Code festival 予選 A 解説

sugim48 and DEGwer

2017/09/15

For International Readers: English editorial starts on page 9.

A: Snuke's favorite YAKINIKU

与えられる文字列を読み込み、最初の 4 文字を YAKI と比較すればよいです。

C/C++ では、以下のような実装で正答を得ることができます。読み込んだ文字列の末尾にヌル文字が挿入されるので、S の長さが 4 に満たないときでも、配列の初期化の必要はありません。

```
#include<stdio.h>
int main()
{
        char s[100];
        scanf("%s", s);
        if (s[0] == 'Y'&&s[1] == 'A'&&s[2] == 'K'&&s[3] == 'I') printf("Yes\n");
        else printf("No\n");
}
```

B: fLIP

ボタンを押す順序は関係なく、また、各行/列に対して 2 回以上ボタンを押す必要はありません。

さて、行に付属するボタンのうちの k 個、列に付属するボタンのうちの l 個を押したとします。このとき、黒マスの個数は $k(M-l)+(N-k)l$ となることがわかります。

よって、k, l の組をすべて試し、$k(M-l)+(N-k)l = K$ となるかどうかを調べればよいです。時間計算量は $O(NM)$ です。

また、この問題は $O(N)$ 時間でも解くことができます。興味のある人は、考えてみてください。

C : Palindromic Matrix

H 行 W 列の行列 $A = (a_{i,j})$ に対して，次の 2 つの条件は等価です．

- A の各行および各列は回文である．
- 各 i $(1 \leq i \leq H)$, j $(1 \leq j \leq W)$ に対して，$a_{i,j} = a_{i,W+1-j} = a_{H+1-i,j} = a_{H+1-i,W+1-j}$ である．

例えば，H および W が奇数である行列が後者の条件を満たすとき，同一の値をとるべき要素を同じ文字で表してグループ分けすると，次のようになります．

$$\begin{pmatrix} a & b & c & b & a \\ d & e & f & e & d \\ a & b & c & b & a \end{pmatrix}$$

このように，行列の要素はサイズ 1, 2, 4 のグループへ分割されることになります．サイズ 1, 2, 4 のグループの個数を，それぞれ g_1, g_2, g_4 とします．H および W が奇数である場合，g_1, g_2, g_4 は次のように計算できます．

- $g_1 = 1$
- $g_2 = \lfloor \frac{H}{2} \rfloor + \lfloor \frac{W}{2} \rfloor$
- $g_4 = \lfloor \frac{H}{2} \rfloor \times \lfloor \frac{W}{2} \rfloor$

H または W が偶数である場合も，同様にして計算できます．

H 行 W 列の行列 A に対して，既に g_1, g_2, g_4 が求まっているとします．このとき，次の 2 つの条件は等価です．

- A の要素を並べ替え，A の各行および各列が回文であるようにできる．
- A の要素を，サイズ 1 のグループ g_1 個，サイズ 2 のグループ g_2 個，サイズ 4 のグループ g_4 個へ分割し，各グループが同一の値の要素のみからなるようにできる．

A が後者の条件を満たすか否かは，次のようにして判定できます．前処理として，各文字 $c = \mathrm{a}, \mathrm{b}, ..., \mathrm{z}$ に対して，c が A に何個含まれるかを数え，その個数を $count_c$ とします．最初に，次の処理を g_1 回行います．

- $count_c \equiv 1, 3 \mod 4$ である c を (存在すれば) 適当にひとつ選び，$count_c$ の値を 1 だけ減らす．

次に，次の処理を g_2 回行います．

- $count_c \equiv 2 \mod 4$ である c を (存在すれば) 適当にひとつ選び，$count_c$ の値を 2 だけ減らす．

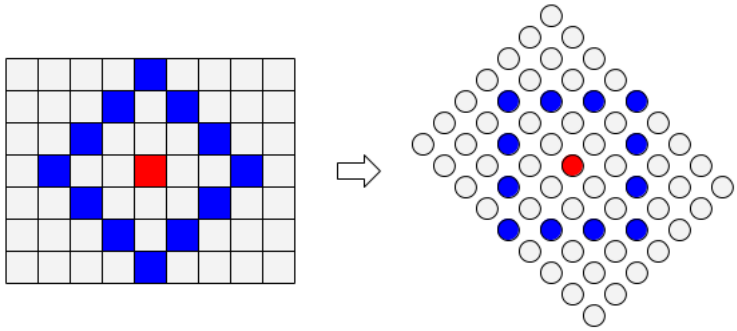
最後に，次の処理を g_4 回行います．

- $count_c \equiv 0 \mod 4$ である c を (存在すれば) 適当にひとつ選び，$count_c$ の値を 4 だけ減らす．

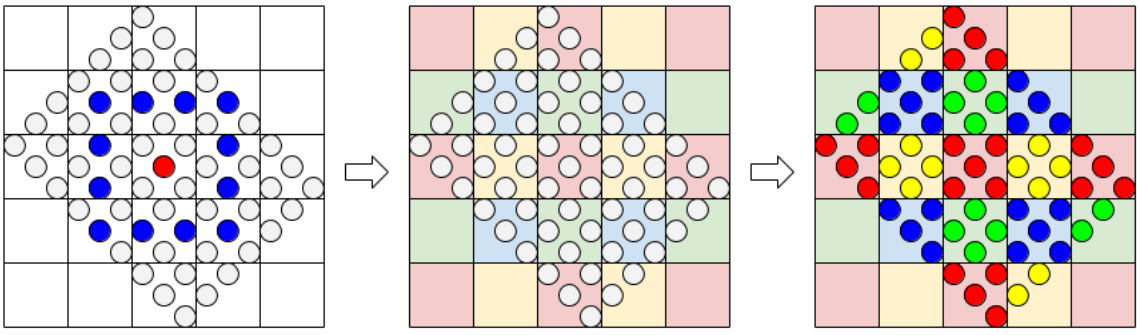
最終的に，すべての文字 c に対して $count_c = 0$ になっているならば，答えは `Yes` で，そうでないならば，答えは `No` です．

D : Four Coloring

この問題では，マス (i_1, j_1) と (i_2, j_2) の間の距離を $|i_1 - i_2| + |j_1 - j_2|$ と定義しています．このような距離はマンハッタン距離と呼ばれます．平面上のマンハッタン距離を扱う問題では，座標系を $45°$ だけ回転すると，見通しが良くなることが多いです．そこで，この問題でも座標系を $45°$ だけ回転することにしましょう．具体的には，各マス (i, j) $(1 \leq i \leq H, 1 \leq j \leq W)$ に対して，平面上の点 $(i + j, i - j)$ を対応させます．ここで，点 (x_1, y_1) と (x_2, y_2) の間のチェビシェフ距離を $\max\{|x_1 - x_2|, |y_1 - y_2|\}$ と定義します．すると，マンハッタン距離がちょうど d であるようなマスのペアは，チェビシェフ距離がちょうど d であるような点のペアに対応します．次図は $d = 3$ の場合の例です．



以降は，「チェビシェフ距離がちょうど d であるような点のペアには，異なる色が塗られている」という条件が成り立つように，各点を 4 色で塗る方法を考えます．まず，$d \times d$ の正方形のタイルで平面を分割します (次図左)．すると，ある点 P からのチェビシェフ距離がちょうど d であるような点は，P を含むタイルの 8 近傍にあるタイルのいずれかに含まれることが分かります．よって，8 近傍のタイルが異なる色になるように各タイルを 4 色で塗り，さらに各タイル内の点を同じ色で塗れば，条件が成り立ちます．実際に，次図中央のようなルールで各タイルを塗ると，8 近傍のタイルが異なる色になります．あとは，各タイル内の点を同じ色で塗ればよいです (次図右)．



E: Modern Painting

ます、人がいない場合、答えは 1 です。

最初に動く人が縦方向に動くか横方向に動くかで場合分けをします。対称性より、縦方向に動く場合のみ考えることにします。答えは、この 2 つの場合の塗られ方の個数の合計です。x 座標を下向き、y 座標を右向きにとります。

左側から右へ動く最初の人が全体で S 番目、右側から左へ動く最初の人が全体で T 番目に動くとします。そのような人が存在しない場合、S, T は ∞ とします。最初に動いた人より左側から縦方向に、全体 S 番目以前に動く人たちの中で最も左側に位置する人の y 座標を L とし、最初に動いた人より右側から縦方向に、全体 T 番目以前に動く人たちの中で最も右側に位置する人の y 座標を R とします。

このとき、求める場合の数は、

- 最初に動く人が左側から右へ動くという条件で、人が y 座標 $L-1$ 以下の領域を塗る場合の数
- 最初に動く人が右側から左へ動くという条件で、人が y 座標 $R+1$ 以上の領域を塗る場合の数
- $L \leq y \leq R$ で、$(0, y)$ にも $(N+1, y)$ にも人がいるような y の個数を K として、2^K

の 3 数の積を、y 座標 L, R に人のいるような全ての $1 \leq L \leq R \leq M$ について足し合わせたものとなります。上に挙げた最初の 2 つの値が各 L, R について全て求まれば、$O(M)$ 時間で簡単に答えを求めることができます。

以下、上に挙げた最初の 2 つの値を求めることを考えます。対称性より、最初の値を求めることのみ考えればよいです。

もし盤面左側に人がいなければ、この値は、

- y 座標 $L-1$ 以下の領域に縦方向に動く人が存在しない場合、1
- そうでない場合、0

となります。

そうでない場合、さて、左側から右へ動く人が X 人いるとし、y 座標 $L-1$ 以下の領域に上側から下へ動く人が Y 人、下側から上へ動く人が Z 人いるとします。左から右へ動く人の存在する座標だけを取ってきて、座標を圧縮しておきます。

先ほどと同様、上側から下へ動く最初の人が全体で P 番目、下側から上へ動く最初の人が全体で Q 番目に動くとして、最初に動いた人より上側から横方向に、全体 P 番目以前に動く人たちの中で最も上側に位置する人の x 座標が G で、最初に動いた人より下側から横方向に、全体 Q 番目以前に動く人たちの中で最も下側に位置する人の x 座標が H であるとします。

さて、上側から下へ動く人の進む距離をすべて決め、また下側から上へ動く人の進む距離もすべて決めてやれば、左側から右へ動く人の進む距離もすべて決まることになります。

上側から下へ動く人の進む距離の組は、

- 進む距離は $G-1$ 以下
- どの人の進む距離も、自分より左にいる人の進む距離以下である

の 2 条件を満たすもの全てで、過不足なく列挙されます。これは、上側から下へ動く人の進む領域と、左

側から右へ動く人の進む領域との境目のなす経路を考えれば、グリッド上を $(1,1)$ のマスの左上の格子点から (G,L) のマスの左上の格子点まで、すべての横線と上側から下へ動く人の存在するような列の左側の縦線のみを使って、最短経路で進む場合の数に等しいことが分かります。下側に関しても同様です。

さて、こうして作った「境目のなす経路」をつなげて、x 座標 G 以上のところを y 座標 $L-1$ と L の境目の縦線で折り返せば、この経路としてありうるものは、グリッド上を $(0,0)$ から $(X,Y+Z)$ まで最短経路で進み、かつ y 座標 Y の地点では 1 回以上縦方向に進むような場合の数と等しくなります。これは、$\binom{X+Y+Z-1}{X-1}$ であることが簡単にわかります。

以上をまとめると、各 L に対し、

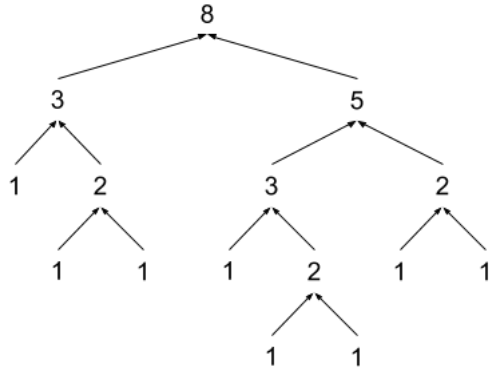
- $E=0$ で x 座標 $U-1$ 以下の領域に左側から右へ動く人が存在しない場合、1
- $E=0$ で x 座標 $U-1$ 以下の領域に左側から右へ動く人が存在する場合、0
- $E>0$ の場合、$\binom{X+Y+Z-1}{X-1}$

を、また各 R に対しても同様のものを求め、これらの値を用いて左から順に適切に求めたい値を計算していくことで、最初に動く人が縦方向に動く場合の塗り方の総数を数えることができます。最初に動く人が横方向に動く場合についても同様に計算することで、$O(N+M)$ 時間でこの問題を解くことができます。

F : Squeezing Slimes

数列 a の両端にそれぞれ 1 を追加しても，答えは変わりません．そこで，簡単のため，a の先頭および末尾の要素は 1 であると仮定します．

各 i $(1 \leq i \leq N)$ に対して，サイズ 1 のスライム a_i 匹から，サイズ a_i のスライム 1 匹までの，合成の過程を 2 分木で表すことを考えます．次図は，$a_i = 8$ の場合の一例です．この 2 分木に対して，葉の深さを左から順に並べた数列を $(d_{i,1}, \ldots, d_{i,a_i})$ とします．次図の例では，$(d_{i,1}, \ldots, d_{i,8}) = (2, 3, 3, 3, 4, 4, 3, 3)$ となります．



$i = 1, \ldots, N$ の順に $(d_{i,1}, \ldots, d_{i,a_i})$ を連結して得られる数列を，$(d_1, \ldots, d_A)$ とします．このとき，各スライムの合成の過程が各 2 分木に合致するような操作列のうち，操作回数の最小値は $\frac{1}{2}\sum_{i=1}^{A-1} |d_i - d_{i+1}|$ となることが示せます (後述の証明 A)．よって，$\sum_{i=1}^{A-1} |d_i - d_{i+1}|$ を最小化するような $(d_1, \ldots, d_A)$ を求めればよいことになります．

i $(1 \leq i \leq N)$ をひとつ固定します．$b_i = \lfloor \log_2 a_i \rfloor$, $c_i = \lceil \log_2 a_i \rceil$ とします．a_i が 2 の冪のときに限り，$b_i = c_i$ となります．このとき，$(d_{i,1}, \ldots, d_{i,a_i}) = (b_i, \ldots, b_i, c_i, \ldots, c_i)$ となるような 2 分木，および $(d_{i,1}, \ldots, d_{i,a_i}) = (c_i, \ldots, c_i, b_i, \ldots, b_i)$ となるような 2 分木が，それぞれ存在します．さらに，$\sum_{i=1}^{A-1} |d_i - d_{i+1}|$ を最小化するためには，$(d_{i,1}, \ldots, d_{i,a_i})$ としては $(b_i, \ldots, b_i, c_i, \ldots, c_i)$ または $(c_i, \ldots, c_i, b_i, \ldots, b_i)$ のみを試せば十分であるということが示せます (後述の証明 B)．

以上より，DP によって最小値が求まります．時間計算量は $\mathrm{O}(N)$ です．

証明 A

元の問題における操作を，数列 $(d_1, \ldots, d_A)$ に対する操作として言い換えると，次のようになります．

- 数列上の長さ M (M は偶数) の区間を選ぶ．区間内の要素を左から順に 2 個ずつペアにする．このとき，次の条件が成り立っている必要がある．
  - 各ペアに対して，2 つの要素は同一の値であり，かつその値は 0 より大きい．

  そして，各ペアに対して，2 つの要素を合成して 1 つの要素にする．合成後の要素の値は，合成前の要素の値より 1 だけ小さい値とする．

この操作を繰り返し，数列の要素をすべて 0 にするのが目標です．

まず，操作回数の下限が $\frac{1}{2}\sum_{i=1}^{A-1}|d_i - d_{i+1}|$ であることを示します．各 i $(1 \leq i \leq A-1)$ に対して，d_i と d_{i+1} は最終的に同一の値 0 になる必要があります．d_i と d_{i+1} の差を 1 だけ縮めるためには，操作を行う区間の左端または右端が，d_i と d_{i+1} の間に来る必要があります．よって，操作回数は $\frac{1}{2}\sum_{i=1}^{A-1}|d_i - d_{i+1}|$ 以上でなければなりません．

次に，操作回数の上限が $\frac{1}{2}\sum_{i=1}^{A-1}|d_i - d_{i+1}|$ であることを示します．そのために，実際に長さ $\frac{1}{2}\sum_{i=1}^{A-1}|d_i - d_{i+1}|$ の操作列を構成します．具体的には，次のルールに従って操作を繰り返します．

- 数列の要素の最大値を d_{max} とする．値が d_{max} である要素の連続区間のうち，極大なものをひとつ選ぶ．このとき，区間の長さは偶数である．そうでないならば，以降どのように操作を行っても，値が d_{max} である要素が残ってしまうことになり，矛盾する．この区間に対して操作を行うと，区間の左端 l と右端 r に対して，$|d_l - d_{l+1}|$ と $|d_r - d_{r+1}|$ はそれぞれ 1 ずつ減る．

よって，以上の操作はちょうど $\frac{1}{2}\sum_{i=1}^{A-1}|d_i - d_{i+1}|$ 回で終了します． □

証明 B

$a_i \geq 2$ と仮定します．$(d_1, \ldots, d_A)$ 上で，$(d_{i,1}, \ldots, d_{i,a_i})$ の直前にある要素の値を p とし，直後にある要素の値を q とします．また，$(d_{i,1}, \ldots, d_{i,a_i})$ の要素の最小値を d_{min} とし，最大値を d_{max} とします．このとき，$d_{min} \leq b_i \leq c_i \leq d_{max}$ が成り立ちます．$d_{i,l} = d_{min}$ であるような l と，$d_{i,r} = d_{max}$ であるような r を，それぞれひとつずつ選びます．以降は $l < r$ と仮定して議論を進めます ($l > r$ の場合も同様です)．すると，$|p - d_{i,1}| + \sum_{j=1}^{a_i-1}|d_{i,j} - d_{i,j+1}| + |d_{i,a_i} - q| \geq |p - d_{min}| + |d_{min} - d_{max}| + |d_{max} - q|$ が成り立ちます．

一方，$(d_{i,1}, \ldots, d_{i,a_i}) = (b_i, \ldots, b_i, c_i, \ldots, c_i)$ の場合，$|p - d_{i,1}| + \sum_{j=1}^{a_i-1}|d_{i,j} - d_{i,j+1}| + |d_{i,a_i} - q| = |p - b_i| + |b_i - c_i| + |c_i - q|$ です．ここで，$d_{min} \leq b_i \leq c_i \leq d_{max}$ より，$|p - d_{min}| + |d_{min} - d_{max}| + |d_{max} - q| \geq |p - b_i| + |b_i - c_i| + |c_i - q|$ が成り立ちます．よって，$l < r$ の場合，$(d_{i,1}, \ldots, d_{i,a_i}) = (b_i, \ldots, b_i, c_i, \ldots, c_i)$ に取り替えることで，$\sum_{i=1}^{A-1}|d_i - d_{i+1}|$ を広義減少させることができます．

以上より，$\sum_{i=1}^{A-1}|d_i - d_{i+1}|$ を最小化するためには，$(d_{i,1}, \ldots, d_{i,a_i})$ としては $(b_i, \ldots, b_i, c_i, \ldots, c_i)$ または $(c_i, \ldots, c_i, b_i, \ldots, b_i)$ のみを試せば十分です． □

CODE FESTIVAL 2017 Qualification Round A Editorial

sugim48 and DEGwer

2017/09/23

A: Snuke's favorite YAKINIKU

Read the input and compare its first 4 characters with "YAKI".

Here is an example C/C++ implementation:

```
#include<stdio.h>
int main()
{
        char s[100];
        scanf("%s", s);
        if (s[0] == 'Y'&&s[1] == 'A'&&s[2] == 'K'&&s[3] == 'I') printf("Yes\n");
        else printf("No\n");
}
```

B: fLIP

The order of pressing buttons doesn't matter, and it doesn't make sense to press the same button multiple times.

Suppose that we press exactly k row-buttons and l column-buttons. Then, the number of black squares will be $k(M-l)+(N-k)l$.

Thus, we can brute force all pairs (k,l), and check whether we can satisfy $k(M-l)+(N-k)l=K$. The complexity of this solution is $O(NM)$.

Exercise: can you solve this problem in $O(N)$?

C : Palindromic Matrix

The condition is satisfied if and only if for each (i, j) $(1 \leq i \leq H, 1 \leq j \leq W)$, $a_{i,j} = a_{i,W+1-j} = a_{H+1-i,j} = a_{H+1-i,W+1-j}$ holds.

For example, when $H = 3, W = 5$, the matrix will look as follows:

$$\begin{pmatrix} a & b & c & b & a \\ d & e & f & e & d \\ a & b & c & b & a \end{pmatrix}$$

Here, if two cells are labelled with the same label, the two cells must contain the same letter.

As we see above, the cells in the matrix will be divided into "groups" of sizes $1, 2$, or 4. Let g_1, g_2, g_4 be the number of groups of size $1, 2, 4$. Now, we want to divide the letters in A into groups, such that

- There are exactly g_1 groups of size 1, g_2 groups of size 2, and g_4 groups of size 4.
- In each group, all the letters in the group are the same.

For each $c = \mathrm{a}, \mathrm{b}, ..., \mathrm{z}$, let $count_c$ be the number of occurrences of c in A.

First, repeat the following operation g_4 times:

- Choose an arbitrary c such that $count_c \geq 4$ and decrease $count_c$ by 4. If such c doesn't exist, the answer is `No`.

Then, repeat the following operation g_2 times:

- Choose an arbitrary c such that $count_c \geq 2$ and decrease $count_c$ by 2. If such c doesn't exist, the answer is `No`.

First, repeat the following operation g_1 times:

- Choose an arbitrary c such that $count_c \geq 1$ and decrease $count_c$ by 1. If such c doesn't exist, the answer is `No`.

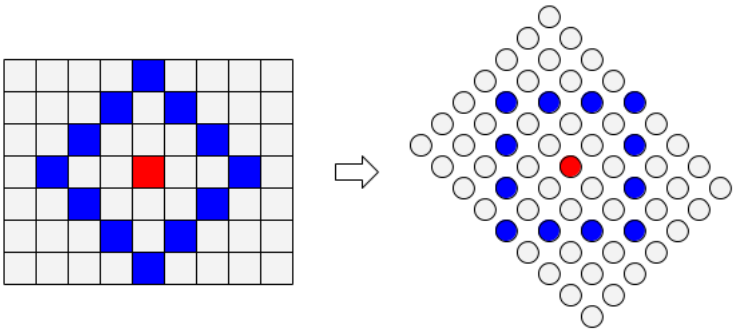
If we can successfully complete these operations, the answer is `Yes`.

D : Four Coloring

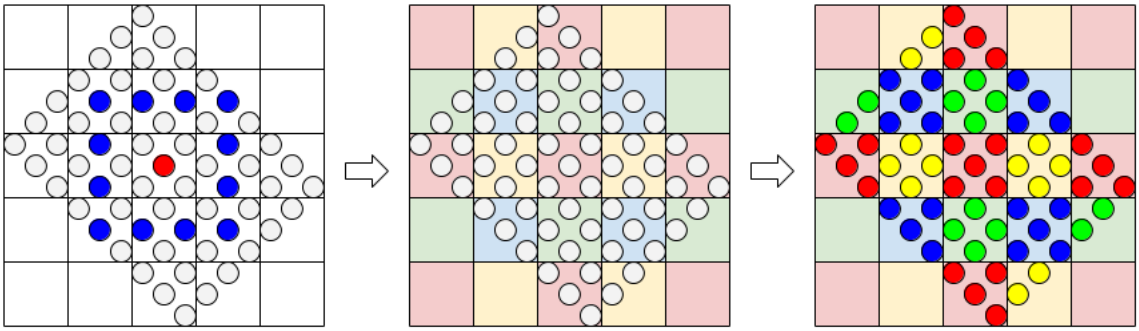
Since this problem uses Manhattan Distance, let's rotate the grid by 45 degrees. The point (i, j) $(1 \leq i \leq H,\ 1 \leq j \leq W)$ is mapped to the point $(i + j, i - j)$.

Now, we must color two points with distinct colors if the Chebyshev Distance between them is d. (Here, Chebyshev Distance is defined as $\max\{|x_1 - x_2|, |y_1 - y_2|\}$.)

Here is an example when $d = 3$:



Let's divide the entire plane into $d \times d$ square parts (picture on the left). You can notice that if Chebyshev Distance between two points P and Q is d, Q is in one of 8-neighbors of P's part. Thus, we can satisfy the condition if we color all points in the same part with the same color, and the color is different from all of its 8-neighbors. For example, choose colors as in the picture in the middle.



E: Modern Painting

If no person exists, the answer is 1. Otherwise, the movement of the first person is either vertical or horizontal. We try both possibilities: by symmetry, we assume that the first person moves vertically.

In this case, the first person completely dominates its column. In general, there can be multiple people who dominates a column. Assume that each column contains at least one person (otherwise, delete unnecessarily columns). Then, the set of columns dominated by a single person forms an interval: let $[L, R]$ be the indices of those columns.

For a fixed $[L, R]$, the number of possible states of the board is the product of the following three values:

- Only consider vertical people in the leftmost $L - 1$ columns and horizontal people facing right. The number of different states formed by these people, under the constraint that one of horizontal people must move first.
- Only consider vertical people in the rightmost $M - R$ columns and horizontal people facing left. The number of different states formed by these people, under the constraint that one of horizontal people must move first.
- The value 2^k, where k is the number of columns between L-th and R-th that have two people.

Thus, the answer (assuming that the first person moves vertically) is the sum of the product of these three values over all pairs (L, R) $(1 \leq L \leq R \leq M)$. If we can pre-compute the first two values for each L, R, it's easy to get the answer in $O(M)$.

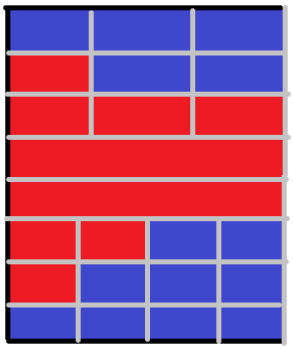
How can we compute the first value for each L (and similarly, the second value)? Let X be the number of people facing right, Y be the number of people facing down in the first $L - 1$ columns, and Z be the number of people facing up in the first $L - 1$ columns.

If $X = 0$, obviously, the answer is

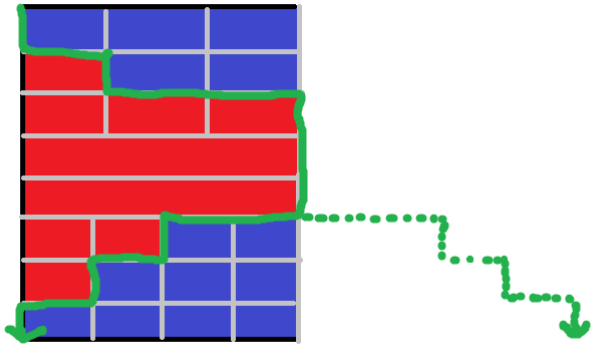
- If $Y = Z = 0$, 1.
- Otherwise, 0.

If $X > 0$, at least one person dominates a row. Let's compress the grid: there are X rows, there are Y columns in the upper part, and Z columns in the lower part (the two parts are separated by dominated rows).

Now the grid will look as follows (here red cells are painted by horizontal people, blue cells are painted by vertical people):



Draw a path along the border of red and blue parts, and let's "flip" the lower part:



You can see that the answer corresponds to the number of paths from $(0, 0)$ to $(X, Y + Z)$, under the constraint that you must move vertically at least once along the flipped line. This is $\binom{X+Y+Z}{X} - \binom{X+Y+Z-1}{X} = \binom{X+Y+Z-1}{X-1}$.

In summary, we compute the following value for each L (here X, Y, Z are defined above):

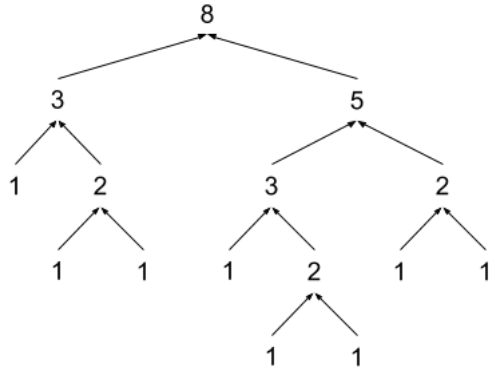
- If $X = Y = Z = 0$, 1.
- Otherwise, if $X = 0$, 0.
- Otherwise, $\binom{X+Y+Z-1}{X-1}$.

Compute similar values for each R, and compute the sum of product of the three values described above in $O(M)$. Do the same for the case where the first person moves horizontally. This solution works in $O(N + M)$ time.

F : Squeezing Slimes

Suppose that by some operations, we get a slime of size a from a slimes of size 1. Let x be the total number of operations, and let l, r be the number of operations involving the leftmost slime and the rightmost slime, respectively.

For example, consider the following example:



Note that some mergeings are performed simultaneously. In particular, we got $(3, 5)$ from $(1, 2, 3, 2)$ and $(2, 3, 2)$ from $(1, 1, 1, 2, 1, 1)$. Here, $(l, x, r) = (2, 4, 3)$.

Let's return to the original problem. For each a_i, we can assign a triplet (l_i, x_i, r_i), as defined above. If we perform the operations for each a_i independently, the total number of operations will be $\sum a_i$. However, we can reduce the number of operations by performing operations between a_i and a_{i+1} simultaneously $\min\{r_i, l_{i+1}\}$ times. Thus, the total number of operations will be $\sum_{i=1}^{N} a_i - \sum_{i=1}^{N-1} \min\{r_i, l_{i+1}\}$.

Now, what we have to do is to choose a triplet for each a_i. We want x_i to be small, while we want l_i, r_i to be big. It turns out that we only need to consider the following ways to decide the triplet (formal proof at the end):

- If $a_i = 2^k$, $(l_i, x_i, r_i) = (k, k, k)$.
- If $2^k < a^i < 2^{k+1}$, $(l_i, x_i, r_i) = (k, k+1, k+1) or (k+1, k+1, k)$.

Since there are at most two candidates for each a_i, we can compute the minimum of $\sum_{i=1}^{N} a_i - \sum_{i=1}^{N-1} \min\{r_i, l_{i+1}\}$ by a simple DP. The complexity of this solution is $\mathrm{O}(N)$.

Proof

First, we need to prove that the triplets we wrote above are valid. (k, k, k) for $a = 2^k$ is obvious, so we want to construct an example of $(k, k+1, k+1)$ when $2^k < a^i < 2^{k+1}$. Let's use an induction: assume that $k > 1$, and we constructed examples for smaller k. (Note that if (l_1, x_1, r_1) is valid for a_1 and (l_2, x_2, r_2) is valid for a_2, $(l_1, x_1 + x_2 - \min\{r_1, l_2\}, r_2)$ is valid for $a_1 + a_2$.)

- If $2^k < a < 2^k + 2^{k-1}$, we can combine $(k-1, k-1, k-1)$ for 2^{k-1} and $(k-1, k, k)$ for $a - 2^{k-1}$.
- If $a = 2^k + 2^{k-1}$, we can combine $(k-1, k-1, k-1)$ for 2^{k-1} and (k, k, k) for 2^k.
- If $2^k + 2^{k-1} < a$, we can combine $(k-1, k, k)$ for $a - 2^k$ and (k, k, k) for 2^k.

We also need to prove that the triplets above are optimal.

We use the following two facts:

- If (l, x, r) is a valid triplet for a, $a \leq 2^x$. This is because in an operation the number of slimes can't be less than half of the number of slimes before the operation.
- If (l, x, r) is a valid triplet for a, $2^{l+r-x} \leq a$. This is because in order to increase the value $l+r-x$, we need to perform an operation for all slimes, and each time the number of slimes is halved.

By these two facts, we can see that the triplets above are optimal in terms of both x and $l + r - x$. Then, it's easy to check that these triplets also minimizes the value $\sum_{i=1}^{N} a_i - \sum_{i=1}^{N-1} \min\{r_i, l_{i+1}\}$.