Final OIE 2021 Día 1

Soluciones a los problemas

XXV OIE 2021

17 de abril 2021

Autores de problemas:

- Alberto Maurel Serrano
- Blanca Huergo Muñoz
- Félix Moreno Peñarrubia

Equipo de preparación:

- Izan Beltran Ferreiro
- Javier Nistal Salas
- Jordi Guillem Rodríguez Manso
- Max Balsells i Pamies
- Miquel Ortega Sánchez-Colomer
- Óscar Balcells Obeso
- Pablo Hidalgo Palencia

Adivina el factorial

Enunciado

Debes adivinar un entero positivo $n \leq 40000$ haciendo preguntas del tipo "¿Es $n! + a$ divisible por b?" para $1 \leq a, b \leq 10^9$.

Autor: Félix Moreno Peñarrubia

Adivina el factorial

Veamos la información que se puede obtener cuando $a = 0$ (o, equivalentemente, $a = b$).

Observación

Sólo es interesante preguntar por $b = p^m$ con p primo.

Podemos generar todos los primos con la Criba de Eratóstenes y considerar las potencias de cada primo que sean menores a 10^9. Con p^m podemos decidir si n es mayor o menor que

$$(m - \left\lfloor \frac{m}{p} \right\rfloor - \left\lfloor \frac{m}{p^2} \right\rfloor - \ldots) \cdot p$$

Por tanto podemos usar esto para hacer una búsqueda binaria.

Adivina el factorial

Hay números que no se pueden determinar con la búsqueda binaria (porque las potencias de primos que servirían para determinarlos son más grandes que 10^9). Es decir, tenemos una lista de números que con la búsqueda binaria podemos determinar si n es menor o mayor o igual a esos números, y entre ellos hay huecos.
Como estos huecos son pequeños, una vez hemos determinado en cuál de estos huecos cae n con la búsqueda binaria, podemos ir comprobando una a una todas las posibilidades para n. Sólo necesitamos escoger a y b para poder determinar si n es cada posibilidad.

Idea

Si p es un primo mucho más grande que 40000, entonces $1!, 2!, \ldots, 40000!$ darán restos distintos al dividirse por p. (O al menos las colisiones no darán problemas).

Por tanto, podemos usar siempre el mismo primo p para cualquier hueco. Para encontrar p podemos hacer cualquier cosa sencilla como empezar desde $9 \cdot 10^8$ e ir comprobando números consecutivos uno a uno hasta encontrar un primo.

Adivina el factorial

Por tanto, lo que podemos hacer es:

1. Generar una lista de números $x_1, x_2, \ldots$ que podemos usar para la búsqueda binaria y determinar en qué intervalo de la forma $[x_i, x_{i+1} - 1]$ cae n.
2. Comprobar todas las opciones $x_i, x_i + 1, \ldots, x_{i+1} - 1$ una por una viendo si el resto al dividir $n!$ por un primo grande p es el mismo que el que daría $(x_i + j)!$.

Esta solución es suficiente para obtener más de 90 puntos en el problema. Para obtener 100 puntos, hay que fijarse en los detalles de cómo queda la distribución de números que podemos determinar con la búsqueda binaria.

Adivina el factorial

Tenemos un problema en hay una lista de elementos $x_1, x_2, \ldots x_m$ y podemos hacer preguntas de si el elemento que buscamos es menor o mayor que un elemento dado. Estas preguntas tiene un coste 1, pero lo que diferencia este problema del problema de búsqueda binaria típico es que si el resultado es x_i entonces pagamos un coste $c_i = x_{i+1} - x_i$ adicional. Queremos minimizar el coste en el peor caso, y la búsqueda binaria no es óptima porque siempre pagamos el mismo coste en preguntas, pero queremos que si el resultado es un x_i con un coste alto hacer menos preguntas para compensar el coste del resultado.
Es interesante pensar cómo resolver este problema en abstracto de forma exacta, pero no nos hace falta para este caso. Vemos que el intervalo más grande entre dos x_i consecutivos tiene longitud 57, entre el 31397 y el 31453. Los otros intervalos tienen como longitud máxima 48. Por tanto, si antes de hacer la búsqueda binaria miramos si es este intervalo el que tiene n, nos ahorraremos unas cuantas preguntas.

Adivina el factorial

Con esto podemos obtener entre 97 y 99 puntos. Para obtener los 100 puntos podemos usar alguna de estas ideas:

- Si eliminamos x_i que estén muy cerca unos de otros no aumentamos el número de preguntas en el peor caso, porque el cuello de botella son los intervalos grandes entre x_i consecutivos. Entonces podemos ahorrarnos un par de iteraciones de la búsqueda binaria intentando que los x_i estén equiespaciados.
- Hay un único intervalo de longitud 48, los demás son de longitud como máximo 42. Podemos comprobar manualmente el de 48 después de comprobar el de 57.

Scrabble

Enunciado

Ana tiene verdadero talento para jugar al Scrabble, el conocido juego de palabras. Los jugadores se turnan para añadir letras al tablero, encadenando nuevas palabras con palabras ya colocadas.
Su amigo Juan no se cree que sea tan buena y decide retarla. Le ha dicho que le comprará una tableta de chocolate si consigue ganarle una partida. Ana acepta sin dudarlo. Tiran una moneda y le toca a Ana el primer turno, por lo que quiere aprovechar esta ventaja al máximo.

Scrabble

Enunciado

Resulta que el primer jugador puede colocar la palabra que él desee en el tablero mientras esta esté en el diccionario, ya que no tiene letras previas con las que encadenarla. Sabiendo las letras que tiene Ana (estas siempre serán 7), ¿cuál es el máximo de puntos que puede obtener esa jugada? El número de puntos es igual a la longitud de la palabra, salvo que esta **supere** los tres caracteres en cuyo caso se duplicará. La que forme Ana debe de estar en el diccionario que te suministraremos. Ana no podrá usar la misma ficha dos veces, es decir, que si tiene una ficha con la letra A y una ficha con la letra N, no podrá formar la palabra ANA salvo que tenga una segunda ficha con la letra A.

Autora: Blanca Huergo Muñoz

Scrabble

Idea

Una palabra en el diccionario puede formarse con las fichas actuales si, para cada letra del alfabeto, se cumple que el número de veces que está la letra en la palabra es igual o inferior al número de fichas que la contienen. Con esta observación, podemos calcular la palabra más larga que podemos formar comprobando esto para todas las palabras del diccionario y quedándonos con la más larga. Tras procesar todo el diccionario, imprimimos la longitud de esta palabra, multiplicada por dos si supera los tres caracteres, y 0 en caso de no haber encontrado ninguna válida.

Scrabble

Código

En problemas con restricciones laxas, como es este caso, podemos aprovecharnos para hacer un programa sencillo, posiblemente no el óptimo, pero que resuelva el problema en cuestión. Así, evitamos gastar tiempo valioso buscando descuidos.
Una solución sencilla es guardar cada palabra como una lista de 26 posiciones, guardando en la posición 0 la frecuencia de la letra A, luego la B, etc.
Si hacemos lo mismo con las fichas que tenemos, podemos hacer cada comparación con un bucle que itera sobre las 26 posiciones.

Estatuas de Merliones

Enunciado

Steven está paseando por Singapur y se ha encontrado una pasarela en la que hay una serie estatuas de merliones a los lados, formando arcos de agua por encima de la pasarela. Steven ha decidido pasear por la pasarela, y mientras pasea va a ir anotando en su libreta a qué lado se va encontrando las estatuas: a su izquierda o a su derecha. Steven empieza caminando hacia adelante en la entrada de la pasarela, pero puede decidir darse la vuelta y caminar en sentido contrario en cualquier momento del paseo. Su paseo tiene que empezar y acabar en el mismo sitio: la entrada de la pasarela. Ahora Steven quiere recordar cómo era la pasarela y el paseo que hizo por ella, pero sólo dispone de las anotaciones en su libreta. A partir de estas anotaciones, debes reconstruir una posible disposición de las estatuas en la pasarela y un posible paseo que pueda haber hecho Steven que de lugar a las anotaciones de la libreta.

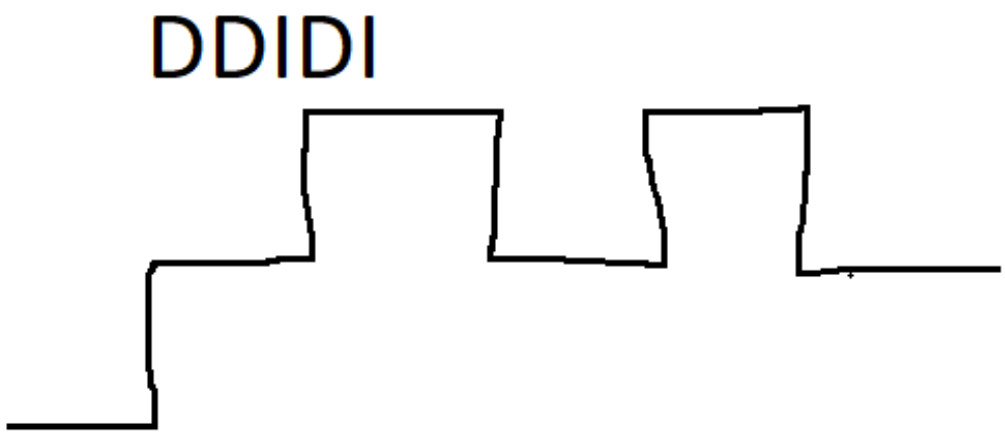
Autor: Félix Moreno Peñarrubia

Estatuas de Merliones

Cambio de perspectiva

Podemos pensar en el camino como escalones que hacia arriba y hacia abajo, una estatua a la derecha será un escalón hacia arriba y una a la izquierda hacia abajo. Nótese que esta forma de verlo es consistente cuando vamos en el otro sentido, una estatua a la derecha se percibe como una estatua a la izquierda como un escalón hacia arriba se percibe como uno hacia abajo y viceversa.

Por ejemplo:



Estatuas de Merliones

Representación de la pasarela

Ahora podemos representar la pasarela con números enteros (posiblemente negativos) que indican la altura de cada posición, dos posiciones consecutivas se diferencian entonces en exactamente uno de altura.
Asumiremos que el comienzo de la pasarela está a altura 0.
IIIDIDDD $\rightarrow$ 0 -1 -2 -3 -2 -3 -2 -1 0

Representación del recorrido

El recorrido puede ser representado de la misma forma que la pasarela. Si no acaba en 0 sabemos que no hay solución.

Estatuas de Merliones

Construcción de la pasarela

Necesitamos una pasarela en la que todos los números aparezcan por primera vez en el mismo orden que en el recorrido, esto puede lograrse comenzando solo con un 0 y añadiendo al final de la pasarela una escalera (escalones consecutivos en un mismo sentido) cada vez que aparezca un número nuevo para llevar hasta él.

Con el recorrido 0 -1 -2 -3 -2 -3 -2 -1 0 1 2 1 0 -1 0
Las escaleras serían las siguientes:

1. -1 al encontrar -1
2. -2 al encontrar -2
3. -3 al encontrar -3
4. -2 -1 0 1 al encontrar 1
5. 2 al encontrar 2

Por tanto la pasarela sería 0 -1 -2 -3 -2 -1 0 1 2.

Estatuas de Merliones

Cuando sí hay solución

Nótese que esta es la pasarela más pequeña que contiene todos los números en el orden necesario. Si en el recorrido los números aparecen por primera vez en el orden contrario a como aparecen por última vez, entonces este recorrido se puede hacer en la pasarela que hemos construido de la siguiente forma:

1. Si solo uno de los números de al lado corresponde con el que marca el recorrido, vamos a ese.
2. Si a ambos lados tenemos el número correcto y el número en el que estamos aparecerá de nuevo en el recorrido, avanzamos.
3. Si a ambos lados tenemos el número correcto y el número en el que estamos aparece por última vez en el recorrido, retrocedemos.

Estatuas de Merliones

Ejemplo

Con el recorrido 0 -1 -2 -1 -2 -1 0 1 0 -1 -2 -1 0
La pasarela es 0 -1 -2 -1 0 1
El orden de primera aparción de los números es 0 -1 -2 1 y el orden de última aparición es 1 -2 -1 0, por tanto cumple la hipótesis.
El recorrido se puede hacer en la pasarela avanzando del 0 hasta el segundo -1, retrocediendo al -2, avanzando hasta el 1 y retrocediendo hasta el primer 0.

Estatuas de Merliones

Cuando no hay solución

Como hemos visto antes, si el recorrido no acaba en 0 no hay solución. Además de esto si en el recorrido los números no aparecen por primera vez en el orden contrario a como aparecen por última vez tampoco hay solución. Esto es porque no podemos llegar de un número al principio de la pasarela sin pasar por todos los números que hay en medio.

Estatuas de Merliones

Ejemplo

Con el recorrido 0 -1 -2 -1 -2 -1 0 1 0
El orden de primera aparción de los números es 0 -1 -2 1 y el orden de última aparición es -2 -1 1 0, por tanto no tiene solución.
Con el recorrido 0 1 2 3 2 1
No acaba en 0, por tanto no tiene solución.

Estatuas de Merliones

Implementación

Nuestro programa debe:

1. Convertir el input a un vector de ints con nuestra representación y comprobar si acaba en 0. Si no imprimimos "NO" y terminamos. O(n)
2. Marcar la primera y la última vez que aparece cada número en el recorrido. O(n)
3. Comprobar la hipótesis sobre el orden de aparición de los números. Si no se cumple imprimimos "NO" y terminamos, si se cumple imprimimos "SI". O(n)
4. Construir la pasarela e imprimirla. Las escaleras como mucho miden la distancia desde el último número por el que se añadió una escalera hasta el actual, por tanto esto es lineal. O(n)
5. Guardar los índices en los que se cambia de sentido al hacer el recorrido en la pasarela e imprimirlos. O(n)

Bicoloración con distancias

Enunciado

Se da un grafo no dirigido con pesos en las aristas. Determinar el máximo d tal que se pueda colorear el grafo de dos colores tal que dos vértices a distancia menor que d sean de colores distintos.
(La distancia entre dos vértices es la mínima suma de pesos en las aristas en un camino entre ellos).

Autor: Félix Moreno Peñarrubia

Bicoloración con distancias

Observación

Para un determinado valor de d, estas dos condiciones son necesarias para que sea posible:

- El grafo sólo con las aristas con peso $< d$ es bipartito. (Necesaria porque si no sólo considerando las distancias con caminos de una arista ya no podríamos bicolorear)
- Por cada vértice de grado ≥ 2, la suma de los pesos de las dos aristas de menor peso es $\geq d$. (Necesaria porque los vecinos con aristas de peso $< d$ tienen que ser de color opuesto al vértice, y si hay dos que suman menos que d entonces habría un camino de dos aristas de suma menor que d entre dos vértices del mismo color).

Bicoloración con distancias

Observación

Para un determinado valor de d, estas dos condiciones son necesarias para que sea posible:

- El grafo sólo con las aristas con peso $< d$ es bipartito. (Necesaria porque si no sólo considerando las distancias con caminos de una arista ya no podríamos bicolorear)
- Por cada vértice de grado ≥ 2, la suma de los pesos de las dos aristas de menor peso es $\geq d$. (Necesaria porque los vecinos con aristas de peso $< d$ tienen que ser de color opuesto al vértice, y si hay dos que suman menos que d entonces habría un camino de dos aristas de suma menor que d entre dos vértices del mismo color).

Idea clave

Por separado no son condiciones suficientes, pero juntas sí, ya que si se satisface la segunda condición entonces la bicoloración dada por la primera condición sirve.

Bicoloración con distancias

Solución

Encontramos mediante una búsqueda binaria el máximo valor de d tal que se satisfagan ambas condiciones. Complejidad $\mathcal{O}((n+m)\log W)$, donde W es el máximo peso que puede haber en las aristas. Caso especial: si no hay vértices de grado ≥ 2, la respuesta es ∞.

Oficina de objetos perdidos

Enunciado

Tenemos n estaciones, situadas en posiciones $a_i, i \in \{0, \cdots, n-1\}$ y con popularidad $p_i, i \in \{0, \cdots, n-1\}$. Queremos encontrar la estación más *cercana* al resto, es decir, que minimice $\sum_{i \neq j} p_j |a_i - a_j|$

Autor: Alberto Maurel Serrano

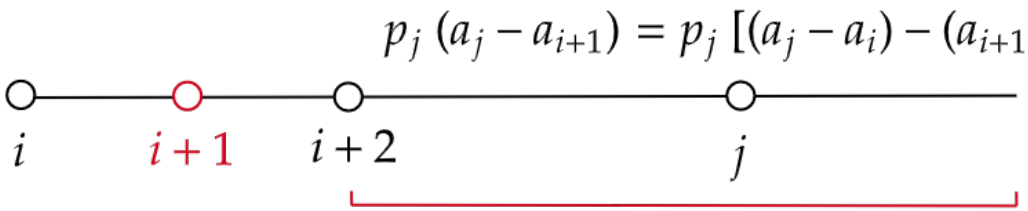
Oficina de objetos perdidos

Solución 1

Dado que $n \leq 3 \cdot 10^5$, no podemos calcular para cada estación el sumatorio directamente. Sin embargo, sí que podríamos calcularlo para el primero y luego aprovechar esa información para calcular los demás sin repetir los cálculos.
Esto lo podemos hacer porque en la fórmula, p_j no depende de la i.

Oficina de objetos perdidos

$$\sum_{j>i} p_j(a_j - a_i)$$



$$\sum_{j>i+1} p_j(a_j - a_{i+1})$$

Oficina de objetos perdidos

$$\sum_{j>i+1} p_j(a_j - a_{i+1}) = \sum_{j>i+1} p_j[(a_j - a_i) - (a_{i+1} - a_i)] =$$

$$= \sum_{j>i+1} p_j(a_j - a_i) - (a_{i+1} - a_i) \sum_{j>i+1} p_j =$$

$$= \sum_{j>i+1} p_j(a_j - a_i) - (a_{i+1} - a_i) \sum_{j>i+1} p_j + p_{i+1}(a_{i+1} - a_i) - p_{i+1}(a_{i+1} - a_i) =$$

$$= \sum_{j>i} p_j(a_j - a_i) - (a_{i+1} - a_i) \sum_{j>i} p_j$$

Oficina de objetos perdidos

Solución 2

Sin embargo, hay otra solución. Se puede demostrar que la función

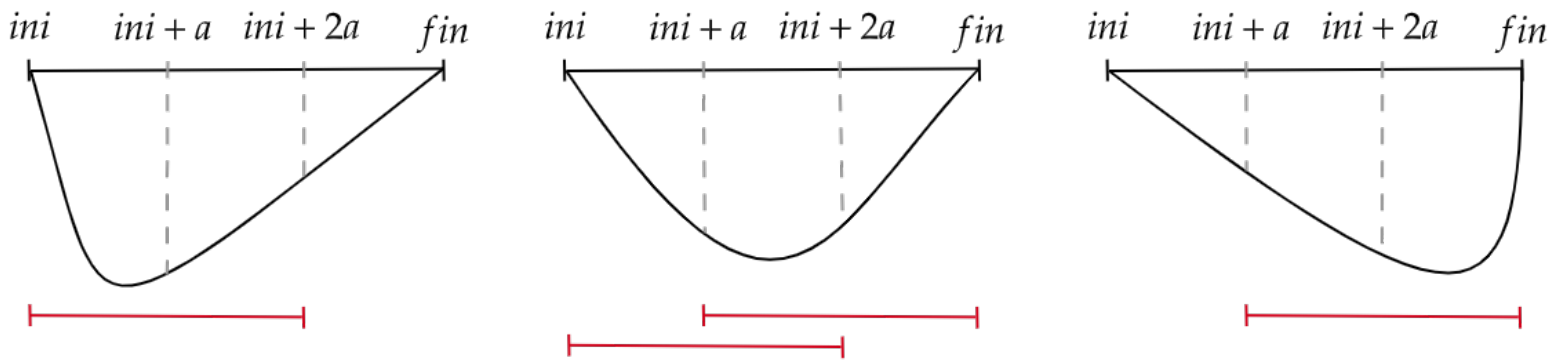
$$\sum_{i \neq j} p_j |a_i - a_j|$$

como función de i es cóncava. El problema entonces se reduce a encontrar el mínimo en una función cóncava, que se puede resolver con búsqueda ternaria.

Referencia: *Guide to Competitive Programming, Antti Laaksonen, Springer, Sección 8.3.1 Ternary Search*

Oficina de objetos perdidos

Sea $a = \frac{fin - ini}{3}$



En cada paso descartamos un tercio del intervalo restante. Podemos calcular directamente el valor de $f(i) = \sum_{i \neq j} p_j |a_i - a_j|$, puesto que no necesitamos calcularlo para todos los n, sino para un número logarítmico de ellos.
Complejidad: **O(n log n)**

Oficina de objetos perdidos

Solución 3

Originalmente, el problema era miniminizar $\sum_{i \neq j} |a_i - a_j|$ (y de hecho esta es la subtarea 3). Existe un teorema que nos dice que el mínimo de esta función es la mediana.
Podemos adaptar el problema a ese *"duplicando"* las estaciones: consideramos que hay p_i estaciones i-ésimas y la solución será la estación que se encuentre en la mediana de este nuevo array.
Complejidad: **O(n)**

¡Gracias a *Pablo Hidalgo* por la idea!
Referencia: *Guide to Competitive Programming, Antti Laaksonen, Springer, Sección 8.3.3 Minimizing sums*