AGC 021 解説

DEGwer

2018/02/24

For International Readers: English editorial starts on page 6.

A: Digit Sum 2

与えられた数 N が 10 進法で K 桁からなるとします。最上位の桁が c であるとすれば、N 以下の正の整数は、

- 下から K 桁目は c 以下
- それ以外の位は 9 以下

です。よって、答えは $c+9(K-1)$ 以下となります。

また、答えが $c+9(K-1)$ のとき、条件を満たす整数は $c999\cdots999$ のみです。この値が N 以下となるような N は $c999\cdots999$ の形をしているものしかありえないため、この場合のみ答えが $c+9(K-1)$ となります。

そうでない場合、整数 $(c-1)999\cdots999$ は N 以下で、各桁の和は $c+9(K-1)-1$ となります。よってこれが最適で、すべての場合について答えが求まりました。

時間計算量は $O(\log N)$ です。

B: Holes

R を大きくしていくことを考えましょう。与えられた点集合の凸包上にない穴について、りんごさんの選ぶ点であって、その穴にすぬけ君が落ちるような領域の面積は定数で抑えられます。よって、R を大きく取れば、求める確率は 0 に収束します。また、凸包の辺上にある場合も、そのような領域の面積は $O(R)$ なので、R を大きく取れば求める確率は 0 に収束します。

そうでない場合、穴 p について確率を求めることを考えましょう。与えられた点集合の凸包上で p に隣り合う穴をそれぞれ q,r とします (q,r は同一の穴であることもあります)。りんごさんの選ぶ点であって、穴 p にすぬけ君が落ちるような領域の面積は、pq の垂直二等分線と qr の垂直二等分線によって 4 つに分けられた領域のうちの p を含む部分 (以後これを T と呼ぶ) から、p 付近の高々有限の面積の領域を除いたものです。この高々有限の面積の部分は極限操作によって無視されることになるので、T と R の共通部分の面積について考えればよいことがわかります。

さて、2 本の垂直二等分線の交点を原点に持っていったとき、T の面積の変化分は $O(R)$ です。よって極限操作によってそのずれは無視され、結局、2 本の垂直二等分線の角度が 2π に対して占める割合が、求める確

率となります。

以上は、$O(N \log N)$ 時間で動くように実装できます。

C: Tiling

面積の条件から、$2(A+B) \leq NM$ が必要です。

さて、以下の 4 通りに分けて考えることにしましょう。

N, M がともに偶数の場合

奇数列目を黒、偶数列目を白に塗ると、縦向きのタイルは同じ色のマス 2 つを、横向きのタイルは異なる色のマス 2 つを覆います。各色のマスは偶数個ずつあるので、もし $2(A+B) = NM$ なら、横向きのタイルは偶数枚ある必要があります。縦向きのタイルについても同様です。

逆に、その条件が満たされる場合、以下のようにして条件を満たすタイルの置き方が構成できます。

- 全体を 2×2 の領域に区切る。各領域を、縦向きのタイル 2 枚または横向きのタイル 2 枚で適切に埋める。
- 余分なタイルを取り除く。

N が偶数、M が奇数の場合

先ほどと同様、奇数列目を黒、偶数列目を白に塗ることを考えると、白のマスは $(M-1)N/2$ 個しかないので、横向きのタイルは $(M-1)N/2$ 枚しか置くことはできません。また、もし $2(A+B) = NM$ なら、先ほどと同様、横向きのタイルは偶数枚ある必要があります。

逆に、その条件が満たされる場合、以下のようにして条件を満たすタイルの置き方が構成できます。

- 右端の列を、縦向きのタイルで埋める。
- 残った部分を 2×2 の領域に区切る。各領域を、縦向きのタイル 2 枚または横向きのタイル 2 枚で適切に埋める。
- 余分なタイルを取り除く。

N が奇数、M が偶数の場合

先ほどの場合と対称です。

N, M がともに奇数の場合

先ほどと同様、横向きのタイルは $(M-1)N/2$ 枚しか置くことはできません。同様に、縦向きのタイルは $(N-1)M/2$ しか置くことはできません。

逆に、その条件が満たされる場合、以下のようにして条件を満たすタイルの置き方が構成できます。

$NM-1=2(A+B)$ でないか、縦向きのタイルの枚数と、置ける縦向きのタイルの最大枚数の偶奇が等しい場合

- 右端の列の下端のマス以外を、縦向きのタイルで埋める。
- 下端の列の右端のマス以外を、横向きのタイルで埋める。
- 残った部分を 2×2 の領域に区切る。各領域を、縦向きのタイル 2 枚または横向きのタイル 2 枚で適切に埋める。
- 余分なタイルを取り除く。

そうでない場合

- 右下の 3×3 領域に、縦向きのタイル 2 枚と横向きのタイル 2 枚を四畳半敷きの塩梅で置く。
- 右端の列の残った部分を、縦向きのタイルで埋める。
- 下端の列の残った部分を、横向きのタイルで埋める。
- 残った部分を 2×2 の領域に区切る。各領域を、縦向きのタイル 2 枚または横向きのタイル 2 枚で適切に埋める。
- 余分なタイルを取り除く。

正当性は、N か M が 1 の場合は必ず前者の場合になることと、前者と後者で最後から 2 番目のステップで置ける横向きのタイルの枚数の偶奇が異なることがわかるので、証明できます。

以上で全ての場合について解けたので、この問題を解くことができました。

D: Reversed LCS

T と T を反転した文字列 T' の最長共通部分列はどのようなものかを、まず考えることにしましょう。

$T_{i_1},T_{i_2},\cdots,T_{i_k}$ と $T_{j_1},T_{j_2},\cdots,T_{j_k}$ (ただし、$i_1<\cdots<i_k, j_1>\cdots>j_k$ が成り立つ) が列として等しいとします。このとき、ある整数 p が存在し、$i_p<j_p$ かつ $i_{p+1}\geq j_{p+1}$ となります (ただし、便宜上 $i_{k+1}=|T|+1, j_{k+1}=-1$ とします)。

さて、$i_{p+1}>j_{p+1}$ のとき、列 $T_{i_1},\cdots,T_{i_p},T_{j_p},\cdots,T_{j_1}$ と列 $T_{j_k},\cdots,T_{j_{p+1}},T_{i_{p+1}},\cdots,T_{i_k}$ はともに回文で、長さの合計は $2k$ です。すなわち、T の部分列であって、回文であり、長さが k 以上のものが存在します。

$i_{p+1}=j_{p+1}$ のときも同様に、列 $T_{i_1},\cdots,T_{i_p},T_{i_{p+1}},T_{j_p},\cdots,T_{j_1}$ と列 $T_{j_k},\cdots,T_{j_{p+2}},T_{i_{p+1}},T_{i_{p+2}},\cdots,T_{i_k}$ はともに回文で、長さの合計は $2k$ です。すなわち、T の部分列であって、回文であり、長さが k 以上のものが存在します。

回文は T と T' の共通部分列となるので、最長共通部分列としては回文であるもののみ考えればよいです。よって、$DP[l][r][x]$ を、区間 $[l,r]$ の文字を x 個まで変更したときの、その区間の部分列であって回文であるものの最大長として DP を行うことで、この問題を $O(N^3)$ 時間で解くことができます。

E: Ball Eat Chameleons

投げ入れる赤のボールの個数で場合分けをします。赤のボールを R 個、青のボールを B 個投げ入れるときの答えを高速に求められれば、$R+B=K$ なる非負整数の組 (R,B) すべてに対してそれらを足し合わせる

ことで、求める答えを得ます。

$R < B$ のとき

いずれかのカメレオンは、食べた青のボールの個数が食べた赤のボールの個数より多くなります。そのカメレオンは最終的に青色となるので、求める場合の数は 0 です。

$R = B$ のとき

全てのカメレオンが、赤と青のボールを等しい個数ずつ食べます。赤と青のボールを等しい個数ずつ食べたカメレオンが最終的に赤になっているための必要十分条件は、ボールを 1 つ以上食べ、かつ最後に食べたボールが青であることです。

全てのカメレオンの最後に食べたボールが青であることから、最後にりんごさんが投げ入れるボールは青である必要があります。また、すべてのカメレオンは最後に赤いボールを食べた後に青いボールを一度以上食べるので、りんごさんが投げ入れたボールを順に並べた列から、(赤, 青) とこの順に並ぶ (連続するとは限らない) 組が disjoint に N 組取れる必要があります。

逆に、これらの条件が満たされれば、(最後に青いボールを食べるカメレオンに、組にならないボールをすべて食べさせることで) 全てのカメレオンを最終的に赤色にすることができます。

格子上を x 座標または y 座標の増加する方向に $R+B$ 回進んで (R, B) に至る経路をりんごさんの投げ入れたボールを順に並べた列に自然に対応付けることにすれば、この条件は、

- $(R, B-1)$ を通る
- $y - x \geq B - N + 1$ なる領域を通らない

と言い換えられます。これは、後述の方法で求められます。

$R > B$ のとき

上の場合とだいたい同じですが、赤いボールのほうを多く食べるカメレオンが存在するため、最後に投げ入れるボールが青である必要はありません (赤いボールのほうを多く食べるカメレオンに、余計なボールをすべて食べさせれば良いです)。

赤と青のボールを同数食べるカメレオンの数は最小で $N - (R - B)$ なので、上の場合と同様の言い換えをすれば、条件は

- $y - x \geq R - N + 1$ なる領域を通らない

と書くことができます。これは、後述の方法で求められます。

さて、あとは $y - x \geq T$ なる領域を通らずに $(0, 0)$ から (X, Y) まで格子上を最短距離で進む経路の個数を高速に求められれば良いです。

$(0, 0)$ から (X, Y) まで行く経路のうち、$y - x \geq T$ なる領域を通る経路について、初めて $y - x = T$ となった地点から先の部分を折り返せば、$(0, 0)$ から $(Y - T, T + X)$ まで行く経路となります。逆に、$(0, 0)$ から $(Y - T, T + X)$ まで行く経路を同様に折り返せば、$(0, 0)$ から (X, Y) まで行く経路のうち、$y - x \geq T$ なる領域を通る経路 になります。よって、両者の個数は等しいので、求める場合の数は、$(0, 0)$ から (X, Y)

まで行く経路の個数 $\binom{X+Y}{X}$ から、$(0,0)$ から $(Y-T,T+X)$ まで行く経路の個数 $\binom{X+Y}{Y-T}$ を引いた値となります。

これは適切な前処理をしておけば定数時間で求められるので、$O(K)$ 時間でこの問題を解くことができました。

F: Trinity

$DP[k][p]$ を、p 行 k 列のマス目のいくつかを黒く塗ったときの列の組 (A,B,C) であって、$A_i = k+1$ なる i が存在しないようなものの個数とします。すなわち、すべての行に黒く塗られたマスが存在する必要があります。求める答えは $DP[M][p]$ たちを用いて簡単に計算できるため、以下、この DP を計算することを考えます。

$DP[k][p]$ から $DP[k+1][p+q]$ への遷移につく係数は何でしょうか？ 以下の 2 通りに分けて考えてみましょう。

$q=0$ の場合、黒く塗られたマスの存在する行が新たに出現することはありません。すなわち、k 列目までには黒く塗られたマスがなく、$k+1$ 列目に初めて存在するような行はありません。

このとき、A_i の値が新たに定義されることはありません。B_{k+1}, C_{k+1} の値が新たに定義されますが、これは $k+1$ 行目に黒く塗られたマスが存在しない場合 1 通り、存在する場合 $\binom{p+1}{2}$ 通りです。よって、遷移の係数は $\binom{p+1}{2}+1$ です。

$q>0$ の場合、p 個の行の間に新たに q 個の行が挿入され、$A_l = k+1$ なる l が新たに q 個出現します。このような l の位置から、新たに出現した行の位置は一意的に復元できます。さて、B_{k+1} がこの $p+q$ 個の行のうち上から $i+1$ 番目の行の番号に等しくなるためには、上から i 個の行は元からあった行でなければなりません。同様に、C_{k+1} がこの $p+q$ 個の行のうち下から $j+1$ 番目の行の番号に等しくなるためには、下から j 個の行はもとからあった行でなければなりません。それ以外は任意に決められるため、遷移につく係数は、

$$\sum_{i+j\le p} \binom{p-i-j+q}{q} = \sum_{i=0}^{p} (i+1)\binom{p+q-i}{q} \tag{1}$$

$$= \sum_{i=0}^{p} \binom{p+q+1-i}{q+1} \tag{2}$$

$$= \binom{p+q+2}{q+2} \tag{3}$$

となり、$O(N^2M)$ 時間でこの DP が計算できます。

さて、後者の遷移につく係数をよく見てみましょう。$\binom{p+q+2}{q+2} = \frac{(p+q+2)!}{p!(q+2)!}$ より、遷移の係数は p の式、q の式、$p+q$ の式の積に分解できます。よって、$X_p = \frac{DP[k][p]}{p!}, Y_q = \frac{1}{(q+2)!}$ として FFT などで畳み込みを行い、係数 $(p+q+2)!$ を後からかけることで、遷移を $O(N\log N)$ 時間で計算できます。前者の遷移は $O(N)$ 時間で計算できるので、$O(NM\log N)$ 時間でこの問題を解くことができました。

AGC 021 Editorial

DEGwer

2018/02/24

A: Digit Sum 2

Suppose that N has K digits, and the leftmost (the most significant) digit is c. Consider the following two cases:

- N is of the form $c999\cdots999$. In this case, for all $x \leq N$, the K-th (from the smallest) digit of x is at most c, and each of the other digits is at most 9, thus the sum is at most $c + 9(K-1)$. By choosing $x = N$, we can achieve the sum $c + 9(K-1)$. Therefore, the answer is $c + 9(K-1)$.
- Otherwise, the answer must be less than $c + 9(K-1)$. On the other hand, the integer $(c-1)999\cdots999$ is less than N (because the leftmost digit is smaller), and achieves the sum $c+9(K-1)-1$. Therefore, the answer is $c + 9(K-1) - 1$.

This solution works in $O(\log N)$ time.

B: Holes

Since R is extremely large, we can assume that a randomly chosen point is sufficiently distant from the holes.

Let's call the chosen point s. When will it fall into a hole p? For each hole q (other than p), $dist(s,p) < dist(s,q)$ must hold. Since s is sufficiently distant from the holes, this condition holds if and only if the angle $\angle spq > \pi/2$.

This gives an very simple $O(N^2 \log N)$ solution. Fix a hole p. Compute the directions from p to each of the other holes, and sort those directions. The probability that p is chosen is $\max\{(\text{the longest gap between two adjacent directions} - \pi), 0\}/(2\pi)$.

Note that we can improve it to $O(N \log N)$. Let's take the convex hull of all holes. If a hole p is not a vertex of the convex hull, the probability that p is chosen is 0. Otherwise, let q, r be the two vertices of the convex hull adjacent to p. Then, he probability that p is chosen is $(\pi - \angle qpr)/(2\pi)$.

C: Tiling

The following three conditions are necessary:

- $A + B \leq \lfloor \frac{NM}{2} \rfloor$ (compare areas)
- $A \leq N\lfloor \frac{M}{2} \rfloor$ (we can put $\lfloor \frac{M}{2} \rfloor$ tiles per row)
- $B \leq M\lfloor \frac{N}{2} \rfloor$ (we can put $\lfloor \frac{N}{2} \rfloor$ tiles per column)

Let's define $S := \lfloor \frac{NM}{2} \rfloor, S_A := N\lfloor \frac{M}{2} \rfloor, S_B := M\lfloor \frac{N}{2} \rfloor$. Then, $A + B \leq S, A \leq S_A, B \leq S_B$ must hold.
On the other hand, we can come up with the following way to put tiles:

- First, put 2×2 squares from the top-left corner (in the most natural way). We will get $\lfloor \frac{N}{2} \rfloor \lfloor \frac{M}{2} \rfloor$ squares.
- We divide each square vertically or horizontally: for each square, we will get two tiles of the same type.
- Additionally, we can put $\lfloor \frac{M}{2} \rfloor$ 1×2 tiles in case N is odd, and $\lfloor \frac{N}{2} \rfloor$ 2×1 tiles in case M is odd.

By this method, we can achieve $(A, B) = (S_A, S-S_A), (S_A-2, S-S_A+2), (S_A-4, S-S_A+4), \ldots, (S-S_B, S_B)$. (And obviously, if $A' \leq A$ and $B' \leq B$, we can achieve (A', B') if we can achieve (A, B).)

Therefore, the only non-trivial cases are $(A, B) = (S_A - 1, S - S_A + 1), (S_A - 3, S - S_A + 3), \ldots, (S - S_B + 1, S_B - 1)$.

It turns out that the answer for these cases depend on the parity of NM:

In case NM is odd

Note that the non-trivial cases don't exist when $N = 1$ or $M = 1$. Thus, when N and M are odd, we can assume that $N, M \geq 3$.

Take a 3×3 square from the bottom-right corner, and put two tiles of each type. In the previous method, this part was filled by an odd number of tiles of each type. Thus, we can achieve the different parity this way. By filling the remaining part in a similar way, we can achieve $(A, B) = (S_A - 1, S - S_A + 1), (S_A - 3, S - S_A + 3), \ldots, (S - S_B + 1, S_B - 1)$.

In case NM is even

Paint the entire grid black and white: a cell in odd-numbered rows are black and other cells are white. Then, each 1×2 tiles contains even number of black cells, and each 2×1 tiles contains odd number of black cells. Thus, the parity of total number of 2×1 tiles must match the parity of total number of black cells.

Therefore, in this case, it turns out that the answer for non-trivial cases are "NO".

D: Reversed LCS

First, let's find a simple way to compute the length of LCS of T and T'.

Suppose that the length of LCS is k. Then, there exist indices $i_1 < \cdots < i_k$ and $j_1 > \cdots > j_k$ such that the sequences $T_{i_1}, T_{i_2}, \cdots, T_{i_k}$ and $T_{j_1}, T_{j_2}, \cdots, T_{j_k}$ are the same.

Then, there exists an integer p such that $i_p < j_p$ and $i_{p+1} \geq j_{p+1}$. (For convenience, let $i_{k+1} = |T| + 1$ and $j_{k+1} = -1$.)

If $i_{p+1} > j_{p+1}$, both the subsequence $T_{i_1}, \cdots, T_{i_p}, T_{j_p}, \cdots, T_{j_1}$ and the subsequence $T_{j_k}, \cdots, T_{j_{p+1}}, T_{i_{p+1}}, \cdots, T_{i_k}$ are palindromes, and the sum of their lengths is $2k$. Thus, T contains a sub-palindrome whose length is at least k.

Similarly, if $i_{p+1} = j_{p+1}$, both the subsequence $T_{i_1}, \cdots, T_{i_p}, T_{i_{p+1}}, T_{j_p}, \cdots, T_{j_1}$ and the subsequence $T_{j_k}, \cdots, T_{j_{p+2}}, T_{i_{p+1}}, T_{i_{p+2}}, \cdots, T_{i_k}$ are palindromes, and the sum of their lengths is $2k$. Thus, T contains a sub-palindrome whose length is at least k.

Therefore, we proved that the length of LCS of T and T' is actually the length of the longest (not necessarily continuous) sub-palindrome of T.

Now, we are interested in the longest possible sub-palindrome we can make by changing at most K characters in T. Define $DP[l][r][x]$ as the length of the longest possible sub-palindrome we can make by changing at most x characters in $T[l..r)$ (a substring of T). This DP works in $O(N^3)$ time.

E: Ball Eat Chameleons

A chameleon becomes red if one of the following holds:

- It ate more red balls than blue balls.
- It ate the same (nonzero) number of red balls and blue balls, and the last ball it ate is blue.

Suppose that there are R red balls and B blue balls in total. For each pair (R, B) such that $R+B = K$, we will compute the number of valid sequences of R red balls and B blue balls. A sequence of R red balls and B blue balls is valid if we can divide it N disjoint (possibly non-continuous) subsequences such that for each subsequence, one of the two conditions above hold. Let's compute the number of such sequences.

In case $R < B$

Obviously, the answer is 0.

In case $R = B$

In this case, all subsequences must contain the same number of reds and blues, and end with a blue. Thus, the following conditions are necessary:

- The last element of the sequence must be a blue.
- It must be possible to choose N disjoint subsequences "red, blue".

It turns out that these conditions are sufficient: we assign each of the N pairs to different chameleons, and assign everything else to the chameleon with the last blue ball.

Now, let's plot a sequence on an xy-grid. A sequence corresponds to a path from $(0, 0)$ to (R, B). We see the sequence from left to right, and when we see an element red (blue), we extend the path to the right (up).

Then, the conditions above correspond to the following:

- It is a path from $(0, 0)$ to (R, B) that goes only up and right, and passes through $(R, B - 1)$.
- It never enters the region with $y - x \geq B - N + 1$.

The number of such paths can be computed in $O(1)$, and it will be described later.

In case $R > B$

It's very similar to the case above, but this time we are not interested in the last ball. A sequence is good iff:

- It must be possible to choose $\max N - (R - B), 0$ disjoint subsequences "red, blue".

(note that this time an arbitrary chameleon with (red) > (blue) receives "everything else")
and if we describe it in the word of paths,

- It is a path from $(0, 0)$ to (R, B) that goes only up and right.
- It never enters the region with $y - x \geq R - N + 1$.

Therefore, we can compute the answer for each (R, B) in $O(1)$, and the solution works in $O(K)$ time.
The only remaining thing is the computation of the number of paths with the following conditions:

- It is a path from $(0, 0)$ to (X, Y) that goes only up and right.
- It never enters the region with $y - x \geq T$.

This is a well-known problem (for computation of Catalan numbers).

Since we can easily compute the number of paths from $(0, 0)$ to (X, Y), we are interested in the number of such paths that enters the region $y - x \geq T$ at least once (let's call it a bad path).

Let's fix a bad path P. Suppose that P enters the region for the first time at the point (p, q). Define a path Q as follows:

- From $(0, 0)$ to (p, q), Q follows P.
- After that, Q is a "reflection" of P: when P goes right, Q goes up, and when P goes up, Q goes right.

Then, Q is a path from $(0, 0)$ to $(Y - T, T + X)$. (Notice that $p - q = T$).
This gives a bijection between a bad path and a path from $(0, 0)$ to $(Y - T, T + X)$.
Therefore, the answer for this is simply $\binom{X+Y}{X}$ minus $\binom{X+Y}{Y-T}$.

F: Trinity

Let $DP[k][p]$ be the answer of the problem for the grid with p rows and $k$columns, with an additional condition that in each row we must paint at least one cell black. Since the answer is $\sum DP[M][p]\binom{N}{p}$, from now on, we will focus on the computation of this DP table.

Fix a grid with k rows and p columns (this time, some rows may be empty). We have three arrays A, B, C that corresponds to this grid. What happens to the three arrays if we attach the $(p+1)$-th column to the right?

- If the i-th row is empty before the attachment, A_i will be $p+1$. Otherwise, the value of A_i doesn't change.
- The first p elements of B, C don't change.
- We get two values B_{p+1}, C_{p+1}: these two values depend on the newly added column.

Let's call a grid nice if each row is non-empty. We can make a nice grid with $p+q$ rows and $k+1$ columns from a grid with p rows and k columns as follows:

- Initially we have a nice grid with p rows and k columns.
- Insert q new empty rows into some positions. Now we have a grid with $p+q$ rows and k columns.
- Attach a new (not necessarily empty) column to the right. Cells that correspond to empty rows must be black. Now we have a nice grid with $p+q$ rows and $k+1$ columns.

We update the DP table as in the process above. We perform "$DP[k+1][p+q] += DP[k][p] * (coefficient)$;" for each pair (k, p, q) in the increasing order of k. The coefficient is the number of ways to get a new three arrays A, B, C in the process above: it turns out that the initial content of the three values doesn't affect this coefficient.

How to compute this coefficient?

If $q = 0$, we only care the last values of two arrays B, C. These are the positions of the topmost/bottommost black cell in the newly added column. There is one way for an empty column, and there are $\binom{p+1}{2}$ ways for a non-empty column. Thus, the coefficient is $\binom{p+1}{2} + 1$.

What happens if $q > 0$? We insert q new rows into p old rows. We can see this as a sequence of p gray balls and q black balls (the black balls corresponds to newly inserted rows). The positions of the black balls corresponds to the positions of l such that $A_l = k+1$.

Furthurmore, we should think about the last elements of B, C. In the word of balls, these correspond to the leftmost/rightmost positions of black balls. However, the problem is that, we don't know the information about gray balls: it can be either black or not black. Therefore, the combinatorial interpretation of the coefficient is as follows:

- Imagine a sequence of p gray balls and q black balls.
- Consider all gray balls that are to the right of the rightmost black ball: you can choose at most one of them and paint it red.

- Consider all gray balls that are to the left of the leftmost black ball: you can choose at most one of them and paint it red.

The annoying part is "at most". Instead, we attach an extra gray ball to each end, and now it becomes as follows:

- Imagine a sequence of $p+2$ gray balls and q black balls. Both ends are gray.
- Consider all gray balls that are to the right of the rightmost black ball: you can choose exactly one of them and paint it red.
- Consider all gray balls that are to the left of the leftmost black ball: you can choose exactly one of them and paint it red.

Now, we can regard red balls as black, and it's just a sequence of p gray balls and $q+2$ black balls. Thus, the coefficient turns out to be $\binom{p+q+2}{q+2}$.

Now we get the coefficient. This solution works in $O(N^2M)$ time, and is fast enough to get partial scores.

How can we improve this? We have two arrays x, y. Initially x is given and y is empty. For each (p, q) such that $q > 0$, we perform the following:

$y_{p+q} += x_p \times \binom{p+q+2}{q+2}$

(We can ignore the case $q = 0$ because it can be done in $O(N)$ obviously).

We want to perform this operation quickly. Let's take a close look at the coefficient:

$\binom{p+q+2}{q+2} = \frac{(p+q+2)!}{p!}$

Thus, by defining $X_p = \frac{x_p}{p!}$ and $Y_q = \frac{y_q}{(q+2)!}$, it changes to the following:

$y_{p+q} += X_p \times \frac{1}{(q+2)!}$

This is just a standard convolution and can be done in $O(N \log N)$. Now the solution works in $O(NM \log N)$.