TOKI Regular Open Contest #15 Editorial

(English version is available on page 8.)

Penulis soal

Judul Soal		**Author**	**Editorialis**
Div 2A	Tzaph dan Tipe Barisan	**maximath_1**	**prabowo**
Div 2B	Tzaph dan Titik Lattice	**maximath_1**	**prabowo**
Div 1A/2C	Tzaph dan FPB KPK	**maximath_1**	**maximath_1**
Div 1B/2D	Tzaph dan Grid XOR	**prabowo**	**maximath_1**
Div 1C	Tzaph dan Barisan Bagus	**maximath_1**	**maximath_1**
Div 1D/2E	Tzaph dan Reaksi Kimia	**maximath_1**	**prabowo**
Div 1E/2F	Tzaph dan Garis Bilangan	**maximath_1**	**maximath_1**
Div 1F	Tzaph dan Permainan Papan	**maximath_1**	**maximath_1**

Div 2A. Tzaph dan Tipe Barisan

Untuk kasus EQUAL, salah satu caranya adalah dengan membandingkan semua elemen dengan elemen pertama menggunakan pengulangan (`for`-loop)
Untuk kasus NONINCREASING ataupun NONDECREASING, juga dapat dicek dengan menggunakan pengulangan dari indeks 2 sampai N, yang membandingkan elemen A_{i-1} dengan A_i.
Apabila semua kasus di atas tidak terpenuhi, maka jawabannya adalah NONE.

Source Code
Kompleksitas waktu: O(N)

Div 2B. Tzaph dan Titik Lattice

Kita dapat mencoba mengiterasikan semua titik-titik yang berada di rentang [-N, N] × [-N, N]. Untuk setiap titik tersebut, hitung nilai $x^2 + y^2$-nya, dan hitung banyaknya nilai-nilai hipotenusa

tersebut pada sebuah array berukuran N^2. Kemudian, kita cukup mencari pada array, sebuah indeks yang bernilai r^2 dan isinya adalah 12.

Source Code
Kompleksitas waktu: O(N^2)

Div 1A/2C. Tzaph dan FPB KPK

Misalkan ada dua bilangan bulat positif A dan B, dengan $A = kx$, $B = ky$, dan $FPB(x, y) = 1$. Maka $FPB(A, B) = k$ dan $KPK(A, B) = kxy$.

Perhatikan bahwa $A * B = FPB(A, B) * KPK(A, B)$.

Oleh karena itu, soal ini dapat direduksi menjadi "Carilah A dan B sehingga $FPB(A, B) = 1$ dan $KPK(A, B) = KPK_{input} / FPB_{input}$", lalu A dan B akhir dikalikan dengan FPB_{input}.

Kita bisa selesaikan soal tereduksi dengan cara mencoba semua bilangan X yang kurang dari atau sama dengan $sqrt(KPK(A, B))$ sehingga memenuhi $FPB(X, KPK/X) = 1$. Ambilah hasil $X + KPK/X$ terkecil. Nilai A dan B merupakan $X \; dan \; KPK/X$.

Perhatikan pula dalam definisi ini, KPK habis dibagi oleh FPB. Maka jika KPK yang diberikan tidak habis dibagi oleh FPB, maka tidak ada solusi.

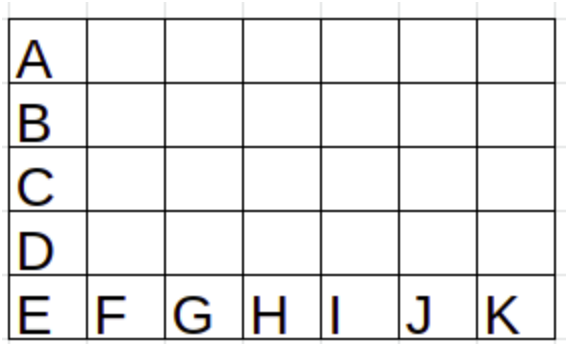
Source code
Kompleksitas waktu: O(sqrt(N))

Div 1B/2D. Tzaph dan Grid XOR

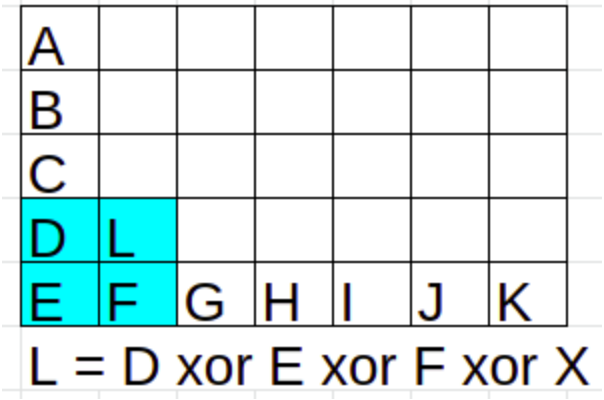
Jawabannya adalah $(K + 1)^{N + M - 1}$.

Konstruksi dari solusi ini adalah sebagai berikut:

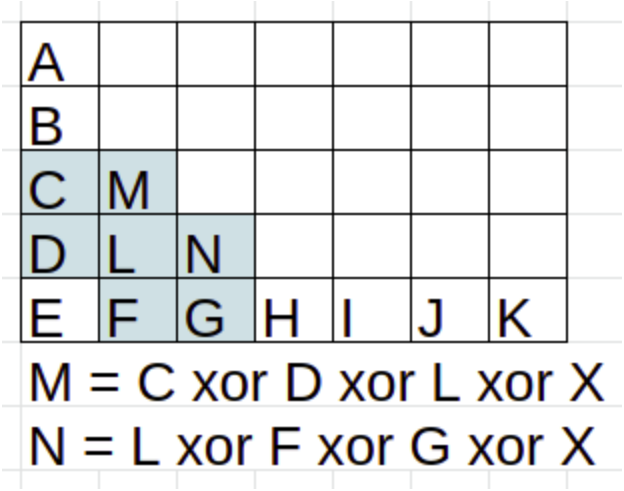
- Isilah petak pada kolom paling kiri dan baris paling bawah dengan sembarang angka. Terdapat N + M - 1 petak dan K + 1 pilihan angka per petak. Banyaknya kemungkinan adalah $(K + 1)^{N + M - 1}$



- Tinjau subgrid 2×2 di ujung kiri bawah, dengan 3 dari 4 petak pada subgrid itu telah terisi. Maka petak terakhir memiliki solusi unik yaitu bitwise xor dari 3 bilangan pada petak yang sudah terisi dan X. Hasil petak ini juga selalu valid karena K berbentuk $2^a - 1$. Banyaknya kemungkinan adalah 1.



- Setelah petak keempat itu terisi, tinjau subgrid 2x2 di atas dan di kanan subgrid 2x2 awal. Kondisi subgrid 2×2 itu sama persis dengan yang awal, yaitu 3 dari 4 petak pada subgrid itu telah terisi. Banyaknya kemungkinan adalah 1.



- Ulangi proses ini hingga semua petak terisi.

Source code

Kompleksitas waktu: O(N + M) atau O(log(N + M)) menggunakan binary exponentiation

---

Div 1C. Tzaph dan Barisan Bagus

Kita buat dua *array* prev[X] dan next[X], dengan prev[X] merupakan indeks terbesar yang lebih kecil dari X dan A[prev[X]] = A[X], dan next[X] merupakan indeks terkecil yang lebih besar dari X dan A[next[X]] = A[X]. Jika X merupakan indeks paling kecil yang mempunyai value A[X], maka prev[X] = 0. Jika X merupakan indeks paling besar yang mempunyai value A[X], maka next[X] = N + 1.
Sekarang, mari kita tinjau sebuah *subarray* di rentang [L, R]. *Subarray* tersebut bagus jika terdapat suatu index X, dengan $L \le X \le R$, sehingga prev[X] < L dan R < next[X]. Perhatikan pula bahwa semua *subarray* pada rentang [A, B] dengan $L \le A \le X \le B \le R$ juga merupakan *subarray* bagus. Maka, kita cukup meninjau *subarray* di rentang [L, X - 1] dan [X + 1, R]. Lakukan hal yang sama untuk kedua interval itu hingga interval berupa [L, R] dengan $L \ge R$. Jika terdapat suatu rentang [A, B] yang tidak terdapat X yang memenuhi prev[X] < A dan B < next[X], maka *array* tersebut tidak bagus.

Kompleksitas kita sekarang adalah $O(N^2)$ untuk kasus terburuk jika tiap kali kita tinjau sebuah rentang [L, R], kita iterasikan X dari L sampai R satu per satu. Untuk mencapai kompleksitas yang lebih rendah, kita iterasikan X secara zig-zag, mulai dari ujung kiri, lalu ujung kanan, dan seterusnya. Secara formal, kita iterasikan X dalam urutan {L, R, L + 1, R - 1, L + 2, R - 2, …}. Dengan ini, kompleksitas kita berjalan dalam kasus terburuk O(N log N).

Source code
Kompleksitas waktu: O(N log N)

Div 1D/2E. Tzaph dan Reaksi Kimia

Misalkan dp[u] adalah gelas kimia minimum yang diperlukan untuk membuat bahan kimia u. Apabila bahan kimia sudah tersedia dari awal, maka dp[u] = 1.

Misal terdapat suatu reaksi $X + Y \rightarrow Z$, dan kita sudah mengetahui nilai dp[X] dan dp[Y]. Untuk membuat Z, terdapat dua kemungkinan:

- Membuat X terlebih dahulu menggunakan dp[X] buah gelas kimia, kemudian menggunakan dp[X] - 1 gelas yang sekarang kosong untuk membuat bahan kimia Y. Jika dp[Y] ternyata lebih besar dari dp[X] - 1, maka kita memerlukan dp[Y] - dp[X] + 1 gelas kimia tambahan, dengan total dp[Y] + 1 gelas kimia. Untuk kasus ini, diperlukan max(dp[X], dp[Y] + 1)
- Sama seperti kemungkinan sebelumnya, tetapi kita membuat Y terlebih dahulu, sehingga memerlukan max(dp[X] + 1, dp[Y]) gelas kimia.

Ambil minimum dari kedua kasus, sehingga secara keseluruhan, untuk membuat bahan kimia Z dari reaksi X + Y → Z, memerlukan min(max(dp[X], dp[Y] + 1), max(dp[X] + 1, dp[Y])) gelas kimia.

Tinjau semua reaksi kimia $X_k + Y_k \rightarrow Z$ yang menghasilkan bahan kimia Z, maka kita memerlukan $\min_k(\max(dp[X_k], dp[Y_k] + 1), \max(dp[X_k] + 1, dp[Y_k]))$ gelas kimia. Dengan formula ini, kita bisa menghitung nilai-nilai dp secara *topological*, dengan bahan-bahan kimia yang ada dihitung terlebih dahulu. Jika bahan kimia tujuan T tidak dapat dicapai, maka jawabannya adalah -1.

Source code
Kompleksitas waktu: O(N + M)

Div 1E/2F. Tzaph dan Garis Bilangan

Lemma: Untuk N > 1000, pasti terdapat solusi untuk semua 1 ≤ M ≤ N.
Bukti: Kita akan membagi kasus menjadi M ≤ 500 dan M > 500.

Untuk M ≤ 500, kita akan mencari X^2 terkecil sehingga X^2 lebih besar atau sama dengan dari M dan memiliki modulo 2 yang sama. X^2 bisa dicapai dengan 2X - 1 kali langkah ke kanan. Untuk langkah-langkah selanjutnya, kita akan lakukan langkah kanan lalu kiri ("RL"), sehingga kita sampai di titik ke X^2 - 2. Lakukan ini berulang-ulang sampai titik M.
Kita harus cek apakah kita masih di titik yang valid setiap kali kita bergerak. Jelas bahwa setiap kali kita melangkah ke kiri, titik kita tetap valid (yaitu merupakan bilangan positif). Kita akan membuktikan bahwa setiap kali kita melangkah ke kanan, titik kita juga tetap valid, yaitu tidak melebihi titik 1000.
Ketika kita melangkah ke kanan, titik tersebut kita bisa ekspresikan dengan $X^2 + 2X + (2k - 1)$ dengan k adalah berapa kali kita melakukan langkah "RL". Di sini, 1 ≤ k ≤ 2X - 3 karena setelah 2X - 2 kali "RL", kita sampai pada titik $X^2 - 4X + 4$, yang sebaiknya kita menggunakan $(X - 2)^2$ daripada X^2.
Polinom tersebut maksimum jika k maksimum, yaitu k = 2X - 3. X yang memenuhi $X^2 + 2X + (2(2X - 3) - 1) \leq 1000$ adalah X ≤ 28. Karena $X^2 = 784 > 500$ (genap) dan $(X - 1)^2 = 729 > 500$ (ganjil), telah dibuktikan bahwa terdapat solusi untuk M ≤ 500.

Bukti yang mirip bisa dibuat untuk M > 500. Kita mencari X^2 terbesar yang kurang dari atau sama dengan M dan memiliki modulo 2 yang sama. Setelah menempuh 2X - 1 langkah ke kanan untuk mencapai X^2, tiap kali kita melangkah ke kiri lalu ke kanan ("LR"), titik kita berada ada 2 lebih besar dari titik kita sebelumnya. Setiap kali kita melangkah ke kiri, titik kita berada bisa diekspresikan dengan $X^2 - 2X - (2k - 1)$ dengan k adalah berapa kali kita melakukan langkah "LR" dan 1 ≤ k ≤ 2X + 1. Jika k = 2X + 2, titik akhir setelah melakukan 2X + 2 kali "LR" adalah $X^2 + 4X + 4$, dengan kita sebaiknya menggunakan $(X + 2)^2$ daripada X^2. Menggunakan k

maksimum yaitu 2X + 1, X^2 - 2X - (2(2X + 1) - 1) > 0 untuk X ≥ 7. Karena X^2 = 49 < 500 (ganjil) dan $(X + 1)^2$ = 64 < 500 (genap), telah dibuktikan bahwa terdapat solusi M > 500.

Menggabungkan kedua bukti diatas, terbukti bahwa pasti terdapat solusi untuk setiap 1 ≤ M ≤ N jika N > 1000. ■

Setelah membuktikan lemma ini, solusi akan dibagi menjadi dua kasus: M ≤ 1000 dan M > 1000.

Untuk M ≤ 1000, kita bisa lakukan BFS dengan state [posisi sekarang][sudah melakukan berapa langkah]. BFS ini bisa dilakukan dalam $O(N^2)$. Karena N bisa sampai 1000000, kita cukup mengambil N = min(N, 1001) karena sudah terbukti bahwa N > 1000 bisa untuk semua M. Cara mendapatkan konfigurasi jawaban menggunakan backtracking pada visited array.

Untuk M > 1000, kita menggunakan konfigurasi pada bukti kedua (M > 500), yaitu mencari X^2 terkecil yang kurang dari atau sama dengan M dan memiliki modulo 2 yang sama. Konfigurasi akhir adalah:

- Melakukan “R” 2X - 1 kali
- Melakukan “LR” sebanyak $(M - X^2) / 2$ kali

Tantangan: Buktikan lemma yang lebih kuat, yaitu “Untuk N > 48, pasti terdapat solusi untuk semua 1 ≤ M ≤ N.”!

Source code
Kompleksitas waktu: $O(\min(N, 1001)^2)$ untuk M ≤ 1000, O(sqrt(M)) untuk M > 1000

Div 1F. Tzaph dan Permainan Papan

Mari kita definisikan sel [N - D, N - 1] sebagai daerah pantulan.
Jika pemain berada di daerah pantulan ini, maka dalam giliran selanjutnya, pemain tersebut antara mencapai sel N dan menyelesaikan permainan atau tetap berada dalam daerah pantulan tersebut.

Kita simulasikan permainan sebanyak N - D giliran. Setelah N - D giliran, dapat dipastikan bahwa kedua pemain berada di sel lebih besar atau sama dengan N - D.

Misalkan pA[k][i] adalah peluang pemain pertama berada di sel i setelah k giliran untuk 1 ≤ i < N dan 1 ≤ k ≤ N - D. Misalkan pAD[k] adalah peluang pemain pertama berada di daerah pantulan setelah k giliran. Secara formal, pAD[k] = pA[k][N - D] + pA[k][N - D + 1] + … + pA[k][N - 1]. Misalkan juga pB[k][i] dan pBD[k] untuk pemain kedua dengan definisi yang sama. Untuk k = 0, pA[0][sA] = 1, pB[0][sB] = 1, dan 0 selain kedua posisi tersebut.

Misalkan pula Q[k] adalah peluang pemain kedua tidak menang setelah k giliran. Untuk k = 0, Q[0] = 1.

Untuk setiap giliran, pAD[k] * D^{-1} * Q[k - 1] menyatakan peluang pemain pertama menang pada giliran ini. Tambahkan nilai ini ke jawaban. pBD[k] * D^{-1} menyatakan peluang kedua menang pada giliran ini, maka Q[k] = Q[k - 1] - pBD[k] * D^{-1}.

Selanjutnya, kita update isi pA[k + 1][i], pAD[k + 1], pB[k + 1][i], dan pB[k + 1].
Peluang pemain yang berada di daerah pantulan tetap berada di daerah pantulan adalah 1 - D^{-1}, maka pAD[k + 1] = (1 - D^{-1}) * pAD[k].
Lalu untuk 1 ≤ i < N - D, pA[k+1][i] = pA[k][i-d-1] + pA[k][i-d] + ... + pA[k][i-1] dan begitu pula untuk pB. Untuk i ≥ N - D, pAD[k+1] += pA[k][N-D-1] + pA[k][N-D-2] + … + pA[k][i-D-1] dan begitu pula untuk pB. Semua operasi penjumlahan dapat dioptimasikan dengan prefix sum.

Setelah N - D giliran, kedua pemain berada di daerah pantulan. Misalkan A adalah peluang pemain pertama menang dan sekarang merupakan giliran pemain pertama dan B adalah peluang pemain pertama menang dan sekarang merupakan giliran pemain kedua. Maka berlaku A = D^{-1} + (1 - D^{-1}) * B. Karena A + B = 1, maka didapat $A = \frac{D}{2D-1}$. Jawaban akhir ditambah dengan pAD[N - D] * $\frac{D}{2D-1}$ * Q[N - D].

Kompleksitas waktu: $O(N^2)$

TOKI Regular Open Contest #15 Editorial

Problem Authors

Problem Title		Author	Editorialist
Div 2A	Tzaph and Sequence Types	**maximath_1**	**prabowo**
Div 2B	Tzaph and Lattice Points	**maximath_1**	**prabowo**
Div 1A/2C	Tzaph and GCD LCM	**maximath_1**	**maximath_1**
Div 1B/2D	Tzaph and XOR Grid	**prabowo**	**maximath_1**
Div 1C	Tzaph and Good Array	**maximath_1**	**maximath_1**
Div 1D/2E	Tzaph and Chemical Reactions	**maximath_1**	**prabowo**
Div 1E/2F	Tzaph and Number Line	**maximath_1**	**maximath_1**
Div 1F	Tzaph and Board Game	**maximath_1**	**maximath_1**

Div 2A. Tzaph and Sequence Types

For the EQUAL case, one way is to compare all elements against the first element using looping (`for`-loop).
For the NONINCREASING or NONDECREASING cases, can be checked using looping from the index 2 through N, which compares element A_{i-1} with A_i.
If all the above conditions were not satisfied, then the answer is NONE.

Source Code
Time Complexity: O(N)

Div 2B. Tzaph and Lattice Points

We can try to iterall all points in the range [-N, N] $\times$ [-N, N]. For every point, compute the value $x^2 + y^2$, and count the number of those hypotenuse values on an array of size N^2. Then, we simply find an index with value r^2 containing 12.

Source Code
Time Complexity: O(N^2)

Div 1A/2C. Tzaph and GCD LCM

Let there be two positive integers A dan B, where $A = kx$, $B = ky$, and $GCD(x, y) = 1$. Then $GCD(A, B) = k$ and $LCM(A, B) = kxy$.

Notice that $A * B = GCD(A, B) * LCM(A, B)$.

From this, the problem can be reduced to "Find A and B such that $GCD(A, B) = 1$ and $LCM(A, B) = LCM_{input} / GCD_{input}$", then A and B will be multiplied by GCD_{input}

We can solve this by iterating all positive integers X less than or equal to $sqrt(LCM(A, B))$ such that $GCD(X, LCM/X) = 1$. Take the result which gives the minimum $X + LCM/X$. The value of A and B will be $X \; and \; LCM/X$.

Take note that in this definition, LCM is divisible by GCD. Thus if the given LCM is not divisible by the given GCD, then no solution exists.

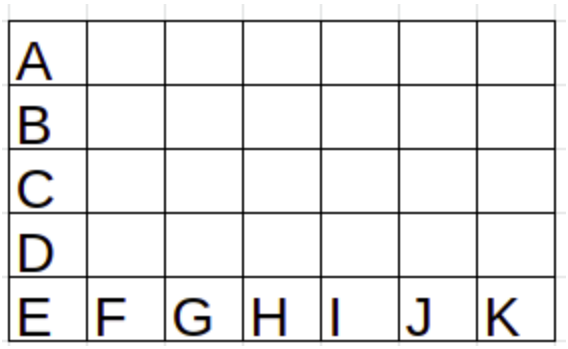
Source code
Time Complexity: O(sqrt(N))

Div 1B/2D. Tzaph and XOR Grid

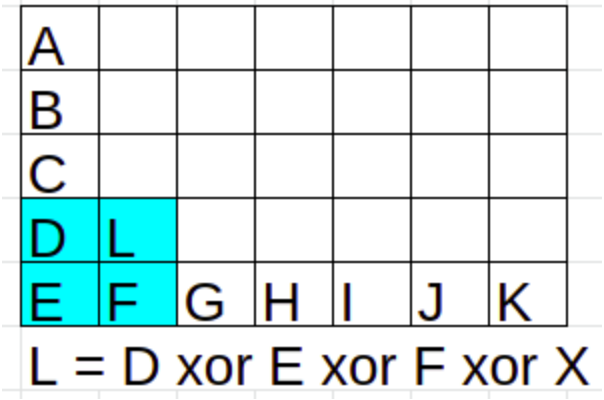
The answer is $(K + 1)^{N + M - 1}$.

The construction of this solution is of the following:

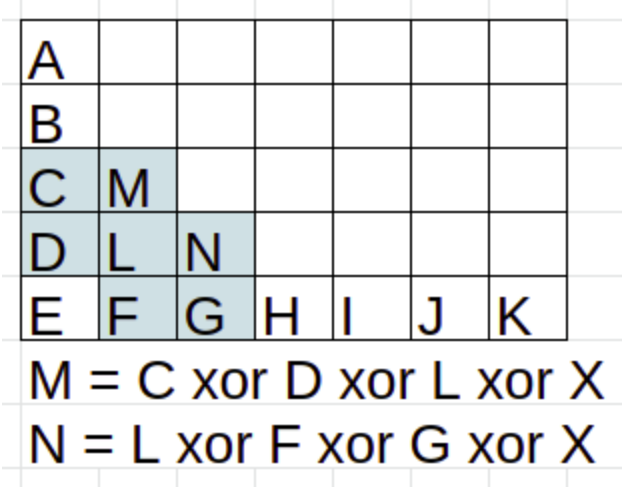
- Fill the cells of the leftmost column and the bottommost row with any number from 0 to K. There are N + M - 1 cells and K + 1 choices per cell. Thus, the number of ways of assigning is $(K + 1)^{N + M - 1}$



- Observe the 2×2 subgrid located at the bottom left corner, where 3 out of 4 of the cells are filled. Then the fourth cell has a unique solution, which is the bitwise xor of the 3 cells and X. The result is also always valid as K is in the form of $2^a - 1$. The number of ways to assign this cell is 1.



- After filling that cell, observe the two 2×2 subgrids located up and right of the initial 2x2 subgrid. The condition of these two subgrids is the same as the initial subgrid before, which is 3 out of 4 cells filled. The number of ways to fill the two new cells is 1.



- Continue the process until the grid is fully filled.

Source code
Time Complexity: O(N + M) or O(log(N + M)) using binary exponentiation

---

Div 1C. Tzaph and Good Array

We will create two arrays prev[X] and next[X], where prev[X] is the largest index less than X and A[prev[X]] = A[X] and next[X] is the smallest index larger than X and A[next[X]] = A[X]. If X is the smallest index that has the value A[X], then prev[X] = 0. If X is the largest index that has the value A[X], then next[X] = N + 1.

Let us observe a subarray with range [L, R]. The subarray is good if there is an index X where $L \le X \le R$, such that prev[X] < L and R < next[X]. Notice that if [L, R] is good, then all subarrays [A, B] with $L \le A \le X \le B \le R$ are also good subarrays. Thus, we only have to observe subarray of range [L, X - 1] and [X + 1, R]. Do the same for these new two subarrays until the subarray has the range [A, B] where $A \ge B$, where these subarrays are good. If there is a subarray [L, R] that doesn't have an index X that satisfies prev[X] < L and R < next[X], then the array is not good.

Our time complexity now is worst case $O(N^2)$ if every time we are observing a subarray [L, R], we iterate X from L to R one by one. To achieve a better complexity, we iterate X in a zig-zag , from the leftmost element, then the rightmost element, and so on. Formally, we iterate X with the order {L, R, L + 1, R - 1, L + 2, R - 2, …}.
With this optimization, the time complexity is reduced to O(N log N).

Source code
Time Complexity: O(N log N)

Div 1D/2E. Tzaph and Chemical Reactions

Let dp[u] be the minimum number of beakers needed to make chemical substance u. If the chemical substance is already available, then dp[u] = 1.

Let there be a chemical reaction X + Y → Z, and we already know the values dp[X] and dp[Y]. To make Z, there are two possibilities:

- Make X first using dp[X] beakers, then use dp[X] - 1 empty beakers to make chemical substance Y. If dp[Y] is larger than dp[X] - 1, we need additional dp[Y] - dp[X] + 1 beakers, with total dp[Y] + 1 beakers. In this case, we need max(dp[X], dp[Y] + 1)
- Same as previous, but we make Y first, hence we need max(dp[X] + 1, dp[Y]) beakers.

Take the minimum of the two cases, so as a whole, to make chemical substance Z from the chemical reaction X + Y → Z, we need min(max(dp[X], dp[Y] + 1), max(dp[X] + 1, dp[Y])) beakers.

Consider all chemical reactions $X_k + Y_k \rightarrow Z$ that produce chemical substance Z, then we need $\min_k(\max(dp[X_k], dp[Y_k] + 1), \max(dp[X_k] + 1, dp[Y_k]))$ beakers. Using this formula, we can compute the dp values in topological order, where the chemical substances that are already available are processed first. If the chemical substance T can not be reached, the answer is -1.

Source code
Time Complexity: O(N + M)

Div 1E/2F. Tzaph and Number Line

Lemma: For N > 1000, there is a solution for all 1 ≤ M ≤ N.
Proof: We will divide cases to M ≤ 500 and M > 500.

For M ≤ 500, we will find the smallest X^2 such that X^2 is greater than or equal to M and they have the same value in modulo 2. We can reach point X^2 by 2X - 1 steps to the right. For the next steps, we will do a step to the right then a step to the left. This brings us to point X^2 - 2. Repeat this until we reach point M.
We must check if our every step is valid. It is obvious that whenever we move to the left, our final point is valid which is a positive integer. We will prove that for every step to the right, our final point is also valid, which is less than or equal to 1000.

Whenever we move to the right, our point can be expressed as $X^2 + 2X + (2k - 1)$ where k is the number of times we have done the step "RL" and 1 ≤ k ≤ 2X - 3 because after 2X - 2 times moving "RL", we will reach point $X^2 - 4X + 4$, which we should've used $(X - 2)^2$ instead of X^2.
$X^2 + 2X + (2k - 1)$ is maximum when k is maximum, which is when k = 2X - 3. The Xs that satisfy $X^2 + 2X + (2(2X - 3) - 3) \leq 1000$ is X ≤ 28. Since $X^2 = 784 > 500$ (even case) and $(X - 1)^2 = 729 > 500$ (odd case), we have proven a configuration for M ≤ 500.

A similar proof can be made for M > 500. We will find the largest X^2 such that X^2 is less than or equal to M and has the same value in modulo 2. We can reach point X^2 after 2X - 1 right steps. Doing a step to the left then to the right ("LR") will bring us to point $X^2 + 2$. Do this repeatedly until we reach point M. It is obvious that every move to the right is always valid. When we move to the left, our point can be expressed as $X^2 - 2X - (2k - 1)$ where k is the number of times we have done the step "LR" and 1 ≤ k ≤ 2X + 1 as if k = 2X + 2, our final point will be $X^2 + 4X + 4$, in which we should have used $(X + 2)^2$ instead of X^2. $X^2 - 2X - (2k - 1)$ is minimum when k is maximum, which is k = 2X + 1. The Xs that satisfy $X^2 - 2X - (2(2X + 1) - 1) > 0$ is X ≥ 7. Since $X^2 = 49 < 500$ (odd case) and $(X + 1)^2 = 64 < 500$ (even case), we have proven a configuration for M > 500.

Combining the two proofs above proves that there is always a solution for every 1 ≤ M ≤ N when N > 1000. ■

After proving the lemma, the solution will be divided into two cases: $M \le 1000$ and $M > 1000$.

For $M \le 1000$, we can do a BFS with states [current position][how many steps]. The BFS has a complexity of $O(N^2)$. Since N can reach a value of 1000000, we can take $N = \min(N, 1001)$ since we have proven that for $N > 1000$, every M is possible. We can obtain the configuration of the solution by backtracking on the visited array.

For $M > 1000$, we can use the configuration on the second proof ($M > 500$), which is finding the largest X^2 that is less than or equal to M and has the same value in modulo 2. The configuration would be:

- Do step "R" $2X - 1$ times
- Do step "LR" $(M - X^2) / 2$ times

Challenge: Prove a stronger lemma, which is "For $N > 48$, there is a solution for all $1 \le M \le N$"!

Source code
Time Complexity: $O(\min(N, 1001)^2)$ for $M \le 1000$, $O(\sqrt{M})$ for $M > 1000$

Div 1F. Tzaph and Board Game

Define the cells $[N - D, N - 1]$ as the bouncing area.
If a player is in the bouncing area, then in the next move the player will either reach cell N and finish the game or still stay in the bouncing area.

We will do the simulation of the game for $N - D$ turns. After $N - D$ turns, it can be guaranteed that the two players are located at cells greater than or equal to $N - D$.

Let $pA[k][i]$ be the probability of the first player being in cell i after k turns, where $1 \le i < N$ and $1 \le k \le N - D$. Let $pAD[k]$ be the probability of the first player being in the bouncing area after k turns. Formally, $pAD[k] = pA[k][N - D] + pA[k][N - D + 1] + \ldots + pA[k][N - 1]$. Let $pB[k][i]$ and $pBD[i]$ have the similar definition but for the second player. For $k = 0$, $pA[0][sA] = 1$, $pB[0][sB] = 1$, and 0 other than the two positions.

Let $Q[k]$ be the probability of the second player not winning after k turns. For $k = 0$, $Q[0] = 1$.

For every turn, $pAD[k] * D^{-1} * Q[k - 1]$ is the probability of the first player winning in this turn. Add this value to the answer. $pBD[k] * D^{-1}$ is the probability of the second player winning this turn, thus $Q[k] = Q[k - 1] - pBD[k] * D^{-1}$.

Then, we will update the values of $pA[k + 1][i]$, $pAD[k + 1]$, $pB[k + 1][i]$, and $pBD[k + 1]$.

The probability of a player in the bouncing area to stay in the bouncing area is $1 - D^{-1}$, thus $pAD[k + 1] = (1 - D^{-1}) * pAD[k]$.
For $1 \le i < N - D$, $pA[k+1][i] = pA[k][i-d-1] + pA[k][i-d] + ... + pA[k][i-1]$ and the same thing for pB.
For $i \ge N - D$, $pAD[k+1] \mathrel{+}= pA[k][N-D-1] + pA[k][N-D-2] + \ldots + pA[k][i-D-1]$ and the same thing for pB. The sum operations can be optimised using prefix sum.

After N - D turns, the two players are in the bouncing area. Let A be the probability of the first player winning and is currently the first player's turn, and B be the probability of the first player winning and is currently the second player's turn. Then $A = D^{-1} + (1 - D^{-1}) * B$. Since A + B = 1,

we will get $A = \frac{D}{2D-1}$. Our final answer is added by $pAD[N - D] * \frac{D}{2D-1} * Q[N - D]$.
Time Complexity: $O(N^2)$