Día 2: Cartas

Resumen del enunciado

Determina una permutación $c_1, \ldots, c_n$ haciendo preguntas de $\min(c_i, c_j)$.

- *Autor: Darío Martínez Ramírez*
- Primera solución con 100 puntos: Daniel Nieto Pérez.

Solución:

- Mantenemos dos candidatos para el máximo c_u, c_v, y el valor $M = \min(c_u, c_v)$.
- Preguntamos $m = \min(c_u, c_i)$:
  - Si $m < M$, sabemos $c_i = m$.
  - Si $m = M$, $c_u = m$ y actualizamos $u = i$.
  - Si $m > M$, $c_v = m$ y actualizamos $v = i$.
- En peor caso son $2n$ preguntas, si se procesa en orden aleatorio son $\approx n + \log n$ en esperanza.

Día 2: Igualando

Resumen del enunciado

Dados n, k, m, r, y un vector a, podemos añadir m a un elemento y restar r al resto, encuentra el número mínimo de operaciones para que todos los elementos de a sean mayores que k.

- *Autor: Manuel Torres Cid*
- Primera solución con 100 puntos: Javier Badesa Perez.

Solución:

- Si se puede conseguir el objetivo con s operaciones, se puede con $s+n$ operaciones, ya que si aplicamos la operación a cada elemento del vector, todos los valores crecen en $m - r \cdot (n-1)$ y $m > r \cdot (n-1)$
- Podemos realizar una búsqueda binaria independiente para cada i de 0 a $n-1$, en la que buscamos entre valores de la forma $t \cdot n + i$, siendo t un entero.

Día 2: Rectángulo

Resumen del enunciado

Adivina las dimensiones de un rectángulo (a, b) con preguntas de si existe una colocación con la que puedas meter dentro otro rectángulo de dimensiones (c, d).

- *Autor: Félix Moreno Peñarrubia*
- Primera solución con 100 puntos: Javier Andrés García Martínez

Solución:

- Para conocer la longitud del lado más pequeño podemos adivinarla aproximando por cuadrados. Para usar la mínima cantidad de queries posibles haremos una búsqueda binaria.
- Una vez que ya tenemos el lado más pequeño, para conocer el grande podemos hacer una segunda búsqueda binaria.

Día 2: Resto

Resumen del enunciado

Dado un vector, calcula preguntas del tipo:
$f(x) = (x \% a_1) + ((x \% a_1) \% a_2) + (((x \% a_1) \% a_2) \% a_3) + \ldots + ((\ldots(x \% a_1)\ldots) \% a_n)$.

- *Autor: Manuel Torres Cid*
- Primera solución con 100 puntos: Daniel Nieto Perez.

Solución:

- ¿Cuál es la complejidad del algoritmo de euclides? ¿Por qué?. La secuencia $(x \% a_1), ((x \% a_1) \% a_2), (((x \% a_1) \% a_2) \% a_3), \ldots, ((\ldots(x \% a_1)\ldots) \% a_n)$ tiene como mucho $\lceil \log(a_1) \rceil$ valores diferentes.
- Podemos calcular las preguntas buscando los valores donde la secuencia cambia, para ello, si nos encontramos en la posición i-ésima con valor x, queremos buscar el primer valor menor que x en el rango $[i+1, n]$. Esto podemos calcularlo con un árbol de segmentos.

Día 2: Resto

- Usando un árbol de segmentos, podemos buscar los $O(\log(n))$ nodos que definen el rango $[i+1, n]$, ordenarlos, buscar el primer nodo que contiene un valor menor que x y realizar una búsqueda dicotómica desde este nodo. Este último algoritmo a veces es conocido como *Walking on segment trees*.

Día 2: Xordenamiento

Resumen del enunciado

Realiza operaciones XOR entre los elementos de una lista hasta conseguir una lista ordenada con pocas operaciones.

- *Autor: Innokentiy Kaurov*

Solución:

- Podemos hacer swap de a_i y a_j con $(i,j),(j,i),(i,j)$. (30 puntos)
- Podemos convertir a en una permutación para simplificar.
- Podemos arreglar un ciclo de longitud l en $3l-3$ operaciones.
- Si $a_1 = 0$ podemos arreglar un ciclo de longitud l en $2l+2$ operaciones usando que se puede hacer swaps con un 0 en 2 operaciones.
- Podemos conseguir $a_1 = 0$ haciendo búsqueda exhaustiva con los 21 primeros elementos. Como mucho 23 operaciones.

Día 2: Xordenamiento

Resumen del enunciado

Realiza operaciones XOR entre los elementos de una lista hasta conseguir una lista ordenada con pocas operaciones.

- *Autor: Innokentiy Kaurov*

Solución:

- Hacemos $a_1 = 0$ en 23 operaciones. El resto lo convertimos en permutación y, si tiene menos de 200 ciclos, resolvemos cada ciclo en $2l + 2$ (máximo de 2400 operaciones), y, si tiene más de 200 ciclos, resolvemos cada ciclo en $3l - 3$ (máximo de 2400 operaciones).
- Alternativa: Si realizamos unas cuantas operaciones aleatorias, la permutación resultante tiene muy pocos ciclos con probabilidad muy alta.