})$です。

E: Reversing and Concatenating

S に現れる文字の中で辞書順最小のものを a とします。最終的に S の先頭に a を何文字連続させることができるかを考えてみましょう。最初の操作での U において a が最大 L 文字連続しているとすると、実は S の先頭に並べることができる a の個数は最大で $2^{K-1}L$ 個であり、実際に $2^{K-1}L$ 個並ぶように操作をする方法は限られていることが示せます。

まず高々 $2^{K-1}L$ 個しか a を並べられないことを示しましょう。これは操作を一回行った後の S において a が連続する個数は高々 L であり、操作をもう一度行っても S において a が連続する個数は高々2倍にしかならないことから分かります。では実際に $\min(N, 2^{K-1}L)$ 個 a を連続させられることを示しましょう。i 回目の操作において、以下のように操作を行えば実際に a をその個数分先頭に並べられます。

- $i < K$ のとき U において a が $2^{i-1}L$ 個連続する部分が末尾となるように次の S を選ぶ。
- $i = K$ のとき U において a が $2^{i-1}L$ 個連続する部分が先頭となるように次の S を選ぶ。

よって、$2^{K-1}L \geq N$ の場合は a を N 個並べた文字列が答えとなります。そうでない場合は、最終的に a が $2^{K-1}L$ 個先頭に並ぶ必要がありますが、その条件を満たすように操作をする方法は、上に述べた方法しかないことが分かります。よって、最初の操作でどの S を選ぶかを決めると、上の条件を満たすように操作をする方法が一意に定まるので、最終的な解の候補が定まります。高々 N 個の解の候補の中で辞書順最小のものを求めるのは $O(N^2)$ で可能であるので、全体として $O(N^2)$ の計算量でこの問題を解くことができます。

F: Counting of Subarrays

ある正整数 K に対してレベル (K, L) に属する数列を**良い数列**と呼ぶことにします。まず数列 $A_1, A_2, \ldots, A_N$ が良い数列であるかどうかを判定する方法を考えましょう。まず $N = 1$ の場合は明らかに良い数列です。$N \geq 2$ の場合は以下が必要十分であることが分かります。

- 数列が一つの値のみからなる場合 $N \geq L$ が必要十分。
- そうでない場合、最小の値 m をとるインデックスの集合を S とする。
- S が $[l_1, r_1], \ldots, [l_k, r_k] (l_i + 1 < r_{i+1})$ の和集合となるように k 個の区間を選ぶ。
- $[l_i, r_i]$ に含まれる要素数が全て L 以上であることが必要。
- そのうえで $[l_i, r_i]$ に含まれる要素を A から削除して、対応する部分に値 $m + 1$ の項を $[\frac{r_i - l_i + 1}{L}]$ 個加えた数列が良い数列であることが必要十分。

では良い部分列の個数はどのようにして数えればよいでしょうか。まず問題を少し拡張して、以下のような問題を解くことにします。

- 数列 A に加えて、数列 $L_1, \ldots, L_N, R_1, \ldots, R_N$ が与えられる。
- このとき長さ 2 以上の良い部分列 $A_i, \ldots, A_j$ に対して $L_i \times R_j$ を足し合わせた値を求める。

実はこのように問題を拡張すると、先ほどの判定法と同様にして最小値を削除し一つ大きな値を適当な個数加える、という操作を行うことでより短い数列に帰着することができます。具体的には以下のようにすればよいです。

- 数列が一つの値のみからなる場合は、よい部分列は長さ L 以上の部分列すべてなので簡単に計算できる。
- そうでない場合、先ほどと同様に区間 $[l_1, r_1], \ldots [l_k, r_k]$ をとる。
- $[l_i, r_i]$ に含まれる長さ L 以上の部分列すべてに対する $L_i \times R_j$ の値の合計値を求める。
- $[l_i, r_i]$ に含まれる要素を A から削除して、対応する部分に値 $m + 1$ の項を $[\frac{r_i - l_i + 1}{L}]$ 個加えた数列を考える。

ここで、新たに追加する $[\frac{r_i - l_i + 1}{L}]$ 個の要素について L_j, R_j の値をどのように定めればよいか考える必要があります。ここで辻褄を合わせるには、以下のように定めればよいです。

- $[l_i, r_i]$ の区間を削除して、そこに新たな項を加える場合を考える。
- 新たに加える項の中で左から j 個目の項の R_j の値を考える。
- $[l_i, r_i]$ の中で左から jX 項目から $jX + X - 1$ 項目までの R の値の合計を R_j とする。

このように定めればよいことは、良い数列かどうかの判定法から分かります。ただし、少しスコアに余分があるので適当な値を引かなければいけません。具体的には以下の値を引けば求めたい値が求まります。

- $[l_i, r_i]$ の区間を削除して新たに追加してできる区間を $[l'_i, r'_i]$ とする。
- $[l'_i, r'_i]$ の長さ L 以上の部分列に対する $L_i \times R_j$ の合計を求める。

以上により求めたい値が求まることは、判定法のアルゴリズムから従います。計算量は解析すると $O(N \log N)$ であることが分かります。これはアルゴリズムに対応する X 分木のようなものを考えると、その頂点数が $O(N)$ であることから従います。以上により、この問題を $O(N \log N)$ で解くことができました。