AGC 015 解説

DEGwer

2017/05/27

For International Readers: English editorial starts on page 4.

A: A +...+ B Problem

N 数の最大値は最小値以上なので、$A > B$ のとき、答えは 0 です。

それ以外で $N = 1$ のとき、最大値も最小値も、その 1 つの値に等しくなるので、$A \neq B$ なら答えは 0 で、そうでないなら答えは 1 です。

それ以外のとき、最小値と最大値以外の $N - 2$ 数は、A 以上 B 以下の範囲で自由に選ぶことができます。それらの合計は全部 A にする場合に最小、全部 B にする場合に最大になり、その間の値は全て作ることができるので、$(N-1)A + B$ 以上 $A + (N-1)B$ 以下の $(B-A)(N-2)+1$ 個の値を作ることができ、これが答えです。時間計算量は $O(1)$ です。

B: Evilator

i 階に上向きの矢印ボタンのみがあるとき、i 階より上の階へは 1 回エレベーターに乗れば到着できます。i 階より下の階には、最上階で乗り換えることで 2 回エレベーターに乗れば到着できます。

i 階に下向きの矢印ボタンのみがあるとき、i 階より下の階へは 1 回エレベーターに乗れば到着できます。i 階より上の階には、最下階で乗り換えることで 2 回エレベーターに乗れば到着できます。

ある階より上/下にある階の個数は簡単に求められるので、$O(N)$ 時間でこの問題を解くことができます。

C: Nuske vs Phantom Thnook

黒マス全体は、4 方向に隣接するマスを隣り合うとしたとき、森になっています。グリッドを特定の長方形領域に制限しても、この性質は変わりません。

さて、森の各連結成分である木は、頂点数が辺数より 1 だけ大きいという性質を持っています。すなわち、各クエリに対する答えは、その領域の 頂点数 − 辺数、すなわち黒マスの数から、黒マスが隣接する箇所の数を引いたものになります。

必要な情報は、黒マスの数、黒マスが縦方向に連続する箇所の数、黒マスが横方向に連続する箇所の数のそれぞれについて、2 次元累積和をとっておけば定数時間で求めることができます。時間計算量は $O(NM + Q)$ となります。

D: A or...or B Problem

$A = B$ のとき、答えは 1 です。そうでないとき、A と B の二進表記を上の桁から順に見ていったときに初めて異なるところが、下から r 桁目であるとします。

下 r 桁以外の値がすべて等しい整数たちの or をとっても下 r 桁以外は変わらないので、これ以降下 r 桁以外を無視し、 A も B も r 桁の整数であるとします。特に、A の最上位は 0 で、B の最上位は 1 です。

まず、or をとる操作で数が小さくなることはないので、作れる最小の整数は A です。また、A 以上 2^{r-1} 未満の整数は、すべて作ることができます。また、2^{r-1} 未満の整数たちの or では 2^{r-1} 未満の整数しか作れないため、それ以上の整数を作るためには最上位が 1 の整数を使う必要があります。

最上位が 1 の整数のみを使う場合、B を上の桁から見て、最上位以外で初めて 1 が現れるのが下から k 桁目だとすると、2^{r-1} 以上 $2^{r-1} + 2^k$ 未満の整数をすべて作ることができ、それ以外は作ることはできません。(すべての $1 \leq i \leq k-1$ に対し、$2^{r-1} + 2^i$ が使えるため)

最上位が 1 の整数と最上位が 0 の整数の両方を使う場合、$2^{r-1} + A$ 以上 2^r 未満のすべての整数を作ることができ、それ以外は作ることができません。

以上で作れる整数の区間が列挙できたので、桁数の線形時間でこの問題を解くことができます。2, 3 個目に挙げた区間は重なることがあることに注意してください。

E: Mr.Aoki incubator

まず、高橋君 i を時刻 0 に青木君にしたときに、最終的に青木君になる高橋君がどのような高橋君であるかを考えます時刻 0 での座標が X_i 以下の高橋君の中での速度の最大値を R_i、時刻 0 での座標が X_i 以上の高橋君の中での速度の最小値を L_i とすると、速度が L_i 以上 R_i 以下の高橋君全員が青木君になることが分かります。

以上の考察より、この問題は次のように言い換えることができます。

問題: 区間 $[L_i, R_i]$ が与えられる。これらの区間のうちいくつかを選び、V_i たちすべてを被覆する方法は何通りあるか。

また、L_i も R_i も X_i に対して単調増加であることがわかるので、これらの区間を L_i も R_i も単調増加になるように並べることができます。

適切に座標圧縮を行い、V_i たちを 1 から高橋君の人数までの整数としておきます。 $DP[i][j]$ を、最初の i 個の区間を使い、座標 j までを被覆するような区間の選び方の数として更新すれば、$O(N^2)$ の DP ができます。この DP は、区間が L_i も R_i も単調増加になるように並んでいるという性質を使い、配列を使いまわして累積和を適切に使うことで、$O(N)$ に改善でき、この問題を解くことができます。

F: Kenus the Ancient Greek

以下、$F[0] = 1, F[1] = 1, F[k+2] = F[k+1] + F[k]$ で定まる数列 (フィボナッチ数列) の k 項目を $F[k]$ であらわします。

2 以上の整数 k に対し、次の条件を満たす整数対 (a, b) を「レベル k の良いペア」と呼ぶことにします。

- ユークリッドの互除法の反復回数が k である
- $a, b \leq F[k+2]$

例えば、レベル 3 の良いペアは $(3,5),(4,7),(5,7),(5,3),(7,4),(7,5)$ の 6 つです。

入力で与えられる整数対を X, Y とします。$X \leq Y$ として一般性を失いません。また、$X = 1$ または $(X, Y) = (2, 2)$ のときは反復回数の最大値が 1 で、その最大値をとるペアの個数は XY であるので、それ以外の場合を考えます。簡単な帰納法により、$F[k] \leq X, F[k+1] \leq Y$ なる最大の k が、ユークリッドの互除法の反復回数の最大値となることが分かります。

さて、この最大値をとるペアは、1 ステップ互除法を進めることを考えれば、あるレベル k の良いペア (a, b) と整数 t を用いて $(a, b + ta)$ と書くことができることが分かります。よって、レベル k の良いペアをすべて列挙して、各ペアについて先の条件を満たす t がいくつあるかを割り算で求めることで、この問題を解くことができます。

さて、レベル k の良いペアはいくつあるでしょうか?実は、ちょうど $2k$ 個あることが証明でき、また、下記の証明から自然に導かれるアルゴリズムを用いて (すべての k に対して合計で) $O(k^2)$ 時間で列挙することが可能です。N 以下のフィボナッチ数の個数は $O(logN)$ 個なので、良いペアを最初にすべて列挙しておけば、$N \leq 10^{18}$(入力の整数の上限) として $O(log^2 N + QlogN)$ でこの問題を解くことができます。

以下にレベル k の良いペアがちょうど $2k$ 個あることの証明の概要を示します。フィボナッチ数の一般項の式 (が近似的に指数関数であらわせること) を用いて適切に不等式評価を行うことで、各良いペアに対し、互除法を走らせたときに商が 2 以上となるような反復は高々 1 回で、さらにそのときの商は 2 であることが分かります。よって、$a < b$ なるレベル k の良いペア (a, b) は、いつ商を 2 にするか (あるいは一度も商を 2 にしないか) で表され、ちょうど k 個です (最初の反復の商は 1 であることに注意してください)。

また、同じく帰納法を用いて、レベル k の良いペア (a, b) のうち $a < b$ であるものは全て $F[k] \leq a \leq F[k+1], F[k+1] \leq b \leq F[k+2]$ を満たすことなども分かり、実装の補助となります。

AGC 015 Editorial

DEGwer

2017/05/27

For International Readers: English editorial starts on page 7.

A: A +...+ B Problem

- If $N = 1$, the only number must be the maximum and the minimum. If $A = B$, the answer is 1, otherwise the answer is 0.
- If $N > 1$,
  - If $A > B$, the answer is obviously 0.
  - Otherwise, the maximum must be B, the minimum must be A, and you can arbitrarily choose the remaining $N - 2$ numbers between A and B. The sum of those $N - 2$ numbers is between $(N - 2)A$ and $(N - 2)B$, so the answer is $(B - A)(B - 2) + 1$.

B: Evilator

- If the direction on the i-th floor is 'U', for each $j > i$, you can reach the j-th floor in one step. For each $j < i$, you can reach the j-th floor in two steps: first go to the topmost floor and transfer there.
- If the direction on the i-th floor is 'D', for each $j < i$, you can reach the j-th floor in one step. For each $j > i$, you can reach the j-th floor in two steps: first go to the bottommost floor and transfer there.

Thus, for each i, you can compute the sum of distances from i to j in constant time. Overall this solution works in $O(N)$.

C: Nuske vs Phantom Thnook

Let's construct a graph. The vertices correspond to the blue cells, and when two blue cells are adjacent, we add an edge between them. From the constraints, this graph must be a forest. Even if we construct graphs for subrectangles, this property still holds.

It is well-known that in a tree, the number of vertices minus the number of edges is always 1. When a forest has k connected components, each connected component is a tree and the number of vertices minus the number of edges in the forest is k. Thus, we can compute the number of connected components in a forest as the number of vertices minus the number of edges.

Therefore, for each query, we can compute the answer as follows:

(The number of blue cells in the rectangle)
- (The number of horizontally adjacent pairs of two blue cells in the rectangle)
- (The number of vertically adjacent pairs of two blue cells in the rectangle)

Each term in this formula can be computed in $O(1)$ if we pre-compute 2-dimensional sums. Overall it works in $O(NM + Q)$.

D: A or...or B Problem

If $A = B$, the answer is obviously 1. We assume that $A < B$.

Let r be the largest integer such that the r-th (0-based) bit of A and the r-th bit of B are different. By the definition of r, the $(r+1)$-th bit or higher bits are the same for all integers between A and B. Thus, we can ignore them, and we can assume that $0 \leq A < 2^r$ and $2^r \leq B < 2^{r+1}$.

Let $T = 2^r$, $X = \{A, A+1, \ldots, T-1\}$, and $Y = \{T, T+1, \ldots, B\}$.

- If we use only integers from the set X, we can construct integers between A and $T-1$, inclusive (choose one element). We can only construct these numbers (The minimum possible OR is A, and the OR of all elements is $T-1$).
- If we use only integers from the set Y, we can construct all integers between T and the OR of all integers in Y. Let k be the largest integer such that $2^r + 2^k$ is in the set Y. Since all of $2^r, 2^r + 2^0, \ldots, 2^r + 2^k$ are in the set Y, we can construct all integers between 2^r and $2^r + 2^{k+1} - 1$. It's easy to see that the OR of all elements in Y is $2^r + 2^{k+1} - 1$.
- If we use both integers from X and integers from Y, we can construct integers between $T + A$ and $2T - 1$, inclusive (choose T and one element from X). We can only construct these numbers (The minimum possible OR is $T + A$, and the OR of all elements is $2T - 1$).

The set of possible integers is the union of the three intervals above. Note that the second and the third intervals may overlap.

E: Mr.Aoki incubator

We call Takahashi "red point", and call Aoki "black point".

First, consider the case where we color only one point red at time 0. Let i be the index of the red point.

Suppose that $X_j < X_i$ and $V_j > V_i$. In this case, point j passes through point i at some time, so point j becomes red. Suppose that point k lies between i and j after a sufficiently long time (i.e., $V_i < V_k < V_j$. It is easy to see that point k also becomes red.

Without loss of generality, we assume that the points are sorted in the increasing order of V. For each i, we define R_i as the maximum j such that $X_j \leq X_i$. From the observation above, all points between i and R_i will be red. Similarly, we define L_i as the minimum j such that $X_j \geq X_i$. When we only choose point i at time 0, the set of red points after a sufficiently long points will be the points between L_i and R_i, inclusive.

We can compute the arrays L and R in linear time. These arrays have a special property: both L and R are monotonously increasing.

Therefore, we can restate the problem as follows:

You are given N intervals $[L_1, R_1], \ldots, [L_N, R_N]$, such that $L_1 \leq L_2 \leq \ldots$ and $R_1 \leq R_2 \leq \ldots$. Each interval is a subinterval of I. In how many ways can you choose a subset of those N intervals such that the union of chosen intervals becomes I?

Construct a DAG with $N + 2$ vertices. The vertices are numbered 1 through N, and there are two special vertices S and T.

- When L_i is the leftmost element of I, we add an edge from S to i.
- When R_i is the rightmost element of I, we add an edge from i to T.
- When $i < j$ and $R_i \geq L_j - 1$, we add an edge from i to j.

The answer is the number of paths from S to T in this DAG. It can be computed in $O(N)$ DP and prefix sums.

F: Kenus the Ancient Greek

We denote the Euclidean step of a pair (a, b) as $f(a, b)$.

When are we interested in a pair (a, b)? If there exists a pair (a', b') such that $a' \leq a, b' \leq b, f(a', b') > f(a, b)$, the pair (a, b) never become the pair with the maximum steps for any query. We call a pair (a, b) *good*, if such (a', b') doesn't exist.

An example of good pair is Fibonacci numbers. Let $F_0 = 1, F_1 = 1$ and $F_i = F_{i-1} + F_{i-2}$ for bigger i. We can see that $f(F_k, F_{k+1}) = k$ for positive k. Also, it's easy to see that this is the "minimum" pair with k Euclidean steps: that is, if $f(x, y) = k$ and $x < y$, $x \geq F_k$ and $y \geq F_{k+1}$ hold.

Ideally we want to generate all good pairs, but it turns out there are too many good pairs. For example, $(2, 2017)$ is a good pair. Most of good pairs are very "thin", and the situation becomes better after one Euclidean step. We can prove that after one step, a good pair becomes an *excellent pair*, defined as follows:

A pair of integers (a, b) is called an *excellent pair of level k* if:

- $f(a, b) = k$
- $a, b \leq F_{k+2}$

For example, there are six excellent pairs of level 3: $(3, 5), (4, 7), (5, 7), (5, 3), (7, 4), (7, 5)$.

Here is a proof: let $(x, px + y)$ be a good pair and $f(x, px + y) = k + 1$. After one step, it becomes (y, x), and $f(y, x) = k$. If this is not an excellent pair, $x > F_{k+2}$. However, we know that $f(F_{k+2}, F_{k+3}) = k + 2$, and since $F_{k+2} < x$ and $F_{k+3} = F_{k+2} + F_{k+1} < px + y$, this contradicts that the pair $(x, px + y)$ is good. Here we used $y \geq F_k$ - this comes from $f(y, x) = k$ and the fact that Fibonacci numbers are the optimal. Thus, we prove that after one step, a good pair becomes an excellent pair.

Now, the only remaining thing to do is to generate all excellent pairs. It turns out that there are exactly $2k$ excellent pairs of level k for each k, but we don't need this fact to solve the problem. Since excellent pair is also an excellent pair, after one step excellent pair becomes an excellent pair, and we can generate the list of all excellent pairs of level $k + 1$ from the list of all excellent pairs of level k. This way, all excellent pairs can be generated in $O(log^2 MAX)$ time.

To handle a query (X, Y), we first compute the maximum Euclidean steps (call it k). Then, we iterate through all $2(k - 1)$ excellent pairs of level $k - 1$, and count the number of good pairs within the rectangle (X, Y) such that after one step, it becomes the desired excellent pair. Be careful with the special case where $k = 1$.

This solution works in $O(log^2 MAX)$ pre-computation and $O(logMAX)$ per query.