AGC032 解説

writer : many

2019 年 3 月 23 日

For International Readers: English editorial starts on page 9.

A: Limited Insertion

操作を逆順に見ると問題は以下のように言い換えられます。

> 数の書かれた N 個のボールが一列に並んでいる。すぬけ君は左から k 番目のボールに k と書いてあるとき、それを選んで取り除くことができる。全てのボールを取り除くことは可能か？

すぬけ君は取り除くことが可能なボールのうち、最も右側にあるものを選ぶしかありません。そうでないとすると二度と取り除くことができないボールが生じてしまい、全てのボールを取り除くことが不可能になることから容易に示されます。以上のことから操作手順は存在するならば実は 1 通りしかないことも分かります。

上記の戦略は $O(N^2)$ でシミュレート可能なため、実行時間にも十分余裕があります。

B: Balanced Neighbors

以下のような条件を満たすようなグラフを考えます。

- 非連結
- 任意の頂点について、自身の番号と隣接する頂点の番号の和が一定

この条件は補グラフが問題文中の条件を満たすための十分条件となっています。この条件を考えると、$\{1,\ldots,N\}$ を 2 つ以上のグループに分割する方法であって、グループに含まれる数の和がグループによらず一定となるような方法を 1 つ見つけられればいいことが分かります。

そのような方法が存在したなら、同じグループに属するような頂点間に辺を張ったグラフが上の条件を満たします。

分割方法の方法の例としては以下の方法があります。

- N が奇数のとき： $\{1,N-1\},\{2,N-2\},\ldots,\{i,N-i\},\ldots,\{N\}$
- N が偶数のとき： $\{1,N\},\{2,N-1\},\ldots,\{i,N+1-i\}$

上の方法で構成されるグラフの補グラフは $O(N^2)$ で構成可能であり、実行時間にも十分余裕があります。

C: Three Circuits

与えられたグラフ G がオイラーグラフでないとき、明らかに答えは `No` です。以降、G はオイラーグラフであることを仮定します。結論から言えば、以下のいずれかの条件を満たすときのみ答えは `Yes` です。

- 次数 6 以上の頂点が存在する
- 次数 4 の頂点が 3 つ以上存在する
- 次数 4 の頂点が 2 つ存在し、その 2 つの頂点をつなぐパスはちょうど 2 つある

次数 6 以上の頂点が存在する場合

そのような頂点 A を始点とするオイラーサイクルを 1 つ考えます。また、A の次数を D とします。このオイラーサイクルは、明らかに A を始点とする $\frac{D}{2}$ 個のサーキットに分解可能です。また、2 つのサーキットを 1 つにまとめられることも明らかです。よって 3 個のサーキットを構成可能です。

次数 4 の頂点が 3 つ以上存在する場合

上記の場合を除くと、次数 4 の頂点と次数 2 の頂点のみ含まれるグラフを考えればいいことが分かります。以降は全ての頂点は次数 2 か 4 であることを仮定します。

次数 4 の頂点 A を始点とするオイラーサイクルを 1 つ考えます。上記の議論から A を始点とする 2 つのサーキットを作ることができます。図 1 のように、このサーキットの少なくとも一方において、2 回訪問される頂点 B が存在したなら、B を始点とするサーキットを追加で 1 つ作ることができ、3 つのサーキットを構成可能です。そうでない場合、それぞれのサーキットでちょうど 1 回ずつ訪問されるような頂点 B, C が存在します。図 2 のように $A \to B, B \to C, C \to A$ というパスを 2 つずつ取れるので、これらを組み合わせて 3 つのサーキットが構成可能です。

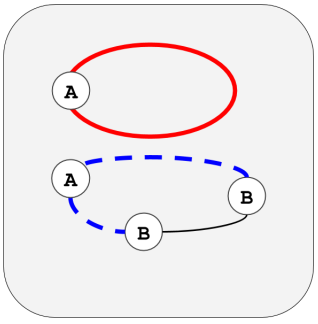

図 1　2 つのサーキットの少なくとも一方に 2 回訪問される頂点が存在する場合

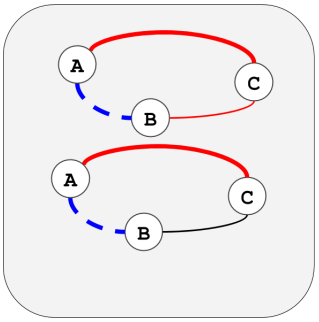

図 2　それ以外の場合

次数 4 の頂点が 2 つ存在する場合

ありうるグラフは以下の 2 通りです。A と B をつなぐパスが 2 つあるときのみ 3 つのサーキットが構成できることがわかります。例えば、A-B 最小カットを取るなどの方法で判定をすることができます。

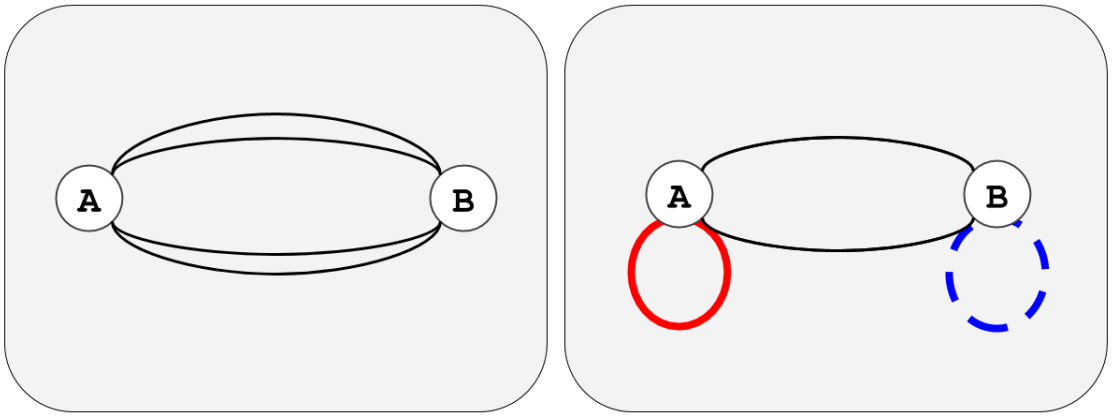

図 3　次数 4 の頂点が 2 つ存在するようなグラフ

次数 4 の頂点が 1 つ以下の場合は明らかです。いずれの場合も $O(N+M)$ で判定可能であり、十分高速です。

D - Rotation Sort

2 種類の操作はそれぞれ次のように言い換えられます.

- コスト A を支払い，ある要素を右側へ動かす.
- コスト B を支払い，ある要素を左側へ動かす.

見通しを良くするために，問題を次のように言い換えます.

> 最初，各 i $(1 \leq i \leq N)$ について要素 p_i が座標 i に置かれている．あなたは，次の 2 種類の操作を好きな順序で繰り返し行うことができる.
>
> - コスト A を支払い，ある要素を右側の (**整数とは限らない**) 座標へ動かす.
> - コスト B を支払い，ある要素を左側の (**整数とは限らない**) 座標へ動かす.
>
> 最終的に，N 個の要素が左から右へ昇順に並んでいるようにしたい．必要な総コストの最小値を求めよ.

明らかに，ある要素を 2 回以上動かすのは無駄です．よって，要素 k の最初の座標を x_k とし，最後の座標を x'_k とすると，この要素に関するコストは次のようになります.

$$\begin{cases} 0 & (x_k = x'_k) \\ A & (x_k < x'_k) \\ B & (x_k > x'_k) \end{cases}$$

このコストの総和を最小化するように，N 個の要素の最後の座標 $x'_1 < x'_2 < \cdots < x'_N$ を決めるのが目標です.

要素 $1, 2, \ldots, N$ の順に座標を決めて置いていくことにすると，次のような DP が可能です.

$\mathrm{dp}[k][x'] :=$ (要素 $1, 2, \ldots, k$ を既に置いており，要素 k の座標が x' であるときの，今までの総コストの最小値)

しかし，x' は連続値なので，このままでは計算ができません．よく考えると，$(-\infty, 1), (1, 2), \ldots, (N-1, N), (N, \infty)$ の各開区間については，x' の正確な値を覚えておく必要はなく，「x' がこの開区間に含まれる」ということだけ覚えておけば十分です．よって，x' の取りうる値は $1, 2, \ldots, N$ とこれらの開区間だけで十分であり，計算ができるようになりました.

愚直に実装すると時間計算量は $\mathrm{O}(N^3)$ で TLE しますが，適当な高速化によって $\mathrm{O}(N^2)$ へ改善できます.

E - Modulo Pairing

まず，a を昇順にソートしておきます．以降，ペア分けを図 1 のように表します．ここで，青は $x+y<M$ なるペアを表し，赤は $x+y\geq M$ なるペアを表します．

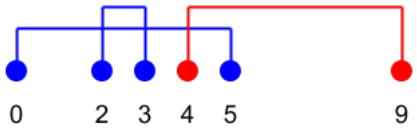

図 1

実は，図 2 のようなペア分けが最適解のひとつとして存在します (証明は後述)．

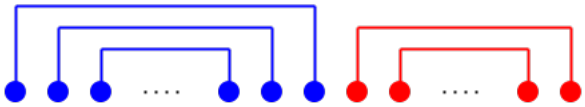

図 2

よって，青と赤の境界線の位置を全探索することで，最適値が求まります．境界線の位置をひとつ固定した後は，境界線の左側 (右側) で図 2 のようにペア分けし，各ペアが青 (赤) であることを確かめた後，全ペアの醜さの最大値を求めればよいです．しかし，この方法の時間計算量は $\mathrm{O}(N^2)$ で，TLE してしまいます．

この方法を高速化するために，次の事実に注目します．

- 境界線が左であるほど，境界線の左側の各ペアは青になりやすくなる．
- 境界線が左であるほど，境界線の右側の各ペアは赤になりにくくなる．
- 境界線が左であるほど，全ペアの醜さの最大値は小さくなる．

よって，境界線の位置を二分探索し，「境界線の右側の各ペアが赤である」という条件を満たすような最も左の境界線を求め，そのときの全ペアの醜さの最大値を求めれば，それが答えです．この方法の時間計算量は $\mathrm{O}(N\log N)$ であり，十分高速です．

図 2 のような最適解が存在することの証明

図 3 はペア分けの局所的な組み換えを表しています．ペア分けの中に矢印の左側のような 2 ペアが存在する場合，矢印の右側のような 2 ペアに組み換えることが常に可能であり，このとき，これら 2 ペアの醜さの最大値が大きくなることはありません．このような組み換えを繰り返すことで，任意の解を図 2 のような解に変形できることが示せます．さらに，このとき全ペアの醜さの最大値が大きくなることはありません．よって，図 2 のようなペア分けが最適解のひとつとして存在します．

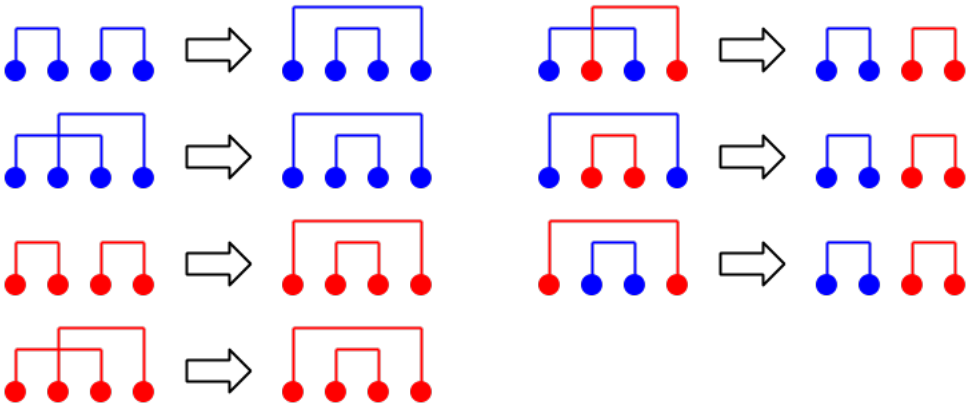

図 3

F - One Third

それぞれの切れ込みに沿って赤い線を引くことにします。また、各赤線に対し、それを時計回りに 120 度回転させた線を青で、反時計回りに 120 度回転させた線を緑で引きます。このとき、値 $|x - 1/3|$ は二本の異なる色の線分がなす最小の角度であることが容易にわかります。

よって、この問題は次と同値です。

> 数直線上の点 0 に赤い点を、点 1/3 に青い点を打ち、次の操作を $N - 1$ 回繰り返す:「0 以上 1/3 以下の座標を一つ、一様にランダムに選ぶ。また、赤青緑から一色をランダムに選ぶ。選んだ位置に選んだ色の点を打つ」。操作後、異なる色の二点間の最小距離の期待値はいくつか。

操作後、区間 $[0, 1/3]$ は数直線上にある $N + 1$ 個の点によって N 個の断片に分割されます。両端の点の色が異なるような断片を **カラフル** であると呼ぶことにすると、最短のカラフルな断片の長さを求めたいです。

次の値を計算しましょう。

- カラフルな断片の個数が k となる確率を $f(k)$ とします。これは、赤青緑の三色からなる長さ $N + 1$ の列であって、先頭が赤、末尾が青であり、異なる二色が隣接する箇所が k 個存在するようなものの個数を 3^{N-1} で割ったものです。単純な動的計画法と二項係数によって求められます。
- カラフルな断片の個数が k であるときの最短のカラフルな断片の長さの期待値を $g(k)$ とします。この場合の k 個の断片の長さの総和の期待値は $k/3n$ です。また、単純な計算により、線分を k 個の断片にランダムに分割するとき、最短の断片の長さの期待値は元の線分の長さの $1/k^2$ 倍であることがわかります。よって、$g(k) = 1/3nk$ です。

k が取りうるすべての値に対する $f(k)g(k)$ の和が答えです。

AGC032 Editorial

writer : many

March 23, 2019

A: Limited Insertion

By reversing the timeline, the problem can be rephrased into the following:

> We have N balls, each with an integer written on it. If the k-th ball from the left has the number k, Snuke can remove that ball. Can he remove all the balls?

Snuke has no choice but to remove the rightmost ball that can be removed. This can be easily shown from the fact that other moves will produce a ball that cannot be removed forever. From this, it can also be seen that the way to remove all the balls is unique if it exists.

The strategy above can be simulated in $O(N^2)$ time, which runs in time easily.

B: Balanced Neighbors

Consider a graph satisfying the following conditions:

- Not connected.
- For every vertex, the sum of the indices of the vertices adjacent to that vertex is constant.

These conditions are sufficient for its complement graph to satisfy the conditions in the statement. Thus, if we can find a way to divide $\{1, \ldots, N\}$ into two or more groups so that the sum of the integers in a group is constant, we have an instance of the above graph by connecting vertices belonging to the same group, and thus we have a solution.

One way to divide $\{1, \ldots, N\}$ is as follows:

- If N is odd: $\{1, N-1\}, \{2, N-2\}, \ldots, \{i, N-i\}, \ldots, \{N\}$
- If N is even : $\{1, N\}, \{2, N-1\}, \ldots, \{i, N+1-i\}$

The complement graph of the graph generated by the division above can be built in $\mathrm{O}(N^2)$ time, which is sufficiently fast.

C: Three Circuits

If the given graph G is not Eulerlian, the answer is obviously `No`. From now on, we assume that G is Eulerlian. The conclusion is: the answer is `Yes` if and only if the following condition is satisfied:

- There exists a vertex with degree 6 or greater.
- There exists three or more vertices with degree 4.
- There exists two vertices with degree 4, and exactly two paths connecting them.

If there is a vertex with degree 6 or greater

Consider a Eulerian cycle beginning with such a vertex A, and let D be the degree of A. This Eulerian cycle can obviously divided into $\frac{D}{2}$ circuits beginning with A. It is also obvious that two of these circuits can be merged into one. Thus, we can build three circuits.

If there are three or more vertex with degree 4

Excluding the above case, we only need to consider graphs that only contains vertices of degree 4 or 2. From now on, we assume that the degree of every vertex is 2 or 4.

Consider a Eulerian cycle beginning with a vertex A of degree 4. From the discussion similar to the above, we can build two circuits beginning with A. If there is a vertex B visited twice in at least one of them, as shown in Fig. 1, we can build one more circuit beginning with B, for a total of three. Otherwise, there exist vertices B and C that is visited exactly once in each of the two circuits. As shown in Fig. 2, we can take two paths in each of the forms $A \to B, B \to C, C \to A$, and we can combine them to form three circuits.

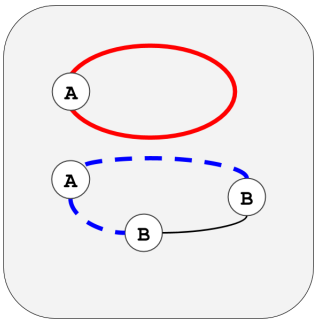

Fig.1 The case where there is a vertex visited twice in at least one of the circuits

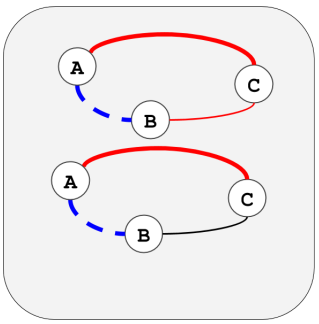

Fig.2 The other case

If there are two vertices with degree 4

The two possible forms of the graph are shown below. It can be seen that three circuits can be built if and only if there are two paths connecting A and B. One way to check this is to find an A-B minimum cut.

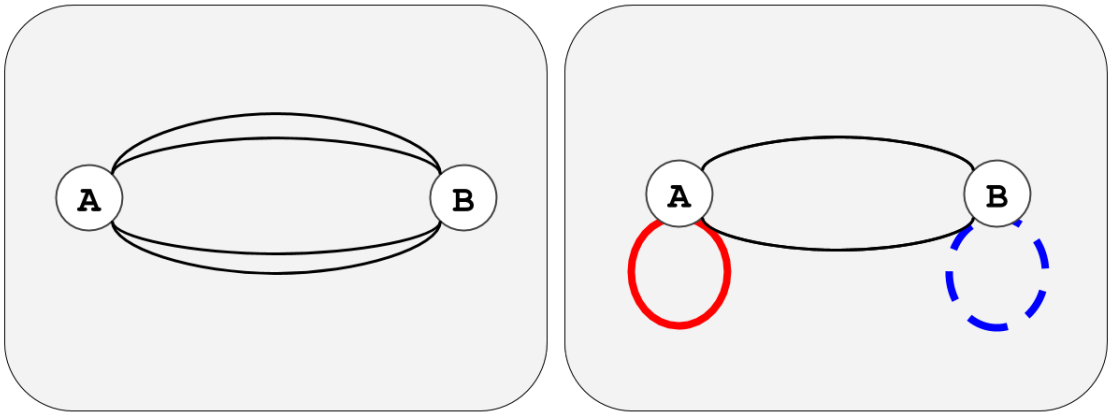

Fig.3 Graphs with two vertices with degree 4

The answer is obvious for a graph with one or less vertex with degree 4.
Therefore, in all the cases we can determine the answer in $\mathrm{O}(N + M)$ time, which is fast enough.

D - Rotation Sort

We can rephrase the two kinds operations as follows:

- Pay a cost A to move an element to some position to the right.
- Pay a cost A to move an element to some position to the left.

Now we rephrase the problem to make it more approachable, as follows:

> Initially, for each i ($1 \leq i \leq N$), an element p_i is placed at coordinate i. You can perform the following two kinds of operations any number of times in any order:
>
> - Pay a cost A and move an element to some coordinate (*not necessarily an integer*) to the right.
> - Pay a cost A and move an element to some coordinate (*not necessarily an integer*) to the right.
>
> In the end, the N elements should be sorted in ascending order from left to right. Find the minimum total cost required.

Obviously, it is unnecessary to move an element two or more times. Thus, the cost for element k is as follows, where x_k and x'_k are its initial and final coordinates:

$$\begin{cases} 0 & (x_k = x'_k) \\ A & (x_k < x'_k) \\ B & (x_k > x'_k) \end{cases}$$

Our objective is to determine the final coordinates of the N elements, $x'_1 < x'_2 < \cdots < x'_N$, so that the sum of these costs is minimized.

By deciding the final coordinates for the elements $1, 2, \ldots, N$ in this order, we have the following DP solution:

$$\mathrm{dp}[k][x'] := (\text{Minimum total cost to place elements } 1, 2, \ldots, k \text{ with } k \text{ placed at coordinate } x')$$

However, this does not work as is, since x' takes a continuous value. We can bypass this problem by not memorizing the exact value of x', and instead memorizing which of the open intervals $(-\infty, 1), (1, 2), \ldots, (N-1, N), (N, \infty)$ contains x', if it is contained in one of them. Thus, the only "values" that x' can take are $1, 2, \ldots, N$ and these open intervals, and now the solution actually works.

The time complexity is $\mathrm{O}(N^3)$ if naively implemented, which gets TLE, but it can be improved to $\mathrm{O}(N^2)$ by properly speeding it up.

E - Modulo Pairing

First, we sort a in ascending order. From now on, we use the visual representation of the pairing as shown in 4. Here, pairs in blue represent pairs such that $x + y < M$, and pairs in red represent pairs such that $x + y \geq M$.

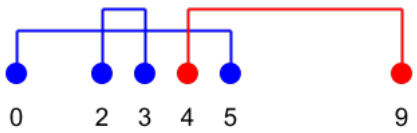

Fig.4

The conclusion is that one optimal pairing is the one such as shown in 5 (we will provide the proof later).

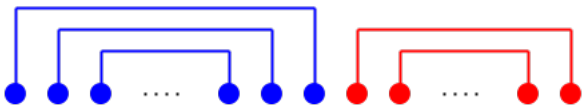

Fig.5

Thus, we can find the optimal solution by trying all possible positions of the border between blue and red. When we fix the border to one position, we make pairs in the left/right of the border as 5, verify that each pair is actually blue/red, and find the largest ugliness of all pairs. However, this method runs in $O(N^2)$ time and gets TLE.

In order to make it faster, we note the following facts:

- When the border is moved left, each pair to the left of the border has more tendency to be blue.
- When the border is moved left, each pair to the right of the border has less tendency to be red.
- When the border is moved left, the largest ugliness of all pairs gets smaller.

Thus, we can find the answer by finding the leftmost possible position of the border such that each pair to the right of the border is red, and finding the largest ugliness of all pairs in that case. This method now runs in $O(N \log N)$ time, which is fast enough.

Proof: there exists a optimal pairing in the form shown in Fig. 5

The following Fig. 6 shows the ways to reconstruct the local pairing. In a pairing, if there are two pairs in a form shown to the left of an arrow, we can always convert them to the corresponding two pairs shown to the right of the arrow, and this does not increase the larger ugliness of the pairs. By repeating this reconstruction, we can show that any pairing can be converted to a pairing in the form shown in

Fig. 5 without increasing the largest ugliness of all pairs. Thus, there exists a optimal pairing in the form shown in Fig. 5

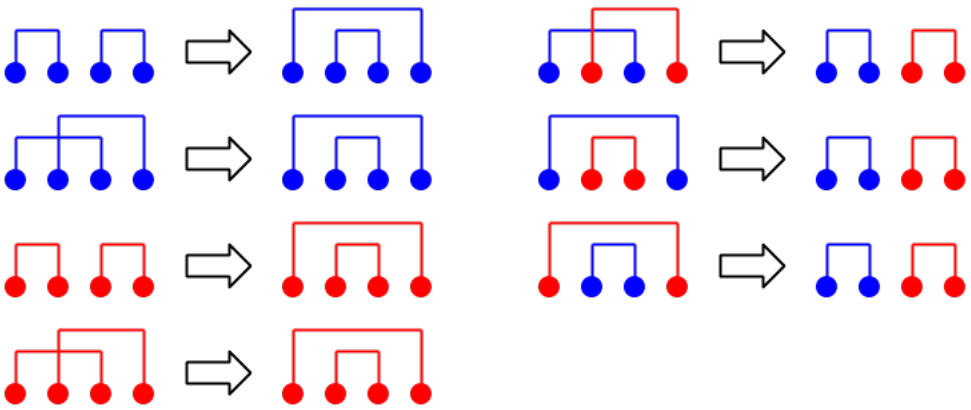

Fig.6

F - One Third

Let's draw a red segment along each cut. Also, for each red segment, draw a blue line 120 degrees clockwise to it, and draw a green line 120 degrees counter-clockwise to it. It's easy to see that the value $|x - 1/3|$ is the smallest angle formed by two segments of different colors.

Therefore, the problem is equivalent to the following:

> On the number line, we plot a red point at 0, and a blue point at 1/3. Then, we repeat the following operation $N - 1$ times: choose a coordinate between 0 and 1/3 uniformly at random, choose one of three colors at random, and draw a point of the chosen color at the chosen position. What is the shortest distance between two points of different colors?

After the operations, there will be $N + 1$ points, and these points will divide the interval $[0, 1/3]$ into N parts. Let's call a part *colorful* if the two points at its ends have different colors. We are interested in the length of the shortest colorful part.

Let's compute the following values:

- Let $f(k)$ be the probability that the number of colorful parts becomes k. This is the number of sequences of colors of length $k + 1$, such that the sequence starts with red, ends with blue, and a pair of adjacent same colors appear k times, divided by 3^{N-1}. This can be computed by a simple DP and binomial coefficients.
- Let $g(k)$ be the expected length of the shortest colorful part in case there are k colorful parts. In this case, the expected total length of the k colorful parts is $k/3n$. A simple calculation shows that when we divide a segment into k parts at random, the expected length of the shortest part is $1/k^2$ times the length of the original segment. Thus, $g(k) = 1/3nk$.

The answer is the sum of $f(k)g(k)$ over all possible k.