Concurso para principiantes de AtCoder 100

Editorial

por E869120, cuadrado1001

16 de junio de 2018

Para lectores internacionales: la editorial en inglés comienza en la página 6.

revisión general

pregunta	Número de respuestas correctas	Número de envíos	Grado de dificultad*	Primera AC	Ejecución más rápida	Código más corto
¡Un feliz cumpleaños!	2321	3656	150	0:40	0ms	19 octetos
B: Los números favoritos de Ringo	1806	6710	280	1:40	0ms	22 bytes
C: *3 o /2	1887	2433	270	1:02	1 ms	42 bytes
D: Pastelería ABC	574	1636	450	5:48	1 ms	138 bytes

Este es el resultado del concurso. Además, el nivel de dificultad es el nivel de dificultad esperado convertido por la puntuación durante el concurso.

Esta vez, dado que es el ABC número 100, escribí una reseña general.

Gracias a todos por participar en el concurso.

El concurso de este año es el 100 aniversario de ABC, por lo que puedes comer mucho en cumpleaños y otras ocasiones. Hubo muchos problemas con el tema de la "torta".

Por ejemplo, el problema A. Estaba previsto que fuera un cumpleaños, pero en realidad era un pastel porque era el 100 aniversario de ABC. Había intenciones ocultas. ¿Lo leíste?

Además, esta vez, aparecieron muchos "múltiplos de 100" en las restricciones. Por ejemplo, la restricción para el problema D es ≤ 10000 y ≤ 1010 . Lo configuré intencionalmente para incluir muchos 10 y 100. Por cierto, 100 en "Número de veces dividido por 100" en el Problema B también se deriva de ABC100.

Por cierto, gracias por leer la primera página del comentario. Disfruta del cuerpo principal del comentario. Por favor.

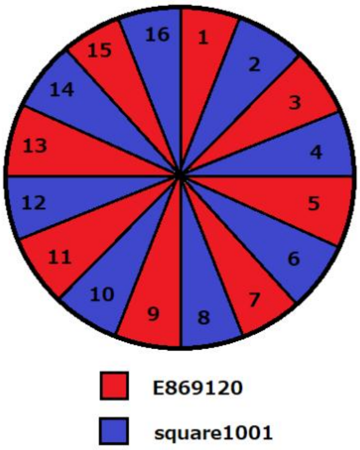
La explicación en sí comienza en la segunda página.

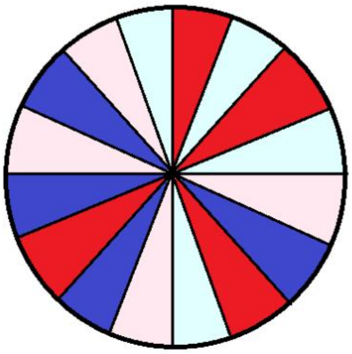
Pregunta A: ¡Feliz cumpleaños!

La respuesta es “¡Sí!” si ≤ 8 y ≤ 8, de lo contrario la respuesta es “:(“.

Primero, los pasteles tomados por la misma persona no pueden estar uno al lado del otro, por lo que no puedes poner más de 16 ÷ 2 = 8 pasteles. sí. En otras palabras, puedes demostrar que no puedes comer tantos pasteles como quieras cuando uno de ellos tiene 9 o más.

Además, cuando ≤ 8, ≤ 8, E869120 en el sentido de las agujas del reloj 1, 3, 5, 7, 9, 11, 13, 15, square1001 en el sentido de las agujas del reloj 2, 4, 6, 8 , 10, 12, 14, 16, puede comer tantos pasteles como quieras mientras cumples las condiciones de cómo obtenerlos.





[Example: A = 4, B = 4]

Una vez que sepa esto, todo lo que tiene que hacer es ramificar condicionalmente usando una declaración if.

Código de muestra: https://beta.atcoder.jp/contests/abc100/submissions/2663851

BNúmeros favoritos de Ringo

Hay muchas soluciones a este problema. A continuación se describen dos soluciones representativas.

Política 1: "Exploración completa"

Considere resolver el problema "Dado un número entero, ¿cuántas veces es divisible por 100?" vinagre. Esto se puede resolver al continuar dividiendo hasta que ya no sea divisible por 100.

Si la respuesta es = 1,2,3,… , Resolveremos el problema anterior sobre . El problema anterior es diez

Se ejecuta un minuto más rápido y tiene un valor máximo de 1010000 (cuando = 2, = 100), por lo que puede escribir un programa que pueda cumplir fácilmente con el límite de tiempo de ejecución.

Código de muestra: https://beta.atcoder.jp/contests/abc100/submissions/2669995

Estrategia 2: "Enumerar el número que cumple las condiciones"

Un número que es exactamente divisible por 100 tiene = 100 como:

, 2 , 3 , 4 , … , 99 , 101 , 102 , 103 , … , 199 , 201 , 202 , 203 , … , 299 , 301 , …

Es decir, solo los números escritos como × 100 (no es un múltiplo de 100) son exactamente veces divisibles por 100. se puede cortar Pensemos en una solución basada en esto.

Sea = (el cociente de −1 dividido por 99). Entonces la respuesta a este problema será mayor o igual a (100 + 1) x 100 y menor o igual a (100 + 99) x 100. Además, dado que hay 99 "números que son divisibles por 100 exactamente veces" por debajo de 100 × 100, es el -99 en el rango al que limitamos la respuesta anteriormente. Entonces la respuesta es (100 + − 99) × 100.

De manera más simple, la respuesta es (−) × 100 = (− $\overline{}$ −1 99) × 100.

Política 2+α: "Método trampa: usar hasta N≤100 para resolver"

Considerando la Estrategia 2, sea = 100, los primeros 100 números que son divisibles por 100 exactamente veces

dientes , 2 , 3 , 4 , … , 97 , 98 , 99 , será 101.

Por lo tanto, suponiendo ≤ 100, la respuesta es = × 100 cuando ≤ 99, y = 100 y 101 = 101 × 100 es la respuesta.

Código de muestra: https://beta.atcoder.jp/contests/abc100/submissions/2663837

Pregunta C: *3 o /2

≤ 10000, ≤ 109 , por lo que hay casos en los que el valor final se acerca a los 90000 dígitos después de muchas operaciones. Incluso si supiéramos cuál es el mejor método, la cantidad de dígitos sería demasiado grande para la simulación, por lo que tendríamos que idear algún otro método.

Primero, al realizar la operación "multiplicar por 3" varias veces, el número de operaciones "dividir por 2" aumentará o disminuirá. ¿Debemos? La respuesta es no. Eso es porque multiplicar números impares no aumenta el número de divisiones por 2, y multiplicar números enteros no disminuye el número de divisiones por 2.

Por lo tanto, una operación de "multiplicar por 3" no tiene efecto sobre el número de operaciones subsiguientes. es decir, "tres veces" se puede reemplazar con la condición "no hacer nada". Por lo tanto, cada operación se reemplaza por ``Para cada número, elija ``dividir por 2" o ``dejar como está", pero no todo.

Ahora consideremos cada operación. "Dividir por 2" reduce el número de operaciones restantes, pero Dado que el número de operaciones restantes no disminuye con "dividir por 2", consideramos minimizar el número de "dividir por 2" en cada operación, es decir, 1 "dividir por 2". Entonces, en una operación, podemos reemplazarlo con ``elija uno de los números y divídalo por 2".

El resto es Winning Run. La respuesta es ($_1$ se divide por 2) + ($_2$ se divide por 2) + … + (número de veces divisible por 2). es divisible por 2, la complejidad computacional es ( ), por lo que la complejidad computacional total es (( )).

Bono: También hay un método de cálculo con la cantidad de cálculo ( )). Podemos lograr esta complejidad encontrando el número de veces que es divisible por 2.

Código de muestra: https://beta.atcoder.jp/contests/abc100/submissions/2670458

Pregunta D: Pastelería ABC

Primero, considere el siguiente problema simple.

Hay N pasteles y se determina la belleza, el sabor y la popularidad.

Elige pasteles M.

Encuentre el valor máximo de (belleza total) + (delicia total) + (popularidad total).

Este problema es fácil de resolver. Si la belleza del pastel es , la delicia es y el grado de popularidad es , entonces el valor deseado es la suma de (+ +), por lo que la solución de tomar M pasteles en orden descendente del valor (+ +) es válido _

Sin embargo, el problema original es (valor absoluto de la belleza total) + (valor absoluto de la delicia total) + (popularidad) Era un problema de maximizar el valor absoluto de la suma).

En otras palabras, también hay una manera de hacer que la suma de algunos elementos sea negativa.

Por lo tanto, para cada elemento de belleza, delicia y popularidad, "maximice en la dirección positiva" y "en la dirección negativa". Es fácil de entender si piensas cuál elegir. Como hay 3 elementos, recorre 2 3 = 8 vías.

Por ejemplo, si desea maximizar la belleza y el gusto en la dirección positiva y maximizar la popularidad en la dirección negativa, + − Es una buena idea clasificar los pasteles en orden descendente y seleccionar M pasteles de la parte superior.

La respuesta es la mejor solución entre las 8 vías buscadas exhaustivamente. La búsqueda exhaustiva se puede simplificar utilizando operaciones bit a bit. Lo haré. La complejidad computacional es ( ) 8 = ( ).

Además, este problema se puede resolver con ( ). El algoritmo para ordenar una secuencia de longitud N y encontrar el M-ésimo valor A más grande con la peor complejidad ( ) es este enlace Después de encontrar el valor A , todo lo que tiene que hacer es calcular la suma de los valores más grandes, por lo que la complejidad del peor de los casos para este problema es ( ).

Código de muestra: https://beta.atcoder.jp/contests/abc100/submissions/2670489

Análisis/revisión general

Problema	Corrige la dificultad de las presentaciones*			Primera CA	más rápida	más corto
¡Un feliz cumpleaños!	2321	3656	150	0:40	0ms	19 octetos
B: Los números favoritos de Ringo	1806	6710	280	1:40	0ms	22 bytes
C: *3 o /2	1887	2433	270	1:02	1 ms	42 bytes
D: Pastelería ABC	574	1636	450	5:48	1 ms	138 bytes

La dificultad esperada es una puntuación en la dificultad de este problema. Es sólo una conjetura de una fórmula.

¡Gracias a todos por participar en el concurso o ver este editorial! Este El concurso es exactamente el ABC número 100, por lo que escribimos algo nuevo, "análisis / revisión".

Como este concurso es un aniversario, el tema de algunos problemas fue “torta”, que se come comúnmente en los cumpleaños (al menos en Japón).

Por ejemplo, el problema A. Se trataba de E869120 y el 16 de square1001 cumpleaños, pero había una intención oculta de celebrar este 100 aniversario de ABC. ¿O hiciste una suposición correcta?

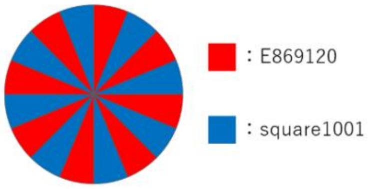
Y además, en este concurso se presentaron muchos “múltiplos de 100”. Por ejemplo, las restricciones del problema D eran ≤ 10000 = 1002 y ≤ 1010 = 1005 . Y el problema B "el número de veces que se puede dividir por 100" se deriva del ABC "100". Además, hay muchos 10 porque la raíz cuadrada de 100 es exactamente 10.

De todos modos, gracias por leer la primera página del editorial en inglés. Por favor, disfruta de la cuerpo principal de este editorial :)

El cuerpo principal de este editorial comienza en la página siguiente.

Problema A: ¡Feliz cumpleaños!

Primero, considere el caso de ≤ 8 y ≤ 8 (ambos quiere no más de 8 piezas de tortas). En este caso, si solo permite que E869120 tome piezas en el área roja y square1001 tome piezas en el área azul, cumplirá la condición (no hay dos pasteles adyacentes para la misma persona).



Por lo tanto, la respuesta para este caso es "¡Yay!".

Segundo, si una persona quiere tomar más de 8 tortas, definitivamente no satisfará la condición, por lo que la respuesta para este caso es ":(".

Si te das cuenta de esto, lo que queda es solo alguna implementación. La implementación de este problema es tan fácil como un pedazo de pastel. Puede usar la declaración if-else para bifurcar dos condiciones.

Código de muestra (C++): https://beta.atcoder.jp/contests/abc100/submissions/2663851

Problema BNúmeros favoritos de Ringo

Hay muchas soluciones posibles en este problema. En esta ocasión, explicaremos dos soluciones representativas.

Enfoque 1Buscar todo

Considere este subproblema: "Se le da un número entero . cuantas veces puede ser dividido por 100?". Esto se puede resolver mediante simulación y funciona muy rápido.

si la respuesta es , deberíamos resolver este subproblema para = 1,2,3,… , más . ya que no es de 1010000 (cuando = 2, = 100), puedes escribir un programa que sea mucho más rápido que el plazo de ejecución.

Código de muestra (C++): https://beta.atcoder.jp/contests/abc100/submissions/2669995

Enfoque 2: enfoque "lápiz y papel"

En realidad, no tienes que hacer fuerza bruta (buscar todas las respuestas posibles). En realidad, esto El problema se puede resolver con lápiz y papel (si Let , está arreglado).

= 100. El número que se puede dividir por 100 exactamente es el siguiente:

, 2 , 3 , 4 , … , 99 , 101 , 102 , 103 , … , 199 , 201 , 202 , 203 , … , 299 , 301 , …

Esto significa que solo × 100 (no es un múltiplo de 100) se puede dividir por 100 exactamente veces. Pensemos en la solución basada en esta propiedad.

Sea el cociente de − 1 dividido por 99. La respuesta de este problema estará entre (100 + 1) × 100 y (100 + 99) × 100. Además, hay 99 números como máximo 100 × 100 y se pueden dividir por 100 exactamente veces. Esto significa que la respuesta será clasificado − 99 desde el valor más pequeño que limitamos el rango de la respuesta hace un tiempo. Por tanto, la respuesta será (100 + − 99) × 100 .

Simplificando esto, la respuesta será (−) × 100 = (− −1 99) × 100 .

Enfoque 2+α "Solución astuta" usando N≤100 Let = 100 .

Al ver el enfoque 2, los primeros 100 números que se pueden dividir por 100 los tiempos exactos son , 2 , 3 , 4 , … , 97 , 98 , 99 , 101 . Por lo tanto, suponiendo ≤ 100, la respuesta será = × 100 si ≤ 99 de lo contrario 101 = 101 × 100 .

Código de muestra (C++): https://beta.atcoder.jp/contests/abc100/submissions/2663837

Problema C: *3 o /2

Veamos las características de “*3” o “/2”. Puede "multiplicar por 3” tiempo ilimitado, pero no se puede "dividir por 2”, porque no puede elegir esta operación si es un extraño número.

Entonces, la solución óptima parece ser la siguiente:

- Para cada operación, divida exactamente 1 número que es un número par y múltiplo otros numeros

De todos modos, ¿cuántas veces dividirás?

Sea ( ) “el número de veces que se puede dividir por 2”. Por ejemplo, (8) = 3, (244) = 2 y (100) = 2.

Si elige "múltiplo por 3", ( ) siempre debe permanecer sin cambios. De lo contrario (si Ud. elige "dividir en 2”), ( ) siempre debe disminuir en 1.

Finalmente, si todos ( ) se convirtieron en cero (esto significa que todos los números son impares), no puede hacer operación más.

Entonces, el número máximo de operaciones es $_1$) + ($_2$) + + ( ). ya que puedes (determinar ( ) con ( ) complejidad, la complejidad de este algoritmo es ( ).

Código de muestra: https://beta.atcoder.jp/contests/abc100/submissions/2670458

Bonificación: puede resolver el problema con complejidad (log log max). Porque, si usa la búsqueda binaria por el valor de ( ), puede determinar ( ) con complejidad (log log).

Problema D: Pastelería ABC

Primero, pensemos en este sencillo subproblema:

Hay pasteles N, que el valor de la belleza, el sabor y la popularidad.
Elegirás pasteles M.
Encuentre el valor máximo posible de (suma de belleza) + (suma de sabor) + (suma de popularidad).

Este subproblema se puede resolver fácilmente. Si la belleza del pastel es , el sabor es , y la popularidad , la respuesta será suma de (+ +) cuando sea la torta elegida.
es Por lo tanto, elegir pasteles de mayor (+ +) es la opción óptima.

Pero el problema original es maximizar (valor absoluto de la suma de la belleza) + (valor absoluto valor de la suma de sabor) + (valor absoluto de la suma de popularidad). Entonces, también hay una manera de hacer que la suma de algunos parámetros sea negativa.

Por lo tanto, puedes entenderlo fácilmente si piensas en "maximizar la fuerza hacia el logro positivo". dirección" o "maximización de la fuerza hacia la dirección negativa" para cada parámetro: belleza, sabor y popularidad. Dado que hay tres parámetros, puede aplicar fuerza bruta a $2^3 = 8$ maneras.

Por ejemplo, si desea suma de belleza y suma de sabor para maximizar la fuerza hacia positivo, y quiere suma de popularidad para maximizar la fuerza hacia negativo, elegir pasteles de mayor (+ −) es la elección óptima.

La respuesta será el valor máximo entre $2^3 = 8$ maneras. Es posible usar recursividad para la fuerza bruta, pero el uso de "fuerza bruta con operaciones de bits" facilitará la implementación. Supongamos que ordena una matriz de longitud en (log), la complejidad del tiempo será (log) × 8 = (log).

Código de muestra (C++): https://beta.atcoder.jp/contests/abc100/submissions/2670489

Bonificación: este problema se puede resolver en tiempo lineal. Primero, puede calcular el -ésimo valor más grande en tiempo lineal. El algoritmo llamado "Mediana de medianas" permite calcular la mediana en tiempo lineal y luego realizar una búsqueda binaria en la matriz, el -ésimo valor más grande se puede calcular en tiempo lineal. La suma del número del 1 al -ésimo valor más grande se calculará, simplemente, "sumando el valor mayor que el -ésimo valor más grande", además necesitamos algunos ajustes.