AGC 024 解説

DEGwer

2018/05/20

For International Readers: English editorial starts on page 6.

A: Fairness

高橋君の持っている整数から中橋君の持っている整数を引いた値は、操作を 1 回行うとどのように変化するかを考えてみましょう。

高橋君、中橋君、低橋君がそれぞれ整数 $(A+x, A, B)$ を持っているとしましょう。1 回の操作後、3 人はそれぞれ $(A+B, A+B+x, 2A+x)$ を持っています。すなわち、所望の値は x から $-x$ に変化します。より正確に言えば、絶対値は符号が反転します。

よって、操作回数が偶数回なら答えは $A-B$ で、奇数回なら $B-A$ です。

B: Backfront

先頭や末尾に移動されることのない整数たちは、その相対的な順序が変更されることはなく、さらにそれらの間に新たに整数が挿入されることもありません。よって、移動されない整数を小さい順に $x_1, ..., x_k$ とすれば、これらは以下の条件を満たす必要があります。

- 全ての i に対し、$x_i + 1 = x_{i+1}$
- 全ての i に対し、元の順列で x_i は x_{i+1} より前にある

逆に、この条件が満たされれば、$x_1 - 1, ..., 1$ の順に整数を先頭に、$x_k + 1, ..., N$ の順に整数を末尾に移動することで列を昇順に並び替えることができます。よって、このような $x_1, ..., x_k$ に対する k の最大値を求めればよいです。

これは、全ての i に対し $Q[P[i]] = i$ として得られる列 Q を前から見ていくことで簡単に求められます。よってこの問題を $O(N)$ 時間で解くことができました。

C: Sequence Growing Easy

操作によって $X_1 > 0$ や $X_{i+1} - X_i > 1$ となることはないので、$A_1 > 0$ の場合と $A_{i+1} - A_i > 1$ なる i がある場合は答えは -1 です。以下、そうでない場合を考えます。

$A_i = t$ とします。このとき、操作によって X_i に t が代入されるためには、$t = 0$ であるか、または $X_{i-1} = t - 1$ であるようなタイミングが必要です。この議論を繰り返し行うことで、$1 \leq s \leq t$ に対し、

X_{i-t+s} に s が代入されているタイミングが必要となることがわかります。よって、このような $(i-t+s, s)$ の組としてありうるものの個数で求める最小値を下から抑えることができます。この値は、$X_{i-1}+1=X_i$ なら答えに 1 を、そうでないなら答えに X_i を足すことで求めることができます。

逆に、以下を繰り返すことによって、その回数の操作で X と A を一致させることができます。

- $X=A$ なら終了する
- そうでない場合、r を $X_r \neq A_r$ なる最大の r とする
- 各 $r-X_r+1, ..., r$ に対し、順に $X_i = X_{i-1}+1$ とする

よって、この問題を $O(N)$ 時間で解くことができました。

D: Isomorphism Freak

木に対し、その直径で同じ頂点を二度通らない最長のパスの長さ (パス上の辺の本数) を表します。また、直径を構成するパス上で最も遠い頂点までの距離が最小になる頂点を木の中心と呼びます。

最も遠い頂点までの距離が異なる二頂点は合同になりえないため、与えられる木の直径を D とすれば、互いに合同でない頂点は少なくとも $\lceil \frac{D}{2} \rceil$ 種類あります。逆に、この下界を達成できることを以下で示します。

D が偶数の場合、与えられる木の中心は 2 個あります。下界を達成するためには、これらの頂点のうち近い方からの距離が同じ頂点たちはすべて合同にする必要があります。すなわち、そのような頂点たちの次数はすべて等しくする必要があります。逆に、この条件が満たされれば、下界は達成されることが容易に確認できます。

よってこのとき、中心のうち近い方からの距離ごとに、そのような頂点の次数の最大値を求め、すべて掛け合わせたものの 2 倍が葉の個数の最小値になります。

D が奇数の場合、中心は 1 個あります。中心を c とします。下界を達成するためには、c からの距離が同じ頂点たちをすべて合同にするか、あるいは中心の隣の頂点 v をひとつ指定し、c, v のうち近い方からの距離が同じ頂点たちをすべて合同にする必要があります。

前者の場合は、D が偶数の場合と同様に解くことができます。後者の場合は、v の候補をすべて試し、D が偶数の場合と同様の処理を行えばよいです。

よって、$O(N^2)$ 時間でこの問題を解くことができました。なお、葉の個数の最小値は 64 ビット符号付き整数の範囲に収まることが (中心からの距離 k の頂点の個数が葉の個数に与える影響を評価することにより) 証明できます。

なお、(大きな整数を $O(1)$ 時間で扱えると仮定すれば) この問題は $O(N)$ 時間で解くこともできます。

E: Sequence Growing Hard

長さ k の数列に対し、その位置 $0, 1, ..., k$ で 1 文字目の前、1, 2 文字目の間、……、k 文字目の後の位置を表すことにします。

空列に整数を挿入していくことで数列を作っていくことを考えましょう。位置 i に整数 t を挿入できる条件は、便宜上数列の末尾に整数 0 が置かれているものとして考えれば、以下のいずれかの場合です。

- 数列の $i+1$ 項目の値が t より小さい

- 数列の $i+1$ 項目が t であり、その後 t 以外で初めて現れる整数が t より小さい

さて、後者の場合は ($i+1$ 項目以降初めて t 以外の整数が現れるのが r 項目であるとすれば) 位置 $r-1$ に t を挿入するとしてもできる数列は変わらないので、前者の操作のみ行うことができるとしてよいです。逆に、前者の操作でできる数列はすべて異なるので、前者の操作を繰り返し行う時の操作列の数を数えればよいです。特に、整数 t を挿入するとき、数列の t より小さい項の個数を X とすれば、$X+1$ 通りの挿入の方法があります。

さて、$DP[k][n]$ を入力 (n,k) に対する答えとします。求めたいものは $DP[K][N]$ です。1 以上 k 以下からなる長さ n の数列が、時刻 i に長さ i になるように作られていくような操作列に対し、整数 $k+1$ を合計 l 個、それぞれ時刻 X_1 と X_1+1 の間、……、時刻 X_l と時刻 X_l+1 の間に挿入するとき、上述の考察より、$k+1$ を追加する操作の操作列は $(X_1+1)(X_2+1)...(X_l+1)$ 個存在します。時刻を後から順に見ていくことを考えれば、最初全ての n に対し $x[n][n]=DP[k][n]$ として初期化し、

- $x[n][i]$ から $x[n][i-1]$ へ、係数 1 で遷移 (時刻を前に動かすことに対応)
- $x[n][i]$ から $x[n+1][i]$ へ、係数 $i+1$ で遷移 (整数 $k+1$ を時刻 i と $i+1$ の間の時刻に挿入することに対応)

という遷移式で DP を行えば、この問題を解くことができます。計算量は $O(KN^2)$ です。

F: Simple Subsequence Problem

長さ N 以下の全ての文字列に対し、その文字列を部分列として含むような S の要素の個数を求めることを考えましょう。これが求まれば、この問題を解くのは容易です。

以下のようなグラフを考えます。

- 長さ N 以下の全ての文字列 s に対し、頂点 $v_{s,0},...,v_{s,|s|}$ を用意する。
- 全ての文字列 s と $0 \leq i \leq |s|-1$ に対し、以下のように辺を張る。
  - s の先頭 i 文字を取り出してできる文字列を s_ϕ とし、$v_{s,i}$ から $v_{s_\phi,i}$ に辺を張る
  - s の $i+1$ 文字目以降で初めて 0 が出てくるのが r_0 文字目であるとする。もし r_0 が定義されるなら、s の i 文字目以前からなる文字列と r_0 文字目以降からなる文字列 (境界を含む) をつなげてできる文字列を s_0 とし、$v_{s,i}$ から $v_{s_0,i+1}$ に辺を張る
  - s の $i+1$ 文字目以降で初めて 1 が出てくるのが r_1 文字目であるとする。もし r_1 が定義されるなら、s の i 文字目以前からなる文字列と r_1 文字目以降からなる文字列 (境界を含む) をつなげてできる文字列を s_1 とし、$v_{s,i}$ から $v_{s_1,i+1}$ に辺を張る

さて、このグラフにおいて、$v_{s,0}$ から $v_{t,|t|}$ へは、t が s の部分列のときちょうど 1 本の、そうでないとき 0 本のパスがあることが示せます。これが示されれば、このグラフの $v_{s,0}$ の形をした頂点に $s \in S$ なら 1 を、そうでないなら 0 を書き、すべての辺に係数 1 をつけて DP を行うことで、全ての t について「$s \in S$ を用いて $v_{s,0}$ とあらわせる頂点からのパスの個数」、すなわち t を部分列として含むような $s \in S$ の個数を求めることができ、よってこの問題を解くことができます。

以下、このことを示します。t が s の部分列かどうかは、t を前から見る貪欲法で判定できることを思い出しましょう。まず、$v_{a,i}$ から $v_{b,j}$ へ辺があるとき b は a の部分列なので、$v_{s,0}$ から $v_{t,|t|}$ へのパスがある場

合 t は s の部分列です。また、t が s の部分列のとき、t を前から順に見て、今見ている文字が 0 なら s_0 への、1 なら s_1 への、t の末尾に到達したなら s_ϕ への辺を辿れば、$v_{s,0}$ から $v_{t,|t|}$ へ辿り着くことができます。さらに、それ以外の辿り方では $v_{t,|t|}$ には辿り着けないことも、頂点 $v_{a,i}$ から辿り着ける頂点に対応する文字列の先頭 i 文字は a と一致することからわかります。

よって主張は示され、この問題を解くことができました。時間計算量は、グラフの頂点数が $O(2^N N)$ であり、辺数がその定数倍なので、合計で $O(2^N N)$ です。

AGC 024 Editorial

DEGwer

2018/05/20

A: Fairness

Let d be Takahashi's integer minus Nakahashi's integer. How does d change after each operation?

Suppose that initially $d = x$. The three people have integers $(A + x, A, B)$ for some A, B. After one operation, the three integers will be $(A + B, A + B + x, 2A + x)$, and $d = (A + B) - (A + B + x) = -x$.

Thus, in each operation the value of d is multiplied by -1. If the number of operations is even, the answer is $A - B$, otherwise the answer is $B - A$.

B: Backfront

Suppose that we can sort the sequence without performing operations on integers $x_1, \ldots, x_k (x_1 < x_2 < \ldots)$. Then, the following conditions must be satisfied:

- Those integers will form a consecutive interval in the sorted sequence. Thus, for each i, $x_i + 1 = x_{i+1}$ holds.
- The relative positions of those integers won't change. Thus, for each i, x_i must be to the left of x_{i+1} in the initial sequence.

On the other hand, if these conditions are satisfied, we can sort the sequence in $N - k$ steps (without moving those integers). For example, we can move integers $x_1 - 1, ..., 1$ to the beginning in this order, then move integers $x_k + 1, ..., N$ to the end in this order. Thus, we want to compute the maximum possible value of k for such $x_1, ..., x_k$.

This can be done by using a permutation Q such that $Q[P[i]] = i$. This solution works in $O(N)$ time.

C: Sequence Growing Easy

By repeating operations, we can never satisfy $X_1 > 0$ or $X_{i+1} - X_i > 1$. Thus, in case $A_1 > 0$ or $A_{i+1} - A_i > 1$ for some i, the answer is -1. From now on, we assume that $A_1 = 0$ and $A_{i+1} - A_i \leq 1$ (and in this case, it turns out that we can always achieve the goal, as we see below).

First, let's compute the lower bound on the number of operations. Suppose that $A_i = t$. In order to satisfy $X_i = t$, at some point during the operations, either $t = 0$ or $X_{i-1} = t - 1$ must be satisfied. By repeating this observation, for each $s (1 \leq s \leq t)$, at some point during the operations, $X_{i-t+s} = s$ must be satisfied.

Thus, the number of different pairs of $(i - t + s, s)$ (that can be obtained this way) gives the lower bound on the number of operations. This value can be computed by adding 1 in case $X_{i-1} + 1 = X_i$, and X_i otherwise.

On the other hand, we can always achieve the goal within the number of operation computed above, as follows:

- If $X = A$, we achieved the goal, we stop performing operations.
- Otherwise, let r be the maximum integer such that $X_r \neq A_r$. For each $i - X_r + 1, ..., r$ in this order, perform an operation $X_i = X_{i-1} + 1$.

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

D: Isomorphism Freak

Let D be the diameter of the given tree. Since two vertices on the diameter are not congruent unless they are at symmetric positions of the diameter, (and the diameter never decreases by performing operations), the colorfulness of the tree must be at least .

We can always achieve this colorfulness, and let's compute the minimum number of leaves in this case.

In case D is odd

Since $\lfloor \frac{D+1}{2} \rfloor + 1 > \lfloor \frac{D}{2} \rfloor$, in order to achieve the minimum colorfulness, we must not change the diameter by operations.

In this case, the tree has two centers. To keep the diameter, we must keep these two centers, and each newly added vertices must be within the distance of $D/2$ from the centers. Here, the distance between a vertex and the centers is defined as the distance between the vertex and the center closer to the vertex.

It's easy to see that we can achieve the minimum colorfulness if and only if (we don't change the centers and the diameter and) all vertices at the same distance from the centers have the same degree. Thus, in this case, the minimum number of leaves can be computed as follows:

- Ignore the edge between the centers and split it into two rooted trees (roots are the centers).
- For each $d (0 \leq d \leq \lfloor \frac{D}{2} \rfloor)$, let x_i be the maximum number of children of vertices at depth d.
- The minimum possible number of leaves is the product of all x_i times two (since we'll get to isomorphic rooted trees).

In case D is even

Since $\lfloor \frac{D+1}{2} \rfloor + 1 - \lfloor \frac{D}{2} \rfloor$, there are two cases: we don't change the diameter, or we increase the diameter by one by operations.

In this case, the tree has a single center. Let's call it c.

There are two cases:

- Keep c as the unique center, and keep the diameter to D.
- For some vertex v adjacent to c, make c, v two centers, and change the diameter to $D + 1$.

The former case is similar to the solution described above. The latter case can be handled by trying all possibilities for v. Thus, this solution works in $O(N^2)$ time.

Note that by using the fact that $\sum x_i \leq N$, we can prove that the answer always fit in a 64-bit integer (because we can bound the value of $\prod x_i$). Also, note that it is also possible to solve this problem in linear time (left as an exercise for readers), assuming that we can perform basic operations on big integers in constant time.

E: Sequence Growing Hard

When can we insert an integer x to the left of an integer y, and make the sequence lexicographically larger? There are two cases:

- $x > y$.
- $x = y$, and the first integer that is not x after this position is greater than x.

However, we don't need to consider the second case. In the second case, if the first integer that is not x is r, we can insert x immediately to the left of this r, and this change doesn't affect the sequence. On the other hand, if we allow only the first case, if we insert an element at different positions we always get different sequences, thus we want to count the number of ways to perform operations when we are allowed to perform operation only in the first case.

Let's convert the sequence of operations to a rooted tree in the following way.

- Start with a tree with a single node. In this tree each vertex has two values: id and label. The only vertex has $id = 0, value = 0$.
- Suppose that in the k-th (1-based) operation, we insert an integer t to the left of the integer that was inserted in the p-th operation (or $p = 0$ in case we add it to the end of the sequence). Then, create a vertex with $id = k, value = t$, and its parent is a vertex with $id = p$.

Now, what we want to compute is the number of rooted trees with the following properties:

- Each vertex has two values: id and label.
- id are assigned sequencially in the order $0, 1, \ldots$.
- The label of each vertex is between 0 and K.
- For each node, its id and its label are strictly greater than parent's id and parent's label, respectively.

Let $DP[n][x]$ be the number of rooted trees (with the properties above) with n vertices such that the label of the root is x. The answer is $DP[N + 1][0]$.

To compute this DP, notice that the parent of a vertex with $id = 1$ must be a vertex with $id = 0$. If there are k vertices in the subtree rooted at $id = 1$, there are $\sum_{y>x} DP[n-k][x] * DP[k][y] * comb(n-2, k-1)$ ways to decide a tree.

Now it's easy to make it $O(KN^2)$ using prefix sums.

F: Simple Subsequence Problem

For each string s such that $|s| \leq N$, let's compute the number of elements in S that contains s as a subsequence. After getting these values, the original problem can be easily solvable.

Our plan is to construct a DAG with the following properties:

- For each string s such that $|s| \leq N$, there is a red vertex labelled with s.
- For each string s such that $|s| \leq N$, there is a blue vertex labelled with s.
- Additionally, some black vertices may exist.
- If t is a subsequence of s, there is exactly one path from red s to blue t.
- Otherwise, there is no path from red s to blue t.

Once we construct this DAG, we can get desired values by writing 1 on each red vertex labelled with an element in S, and run a DP on the DAG.

Let's construct the DAG. First, how can we list all subsequences of a particular string s? Remember that to check if a string t is a subsequence of s, we shuold choose each character in t from s greedily from left to right.

Let $[s]$ denote the set of all subsequences of s. For example, how can we compute [000110101]?

If t is a subsequence of 000110101, there are three cases:

- t starts with 0. In this case the remaining part of t must be a subsequence of 110101. These strings can be represented by 0[00110101] (a single 0 followed by a subsequence of 00110101).
- t starts with 1. In this case the remaining part of t must be a subsequence of 10101. These strings can be represented by 1[10101].
- An empty string.

Thus, [000110101] is a disjoint union of 0[00110101], 1[10101], and [].

Now, we can see that the following construction satisfies all the properties:

- For each pair of strings (s, t) such that $|s| + |t| \leq N$, prepare a black vertex labelled with $s[t]$.
- If t contains 0 and the (0-based) position of the first occurrence of 0 in t is i, add an edge from black $s[t]$ to black $s0[t']$, here t' is the suffix of t with length $|t| - i - 1$.
- If t contains 1 and the (0-based) position of the first occurrence of 1 in t is i, add an edge from black $s[t]$ to black $s1[t']$, here t' is the suffix of t with length $|t| - i - 1$.
- If t is empty, add an edge from black $s[t]$ to black $s[]$.
- For each string s, add an edge from red s to black $[s]$, and add an edge from black $s[]$ to blue s.

The number of vertices in the DAG is $O(2^N N)$, and this solution works in $O(2^N N)$ time.