Día 1: Caminos

Resumen del enunciado

Encuentra la flecha que se tiene que cambiar para que haya un camino que pase por todas las casillas.

- *Autora: Blanca Huergo Muñoz*

Solución:

- No es difícil comprobar si la respuesta es una casilla concreta en tiempo $O(n^2)$.
- Casi todas las casillas deben tener otra casilla que apunta a ella, excepto la que sea adyacente a la casilla modificada. Se comprueban las 4 casillas adyacentes a ella.

Día 1: Igualando 2

Resumen del enunciado

Se te dan n números y puedes hacer las operaciones sumar 1 a uno de ellos, o elegir $k+1$ de ellos, restarle 1 a k de los números y sumarle k al que queda. Haz que todos sean iguales usando la primera operación el mínimo número de veces.

- *Autor: Catalin Covaci*

Solución:

- Primer caso: $k \leq n-2$
- Nos damos cuenta de que la segunda operación no cambia la media de los números. Por tanto tenemos que hacer la primera hasta que la media sea entera. Para esto necesitas $(n - suma)\ \%n$ operaciones.
- Nos damos cuenta de que aplicando la operación 2 muchas veces podemos hacer lo contrario: Sumar 1 en k posiciones y restar k en una.
- Aplicando la operación 2 en $i_1, \ldots, i_k, i_{k+1}$ y la opuesta en $i_1, \ldots, i_k, i_{k+2}$ acabamos sumando 1 en la posición i_{k+2} y restando 1 a la posición i_{k+1}. Usando esto repetidamente, si la media es entera hacemos que todos sean iguales.

Día 1: Igualando 2

- Segundo caso: $k = n - 1$
- Si miramos los números módulo n, la segunda operación suma 1 a todos. Por tanto hay que aplicar la primera hasta que todos sean iguales módulo n.
- Puedes calcular cuantas operaciones hacen falta para hacer que todos valgan $x \mod n$ para todo residuo x y escojer el mejor. Con una sliding window, se puede calcular en $O(n)$.
- Una vez todos valen lo mismo módulo n, para hacerlos iguales simplemente opera restando a los k más grande y sumando al más pequeño. Se puede ver que haciendo esto eventualmente serán iguales.

Día 1: Senderismo

Resumen del enunciado

Dado un grafo no dirigido con pesos, queremos encontrar un camino desde el nodo 1 hasta el nodo n, de tal forma que el número de veces que incrementamos el índice entre nodos adyacentes en nuestro camino sea mínimo. En caso de empate, escogemos el camino con peso mínimo.

- *Autor: Manuel Torres Cid*

Solución:

- Consideraremos la distancia que queremos minimizar como una pareja {número de incrementos, peso} (ordenando como un `std::pair` de C++).
- Usaremos el algoritmo de Dijkstra para minimizar esta distancia. Para evitar procesar nodos más de una vez, ordenamos nuestro heap (`priority_queue` en C++) según la pareja {número de incrementos, -índice del nodo} (también ordenando como un `std::pair` de C++). Ordenar directamente por distancia tendría una complejidad cuadrática.

Día 1: ADN

Resumen del enunciado

Construye una biyección explícita entre tuplas de cadenas binarias (X, Y_1, Y_2, Y_3) con $X > \mathbf{máx}(Y_1, Y_2, Y_3)$ y parejas de cadenas de ADN con un número impar de As.

- *Autor: Félix Moreno Peñarrubia*

Solución:

$$2025 = 1^3 + 2^3 + \ldots + 9^3 = (1 + 2 + \ldots + 9)^2.$$

Consideremos hacer una biyección intermedia a parejas de números $((S_1, T_1), (S_2, T_2))$ con $S_i < T_i$.

Día 1: ADN

Biyección $(X, Y_1, Y_2, Y_3) \leftrightarrow ((S_1, T_1), (S_2, T_2))$:

$$f((X, Y_1, Y_2, Y_3)) = \begin{cases} ((Y_1, Y_2), (Y_3, X)) & \text{si } Y_1 < Y_2 \\ ((Y_3, X), (Y_2, Y_1)) & \text{si } Y_1 > Y_2 \\ ((Y_2, X), (Y_3, X)) & \text{si } Y_1 = Y_2 \end{cases}$$

Biyección $(S_i, T_i) \leftrightarrow Z_i$: codificamos las parejas de bits 00 y 11 como `G` y `T`. Las parejas 01 y 10 las asignamos a `A` y `C`, pero el primer 01 lo decidimos por la paridad.

Día 1: Vecindario

Resumen del enunciado

Colorea una cuadrícula $n \times m$ con el mínimo número de colores de forma que dos casillas a distancia menor que d sean de colores diferentes.

- *Autor: Catalin Covaci*

Solución:

- En general, $\chi(G) \geq \omega(G)$ para cualquier grafo. Aquí $\chi(G) = \omega(G)$.
- Si $d \leq \text{mín}(n, m)$, se tesela el plano con el clique.
- Si $d \leq \text{máx}(n, m)$, se tesela el plano teniendo en cuenta la distancia mínima en la dirección del lado más corto.
- Si $d > \text{máx}(n, m)$, se encuentra el clique más grande y se rellena lo restante de forma *greedy*.