ABC042/ARC058 解説

三上和馬 (kyuridenamida)/森田晃平 (yosupo)

A 問題 和風いろはちゃんイージー

文字列 A, B, C の中に、長さ 5 の文字列が 2 つ、長さ 7 の文字列が 1 つあるときに限り、五七五を作ることができる。

B 問題 文字列大好きいろはちゃんイージー

全ての文字列の長さが全て L であるという条件から、与えられる文字列を辞書順で昇順に並び替え、その順で結合すれば辞書順最小を達成することができる。

一般に、文字列の長さが異なる場合でも、2 つの文字列 a, b に対し a が b より小さいことを $a + b < b + a$ と定義し、それらに基づいて文字列をソートしたものを昇順に結合すれば辞書順最小を達成することができる。証明は省略します。

C 問題 こだわり者いろはちゃん

$\{1, 2, 3, 4, 5, 6, 7, 8, 9\} \notin \{D_1, D_2, ..., D_N\}$ であることが保証されているので、0 しか使えないようなケースは無く、必ず解は存在する。

仮に使える数字の種類が 1 通りだったとしても、それらを N の桁数より 1 つ多い数だけ並べた数を考えればそれが解の上限となるため、解の上限は $10N$ である。N,$N+1$,$N+2$,... について順に解の条件を満たすか確かめ、条件を満たすと分かった時点でその数を出力するという解法で間に合う。

全体の時間計算量は $O(N \log N)$ である。

D 問題 いろはちゃんとマス目

※逆元の計算方法について誤記があったので修正しました。申し訳ありません。(7/24)

左上の座標が $(0, 0)$、右下の座標が $(H - 1, W - 1)$ であるとする。

$B \leqq i \leqq W$ を満たす全ての i について、 $(0, 0)$,$(H - A - 1, i)$,$(H - A, i)$,$(H - 1, W - 1)$ を順に通る経路の個数を数え上げ、それらを足し合わせることで、過不足なく求める経路が数えられる。図 1 はこのことを説明している。

上記の考え方に基づいて考えると、下に y 回、右に x 回移動するような経路の個数を効率的に求める必要があるが、これは $C(x + y, x)$ に等しい ($C(n, r)$ は n 個のもの中から r 個選ぶ組み合わせの数を表す)。

$C(n, r) = \frac{n!}{r!(n-r)!}$ であるから、 $0 \leqq x \leqq H + W - 2$ に対して $x!(\mathrm{mod}\ 10^9 + 7)$ と $(x!)^{-1}(\mathrm{mod}\ 10^9 + 7)$ を予め $O(H + W)$ かけて計算してテーブルにしておくことで、各

$C(n,r)$ は $O(1)$ で求めることができる。10^9+7 は素数なので、以下が言える。

- $x \equiv 0 (\mathrm{mod}\ 10^9+7)$ でない全ての x に対し $x^{-1} (\mathrm{mod}\ 10^9+7)$ が存在する。
- フェルマーの小定理より $x^{-1} \equiv x^{10^9+5} (\mathrm{mod}\ 10^9+7)$ である。

したがって、階乗数の逆元 $(x!)^{-1} \equiv (x!)^{10^9+5} (\mathrm{mod}\ 10^9+7)$ を二分累乗法等の適切な方法で計算して予め求めておけば良い。二分累乗法のオーダーは $O(1)$ ではあるが定数倍が 30 倍程あることに注意せよ。ちなみに、求めたい階乗数の最大値を $M!$ として、最初にその逆元 $(M!)^{-1}$ を二分累乗法で求めた後、その数に M をかけると $((M-1)!)^{-1}$ が求まり、その数に $M-1$ をかけると $((M-2)!)^{-1}$ が求まり、...、というふうに計算していけばより高速に逆元テーブルを構築することができるが、今回その必要はない。

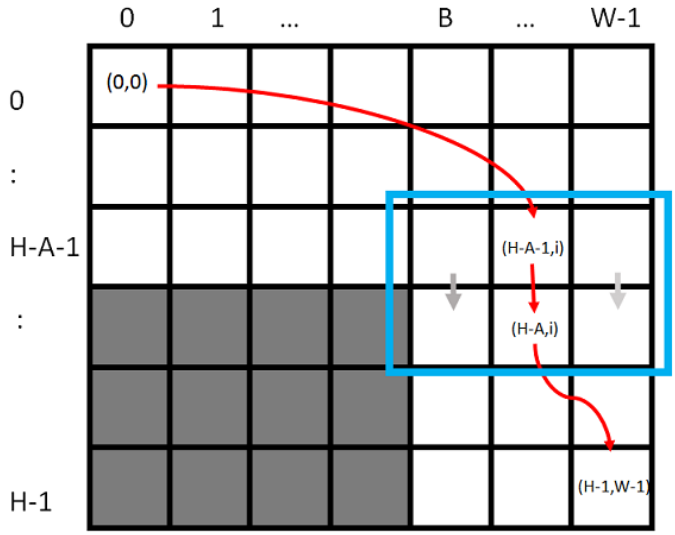

Fig1. (H,W,A,B)=(6,7,3,4)

E 問題 和風いろはちゃん

どこにも XYZ を含まない数列の個数を数え上げれば良い。もちろん XYZ を含む数列の数を数えても問題ないが、以下 XYZ を含まない数列の個数を数える方法を説明する。

左から 1 つずつ数列の値を決定していく全探索をメモ化することを考える。つまり、数列の今見ているところの値を XYZ が生じないように定め、今見ているところを 1 個右に移し、また値を決めて…というのを繰り返す。

これはどのようにメモ化出来るだろうか？

まず、XYZ=575 の時の、$1,1,...,1$(1 が 17 個) の場合が、数列の値を最も使うパターンである。つまり、直前 16 個の値を保存すれば良い。

これで状態数は $N * 10^{16}$ になる。全探索に比べたらかなりの改善だが、もちろん間に合わない。

もう少し考える。例えば直前に現れた値が 5,5,5,5,7 などであればこれを全部保存する必要はない。つまり、直前に現れた値を合計が 16 以下でとなる範囲でだけ保存すれば良いことがわかる。

これにより状態数は減るが、どのぐらい減るのだろうか？

ここで、1=”1”, 2=”10”, 3=”100”, 4=”1000”... と 0/1 文字列に変換して考えてみる。つまり、直前に現れた値が 1,2,3 だとすると”110100”と変換する。この文字列の長さは値の sum と等しい。つまり、前の値を合計が 16 以下まで保存するというのは、長さ 16 の 0/1 文字列に対応することがわかる。よって状態数は $N * 2^{16}$ 。もちろん実際には文字列で保存するのではなく、16 bit の整数として保存し bit DP を行う。

これにより計算量は $O(A * N * 2^{X+Y+Z})$ となる (A は数列の値の種類、つまり 10)。これで間に合う。

F 問題 文字列大好きいろはちゃん

dp[i][j] := i 番目以内の文字列を使い、ちょうど長さを j 文字にするときの辞書順最小の文字列

と定義する。ただし $i+1$ 文字目以降の文字列を使って埋めても $K-j$ 文字にできないときは定義されないこととする。$K-j$ 文字にできるかどうかの判定をするための DP テーブルを構築するのは容易である。

文字列をそのまま持つと空間計算量は $O(NK^2)$ だが、最適な文字列のみをうまく保持することを考えると、dp[i][0], dp[i][1], ..., dp[i][K] はそれぞれ「定義されない」or「ある共通の文字列の prefix」のどちらかになる。つまり、各 i に対してある共通文字列 $\mathrm{CS_i}$ (長さは高々 K) と、dp[i][j] が定義されているかどうかの判定フラグを持つことで、空間計算量が $O(NK)$ に抑えられる。

各 i に対する共通文字列について、それ自体が辞書順最小となるような dp[i][j] と判定フラグを保持することだけを考えれば良い。

dp テーブルの更新時に、dp テーブルの文字列や s_i、またはそれらを結合したもの同士の文字列の辞書順比較を高速に行う必要が出てくるが、LCP (Longest Common Prefix) を求めればその処理が高速に行えることが分かる。扱う文字列を全てローリングハッシュテーブルで管理し部分文字列のハッシュ値を $O(1)$ で求められるようにしておき、それら対して二分探索を行うことにより、 時間計算量 $O(logk)$ で LCP を求めることができる。

全体の時間計算量は $O(NKlogK)$、空間計算量は $O(NK)$ となる。128 以下の 2 べきの数を法としたローリングハッシュでは通らないようにテストケースを工夫した。したがって除算を行う必要があるが、除算は重くその定数倍のせいで、ローリングハッシュを用いた解法を通すのは難しい。

そこで、比較部分のアルゴリズムをさらに改善することを考える。

どのような文字列の比較を行うか考える。すると

- $\mathrm{CS_i}$ のうち先頭何文字か
- $\mathrm{CS_i}$ のうち先頭何文字か + s_i

という文字列同士のみを比較することがわかる。ここで、$s_i + \mathrm{CS_i}$ について $Z-Algorithm$ を適用したテーブルを用意しておくと、これらの文字列同士の比較は O(1) で行うことが出来る。

よって時間計算量は $O(NK)$ となる。これが想定解である。

以下、上記の説明に基づいた dp テーブル更新の一例を示す。定義されている値のみ列挙している。比較パートについては触れていない。

$N = 5, K = 7$ のケースで、各 s_i は以下の通りとなっている。

- s_1=”aaaaaa”
- s_2=”abc”
- s_3=”xxxx”
- s_4=”ddd”
- s_5=”abcd”

$i = 0, 1, 2, 3, 4, 5$ における DP テーブルの更新は次のようになっている。

1. $i = 0$: dp[0][0]=””
2. $i = 1$: dp[1][0]=””
    - s_1 を使うと以降どう使っても長さ $K = 7$ の文字列を達成できないことから、dp[1][6] は定義されない。
3. i=2: dp[2][0]=””, dp[2][3]=”abc”
    - 共通文字列 $\mathrm{CS_2}$ =”abc”
4. $i = 3$: dp[3][0]=””, dp[3][3]=”abc”, dp[3][7]=”abcxxxx”
    - 共通文字列 $\mathrm{CS_3}$ =”abcxxxx”
    - dp[1][4]=”xxxx”としても、辞書順最小の共通文字列にはなりえないので dp[1][4] は定義されない。
5. $i = 4$: dp[4][0]=””, dp[4][3]=”abc”, dp[4][7]=”abcxxxx”
6. $i = 5$: dp[5][0]=””, dp[5][3]=”abc”, dp[5][7]=”abcabcd”
    - 共通文字列 $\mathrm{CS_5}$ =”abcabcd”