Final OIE 2021 Día 2

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

Atrapasueños

Enunciado

Dados dos enteros n, k se pide, en caso de que sea posible, construir una permutación p de $1, \ldots, n$ tal que si tienes n puntos en una circumferencia numerados de 1 a n y dibujas los $n-1$ segmentos uniendo p_i con p_{i+1} entonces hay exactamente k pares de segmentos que se intersecan en el interior del círculo.

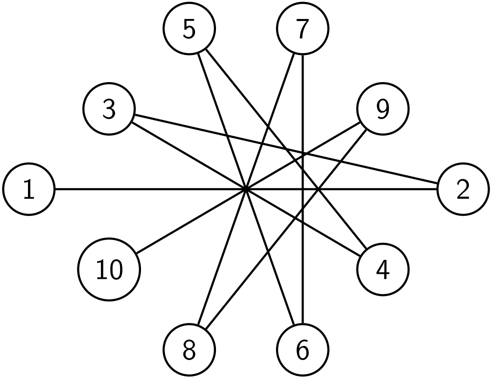
Autor: Félix Moreno Peñarrubia

Atrapasueños

El número máximo de segmentos anteriores que puede intersecar el i-ésimo segmento es $i-2 \implies$ el número máximo posible de intersecciones es $1+2+\ldots+(n-3)=\frac{(n-3)(n-2)}{2}$.
Si $k>\frac{(n-3)(n-2)}{2}$, la respuesta es NO. Veremos que en cualquier otro caso sí que se puede. Si $k\leq\frac{(n-4)(n-3)}{2}$, entonces podemos unir el punto 1 y el 2 y considerar los puntos $2,\ldots,n$ como si estuviéramos en el caso $n-1$.
Por tanto nos va a interesar el caso $k\in\left[\frac{(n-4)(n-3)}{2}+1,\frac{(n-3)(n-2)}{2}\right]$

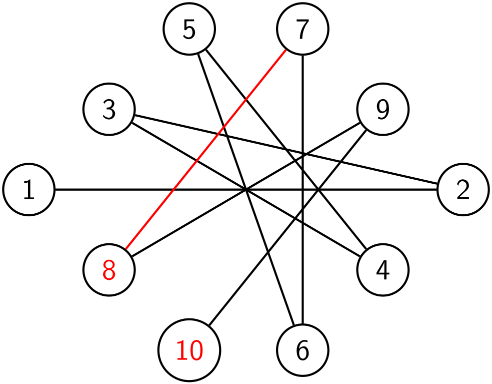
Atrapasueños

Caso $k = \frac{(n-3)(n-2)}{2}$ con $n = 10$



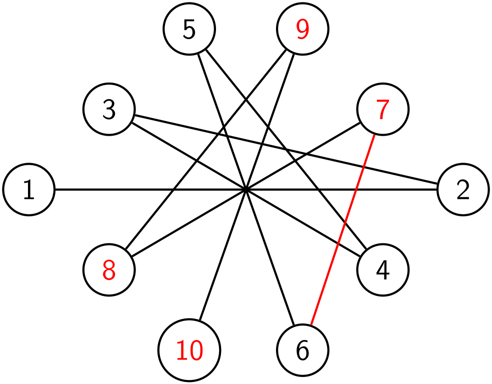
Atrapasueños

Caso $k = \frac{(n-3)(n-2)}{2} - 1$ con $n = 10$



Atrapasueños

Caso $k = \frac{(n-3)(n-2)}{2} - 2$ con $n = 10$



Mínimo XOR

Enunciado

Hay un conjunto de n enteros no negativos S que hay que determinar. Se pueden hacer preguntas en las que se envía un entero no negativo y y se devuelve el valor $\min_{x \in S} x \oplus y$, donde $\oplus$ es la operación disyuntiva exclusiva bit a bit (XOR).
Determinar el conjunto usando como máximo $2n$ preguntas.

Autor: Félix Moreno Peñarrubia

Mínimo XOR

Propiedades de $\oplus$ importantes:

- $\oplus$ es asociativo (y conmutativo).
- $x \oplus 0 = x$.
- $x \oplus x = 0$.

Mínimo XOR

Propiedades de $\oplus$ importantes:

- $\oplus$ es asociativo (y conmutativo).
- $x \oplus 0 = x$.
- $x \oplus x = 0$.

Esto significa que podemos recuperar el x que hace mínimo el valor $x \oplus y$:

$$\arg\min_{x \in S} x \oplus y = \left(\min_{x \in S} x \oplus y\right) \oplus y$$

Mínimo XOR

Propiedades de $\oplus$ importantes:

- $\oplus$ es asociativo (y conmutativo).
- $x \oplus 0 = x$.
- $x \oplus x = 0$.

También:

- $x \oplus \overline{y} = \overline{x \oplus y}$ (donde $\overline{x}$ indica el complemento bit a bit de x)
- $\max_{x \in S} x \oplus y = \overline{\min_{x \in S} \overline{x \oplus y}} = \overline{\min_{x \in S} x \oplus \overline{y}}$

Esto significa que podemos extraer el mínimo y el máximo del conjunto preguntando por $y = 0$ y por $y = \overline{0}$.

Mínimo XOR

$$
\begin{array}{c|cc}
x_n & 10110 & \dots \\
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
x_{i+1} & 10001 & \dots \\
x_i & 01101 & \dots \\
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
x_1 & 00001 & \dots
\end{array}
$$

Organigramas equilibrados

Enunciado

Tu mafia de confianza tiene un organigrama muy estricto. Cada uno de sus miembros tiene un solo superior (salvo el *capo di tutti i capi*, que no tiene superiores) y cero, uno, o dos subordinados. El *capo di tutti i capi* quiere armar a la banda, pero debe cumplir la siguiente restricción. Para cada miembro que tiene dos subordinados, la cantidad de miembros armados que cuelgan del primer subordinado (incluyéndolo) debe coincidir con la cantidad de miembros armados que cuelgan del segundo, pues de lo contrario habría un desequilibrio de poder entre los subordinados.
Sabiendo el organigrama de la organización, te han encargado decidir cuál es el número máximo de miembros que pueden armar.

Autor: Félix Moreno Peñarrubia

Organigramas equilibrados

Idea

Modelaremos el problema con un árbol, donde el *capo di tutti i capi* sea la raíz y los subordinados directos de cada nodo sus hijos. Esto significa que las aristas serán unidireccionales, de padres a hijos.
Si un nodo no tiene hijos (es una hoja del árbol), el máximo número de capos armados que puede haber es 1 (ese mismo nodo).
Si un nodo tiene 1 hijo, el máximo número de capos armados que puede haber es 1 + el máximo número de capos armados en el subárbol que cuelga de su hijo (incluyéndolo).

Organigramas equilibrados

Idea

El caso difícil es cuando hay más de 1 hijo. La observación clave para resolverlo es ver que, si el máximo número de capos armados colgando del subárbol de un hijo es k, se puede construir un subárbol con cualquier número entre 1 y k de capos armados. Por lo tanto, podemos resolver este problema de forma voraz (greedy). Calculamos para cada hijo el máximo de capos armados en su subárbol. Cogemos el mínimo de estos 2 y lo retornamos, multiplicado por 2, y sumando 1 (ya que podemos armar al padre).

Organigramas equilibrados

Código

La idea se traduce de forma sencilla a una solución recursiva basada en DFS (búsqueda en profundidad). La fórmula para el caso de 2 hijos es 1 + 2 * mínimo, para 1 hijo es 1 + minimo y para 0 hijos es 1. Esto significa que, en el caso general, la fórmula es 1 + número de hijos * minimo. Por lo tanto, calculamos de forma recursiva el mínimo de los hijos y retornamos el valor devuelto por la fórmula. Al ser un árbol, no necesitamos usar programación dinámica, ya que no volveremos a necesitar el valor de un nodo tras retornarlo. Por lo tanto, la respuesta será `armados(0)`.

Caballeros y escuderos

Enunciado

Hay una fila con n personas, de las cuales m son caballeros y el resto escuderos. Los caballeros siempre dicen la verdad y los escuderos siempre mienten.
Puedes preguntar a una persona de la fila "¿En qué dirección se encuentra el caballero más cercano? (distinto a ti, en caso de que seas caballero)" (la distribución será tal que siempre hay un único caballero más cercano).
Debes identificar a todos los caballeros con como máximo $100 \cdot (m+1)$ preguntas. ($n \leq 10^9$)

Autor: Félix Moreno Peñarrubia

Caballeros y escuderos

Primero vamos a resolver el problema asumiendo que conocemos las respuestas de todas las personas de la fila. No es obvio que a partir de las respuestas se pueda determinar únicamente las posiciones de los caballeros.

E	E	C	E	E	C	C	E	E	E	E	C	E
$<$	$<$	$>$	$>$	$<$	$>$	$<$	$>$	$>$	$<$	$<$	$<$	$>$

Caballeros y escuderos

Primero vamos a resolver el problema asumiendo que conocemos las respuestas de todas las personas de la fila. No es obvio que a partir de las respuestas se pueda determinar únicamente las posiciones de los caballeros.

E	E	C	E	E	C	C	E	E	E	E	C	E
$<$	$<$	$>$	$>$	$<$	$>$	$<$	$>$	$>$	$<$	$<$	$<$	$>$

Idea clave 1

Hay una correspondencia uno a uno entre los $<>$ y los caballeros.
Asumiendo que no hay caballeros en la primera o última posición de la fila.

Caballeros y escuderos

Primero vamos a resolver el problema asumiendo que conocemos las respuestas de todas las personas de la fila. No es obvio que a partir de las respuestas se pueda determinar únicamente las posiciones de los caballeros.

E	E	C	E	E	C	C	E	E	E	E	C	E
$<$	$<$	$>$	$>$	$<$	$>$	$<$	$>$	$>$	$<$	$<$	$<$	$>$

Idea clave 1

Hay una correspondencia uno a uno entre los $<>$ y los caballeros.
Asumiendo que no hay caballeros en la primera o última posición de la fila.

En general, a partir del $<>$ no podemos saber cuál de los dos es el caballero, pero el primer caballero tiene que ser el que tenga la posición $>$ en el primer $<>$, y a partir de ahí los demás los determinamos teniendo en cuenta que las distancias entre caballeros consecutivos deben ser impares. Así la respuesta queda únicamente determinada.

Caballeros y escuderos

Por tanto, encontrar los caballeros se reduce a encontrar los $<>$. Vamos a ver cómo hacer eso eficientemente. Asumiremos a partir de ahora que al principio y al final de la fila hay escuderos (después veremos cómo hacerlo cuando no los hay).
Entonces, al principio de la fila hay un $<$ y al final hay un $>$. Si solo hubiera un único $<>$, lo encontraríamos usando búsqueda binaria para ver cuándo acaban los $<$ y empiezan los $>$. Esta idea también funciona para cuando hay varios $<>$: empezando por el intervalo completo, mantenemos un intervalo que tiene un $<$ en el extremo izquierdo y un $>$ en el derecho, y lo vamos reduciendo por la mitad, cuando lleguemos a longitud 2 tendremos un $<>$.

Idea clave 2

Podemos encontrar un $<>$ haciendo búsqueda binaria.

Caballeros y escuderos

¿Podemos encontrar más de un $<>$ con la búsqueda binaria?
Un intento que podríamos hacer es: una vez hemos encontrado nuestro $<>$ inicial, repetir la búsqueda binaria en los dos intervalos a los lados de este $<>$. Pero ahora ya no empezamos por un intervalo que tenga $<$ y $>$ como extremos: podríamos hacer la misma búsqueda binaria, pero es posible que nos converja al extremo del intervalo, es decir, "quiere converger" al mismo $<>$ que hemos visto antes. Iniciando la búsqueda binaria en diferentes intervalos puede que converja a diferentes $<>$, pero no hay ninguna garantía de encontrarlos todos de esta forma.
Por tanto, necesitamos una idea diferente para encontrar el resto de $<>$.

Caballeros y escuderos

Si tenemos a un $<>$, ¿cómo encontrar los $<>$ anterior y posterior? Entre dos $<>$ hay un $><$:

C?	C?	E	E	E	E	E	E	C?	C?
$<$	$>$	$>$	$>$	$>$	$<$	$<$	$<$	$<$	$>$

Caballeros y escuderos

Si tenemos a un $<>$, ¿cómo encontrar los $<>$ anterior y posterior? Entre dos $<>$ hay un $><$:

C?	C?	E	E	E	E	E	E	C?	C?
$<$	$>$	$>$	$>$	$>$	$<$	$<$	$<$	$<$	$>$

¡Cuidado! No tiene por qué ser simétrico del todo:

C?	C?	E	E	E	E	E	C?	C?
$<$	$>$	$>$	$>$	$>$	$<$	$<$	$<$	$>$

Caballeros y escuderos

Si tenemos a un $<>$, ¿cómo encontrar los $<>$ anterior y posterior? Entre dos $<>$ hay un $><$:

C?	C?	E	E	E	E	E	E	C?	C?
$<$	$>$	$>$	$>$	$>$	$<$	$<$	$<$	$<$	$>$

Idea clave 3

Haciendo saltos de potencias de 2 sucesivas, podemos encontrar el punto $><$ y por tanto los $<>$ adyacentes a un $<>$ determinado.

Caballeros y escuderos

Si tenemos a un $<>$, ¿cómo encontrar los $<>$ anterior y posterior? Entre dos $<>$ hay un $><$:

C?	C?	E	E	E	E	E	E	C?	C?
$<$	$>$	$>$	$>$	$>$	$<$	$<$	$<$	$<$	$>$

Idea clave 3

Haciendo saltos de potencias de 2 sucesivas, podemos encontrar el punto $><$ y por tanto los $<>$ adyacentes a un $<>$ determinado.

Hay que tener cuidado de no salirse de la fila cuando se llega a los últimos $<>$ de cada lado.

Caballeros y escuderos

Coste total de encontrar los m caballeros (peor caso, acotado generosamente por arriba):

- $\log n$ por la búsqueda binaria inicial.
- $2 \cdot \log n$ por cada avance de un $<>$ al adyacente ($\log n$ por hacer los saltos en potencias de 2 y $\log n$ para la búsqueda binaria para encontrar el $><$).
- $2 \cdot \log n$ por detectar los $<>$ en los dos extremos.

Total:

$$\log n + (m-1) \cdot 2 \log n + 2 \log n = m \cdot 2 \log n + \log n < (m+1) \cdot 2 \log n <$$

$$< (m+1) \cdot 60 < 100 \cdot (m+1)$$

Caballeros y escuderos

Detalles de implementación:

- Hemos asumido en nuestra solución que no había caballeros en los extremos de la fila. Una forma elegante y sencilla de solucionar este caso es "extender" la fila con infinitos escuderos a izquierda y a derecha. Entonces sí que se cumple la correspondencia de $<>$ con los caballeros. Si se hace una función para hacer las preguntas, basta con hacer que la función retorne $<$ si el índice es < 0 o $>$ si es $> n-1$.
- En algunos casos hay que ir con mucho cuidado con los índices, porque es fácil cometer errores de ± 1 (debido principalmente a la incertidumbre que hay en las posiciones de los caballeros). Una forma de simplificar la casuística es programar exploraciones lineales alrededor de "puntos de interés". Como el límite de preguntas era muy generoso no hay ningún problema en hacer más preguntas de las estrictamente necesarias.

Rompepuertas

Enunciado

Tenemos n puertas con unas ciertas durezas $d_0, \ldots, d_{n-1}$ y m personas con fuerzas $f_0, \ldots, f_{m-1}$. Los concursantes salen de forma ordenada y tienen que romper las puertas por orden. Hay que averiguar cuántas puertas son capaces de romper.
Además, un concursante puede romper varias puertas si la suma de sus durezas no sobrepasa su fuerza.

Autor: Alberto Maurel Serrano

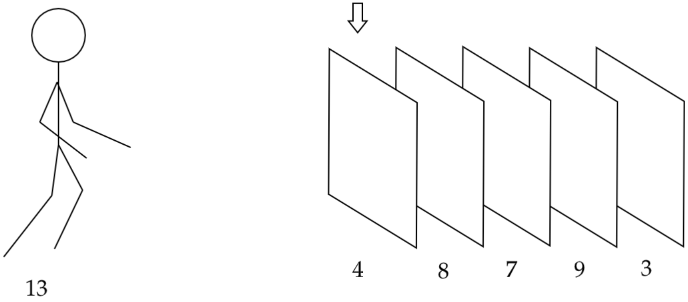
Rompepuertas

Simplemente tenemos que simular el proceso. Para ello, llevamos un índice que nos indique hasta qué puerta hemos roto hasta el momento. Para cada concursante, rompemos tantas puertas como permita su fuerza, manteniendo el índice actualizado. Esto nos lleva a un algoritmo con complejidad en $O(n + m)$

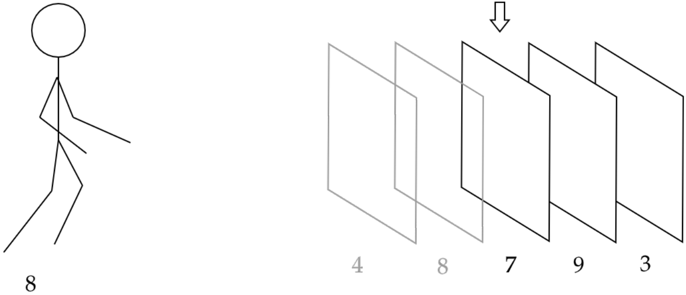
Cuidado

- El excedente de fuerza hay que descartarlo: *"Esto significa que o las rompes o no las rompes, pero golpearlas con menos fuerza de la que necesitan para romperse no las debilitará."*
- Puede que algún concursante no pueda romper ninguna puerta, o que no hagan falta todos los concursantes para romperlas todas.

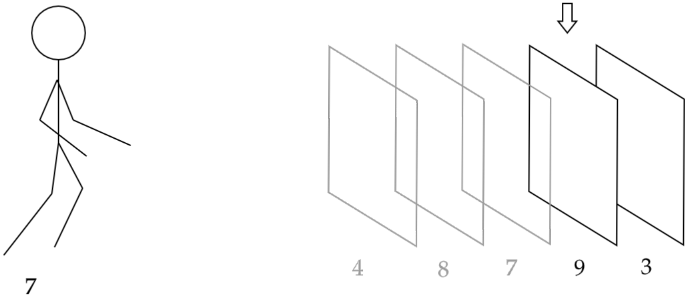
Rompepuertas



Rompepuertas



Rompepuertas



Rompepuertas