Contest de Selección IOI 2022 - Solucionario

Federación Olímpica Peruana de Informática

5 de junio de 2022

Tutorial del problema: "3 Dígitos"

Autor(es): Racsó Galván

Grupo 1 (Complejidad Esperada: $Q \leq 720$)

Para resolver este grupo nos basta con consultar todos y cada uno de los números de 3 dígitos diferentes posibles.

Grupo 2 (Complejidad Esperada: $Q \leq 507$)

Para resolver este grupo podemos notar que si considerábamos la solución del grupo 1 pero asumiendo que los 3 dígitos diferentes formaban una secuencia creciente (ya que no importa el orden en las consultas) se lograba una cantidad de consultas de a lo mucho 125, así que una posibilidad es probar todas esas secuencias y al hallar alguna que dé `3` para luego probar con todas las permutaciones de estos 3 dígitos.

Grupo 3 (Complejidad Esperada: $Q \leq 16$)

Para resolver este grupo debemos notar que si definimos x_i como 1 si el i-ésimo digito es usado en el número oculto y 0 en caso contrario tendremos 10 incógnitas que hallar. Por teoría de álgebra lineal si tenemos 10 ecuaciones linealmente independientes (es decir, que ninguna sea una combinación lineal de las demás) podremos resolver el sistema. Luego de definir qué consultas realizar (la más simple es usar las consultas i, $i+1$, $i+2$ de manera cíclica desde $i = 0$ hasta $i = 9$) para formar la matriz, podemos aplicarle eliminación gaussiana en código (o en papel, ya que esta matriz es fija) para obtener los x_i con valor 1, que son necesariamente 3. Luego de ello probamos las 6 permutaciones posibles.

Si M es nuestra matriz propuesta y b el resultado de las consultas, tendríamos la siguiente relación:

$$Mx = b$$

Y si M es invertible (el sistema tiene solución):

$$x = M^{-1}b$$

Para la matriz propuesta como ejemplo, tendríamos algo como:

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}^{(-1)} = \frac{1}{3} \begin{pmatrix} 1 & -2 & 1 & 1 & -2 & 1 & 1 & -2 & 1 & 1 \\ 1 & 1 & -2 & 1 & 1 & -2 & 1 & 1 & -2 & 1 \\ 1 & 1 & 1 & -2 & 1 & 1 & -2 & 1 & 1 & -2 \\ -2 & 1 & 1 & 1 & -2 & 1 & 1 & -2 & 1 & 1 \\ 1 & -2 & 1 & 1 & 1 & -2 & 1 & 1 & -2 & 1 \\ 1 & 1 & -2 & 1 & 1 & 1 & -2 & 1 & 1 & -2 \\ -2 & 1 & 1 & -2 & 1 & 1 & 1 & -2 & 1 & 1 \\ 1 & -2 & 1 & 1 & -2 & 1 & 1 & 1 & -2 & 1 \\ 1 & 1 & -2 & 1 & 1 & -2 & 1 & 1 & 1 & -2 \\ -2 & 1 & 1 & -2 & 1 & 1 & -2 & 1 & 1 & 1 \end{pmatrix}$$

Como los valores finales solo son 0 o 1, podemos ignorar la fracción y simplemente verificar si el resultado final de cada x_i es 0 o algo diferente.

Otra forma de resolverlo es usando backtracking para hallar los x_i, ya que solo hay $\binom{10}{3}$ posibles soluciones y la solución es única.

Tutorial del problema: "Intersección Interna"

Autor(es): Racsó Galván

Para resolver este problema nos basta con notar que si tenemos dos segmentos (a, b) y (c, d) se deben cumplir algunas condiciones para que tengan intersección interna:

- Ninguno de sus extremos debe coincidir, es decir, no deben haber repetidos entre a, b, c y d.
- Podemos ver los extremos de los segmentos como paréntesis que abren y cierran, así que necesitaríamos una configuración como la siguiente:

  ([)]

  Donde () son los paréntesis del primer segmento y [] son los del segundo.

Podemos verificar lo anterior con las siguientes condiciones asumiendo que $a < c$:

- $|\{a, b, c, d\}| = 4$, para que no hayan repeticiones.
- $a < c < b < d$, para que haya al menos un punto de intersección interno.

Grupo 1 (Complejidad Esperada: $O\left(m^2\right)$)

Para resolver este grupo podemos probar con todos los pares de segmentos posibles y verificar que la condición de intersección se cumpla.

Grupo 2 (Complejidad Esperada: $O\left(nm\right)$))

Para resolver este grupo podemos agrupar todos los pares que tengan el mismo extremo izquierdo a y asignarnos a $L[a]$. Luego podemos iterar desde la posición 1 hasta la n y realizar el siguiente procedimiento:

- Primero, calculamos la respuesta en $O(n)$ para cada uno de los segmentos de $L[a]$.
- Luego, sumamos 1 a la posición del extremo derecho de cada uno de los segmentos de $L[a]$.

Por la forma en la que se actualiza, cuando lleguemos a cada posición habremos procesado todos los segmentos con extremo izquierdo anterior al nuestro, así que nos basta con realizar un escaneo lineal en el rango $(a, L[a]_i)$ para cada segmento.

Grupo 3 (Complejidad Esperada: $O\left(n + m \log n\right)$)

Para resolver este grupo podemos optimizar la solución del grupo 2 y usar alguna estructura de datos que nos permita modificar una posición y obtener suma en rango (Puede ser con Fenwick o Segment Tree) en $O(\log n)$.

Tutorial del problema: "Producto de Dígitos"

Autor(es): Racsó Galván

Grupo 1 (Complejidad Esperada: $O\left(n \log n\right)$)

Para resolver este grupo bastaba con iterar sobre todos los valores del 1 al n y verificar cuáles cumplían con la propiedad.

Grupo 2 (Complejidad Esperada: $O\left(\sum_{l=1}^{11}\binom{l+8}{8}\log n\right)$)

Para resolver este grupo vamos a aislar el caso en el que el producto de los dígitos es 0 (pues esto implica que hay un 0 en la representación del número), en dicho caso se debe dar la igualdad $x = b$, así que para contarlo nos basta verificar que $n \geq b$.

Por otro lado, notamos que si fijamos el producto x y los dígitos que usamos para formarlo podemos hallar el valor correspondiente de x y simplemente verificar que x conste de dichos dígitos. Para ello, fijaremos la cantidad de dígitos a usar y solo usaremos secuencias de dígitos **no decrecientes**, pues el orden de los dígitos se verifica luego de haber armado el producto, no antes. Esto se da porque 456 y 654 dan el mismo producto de digitos, y en este paso solo estamos interesados en el producto pero no en el orden especifico de cada digito. Recordemos que si queremos formar una secuencia de longitud l con valores entre 1 y 9 tenemos $\binom{l+9-1}{9-1} = \binom{l+8}{8}$ posibilidades.

Como l va desde 1 hasta 11, tendremos una complejidad de $O\left(\sum_{l=1}^{11}\binom{l+8}{8}\log n\right)$, pues nos toma $O(\log n)$ verificar que los dígitos correspondan al valor de $P(x) + b$.

Tutorial del problema: "Ambos"

Autor(es): Racsó Galván

Grupo 1 (Complejidad Esperada: $O\left(n^2\right)$)

Para resolver este grupo podemos fijar el extremo izquierdo de cada subarreglo y desplazar el extremo derecho con un doble `for`; si llevamos dos variables booleanas que nos digan si ya usamos 'x' y 'o', respectivamente, podemos calcular la respuesta en $O(n^2)$.

Grupo 2 (Complejidad Esperada: $O\left(n\right)$)

Para resolver este grupo debemos notar que, como son solo 2 símbolos, los subarreglos que no cumplen con la propiedad son aquellos que tienen todos sus elementos iguales. De esta manera, cuando tenemos una secuencia de k símbolos iguales consecutivos, esta generará $\frac{k(k+1)}{2}$ subarreglos inválidos.

Entonces, para hallar la respuesta nos basta con hallar todas las secuencias maximales de símbolos iguales y quitar $\frac{k(k+1)}{2}$ al contador, el cual debe estar inicializado en $\frac{n(n+1)}{2}$. Esto lo podemos hacer con un solo `for` o usando *two pointers*.

Tutorial del problema: "¿Hay camino?"

Autor(es): Racsó Galván

Grupo 1 (Complejidad Esperada: $O\left((n+m)q\right)$)

Para resolver este grupo nos basta usar un BFS o DFS para cada consulta, así obtendremos una complejidad de $O((n+m)q)$.

Grupo 2 (Complejidad Esperada: $O\left(\frac{nq}{B}\right)$, $B = 64$)

Para resolver este grupo debemos realizar algunas observaciones:

1. Ya que las aristas van de un nodo u a un nodo v con $u < v$, podemos definir $DP(u)$ como el conjunto de nodos alcanzables por u y calcularlo de la siguiente manera:

$$DP(u) = \bigcup_{(u,v)\in E} DP(v) \cup \{u\}$$

Ya que $v > u$, si procesamos los nodos de manera descendente podremos calcular el DP con en $O(n \cdot merge)$, donde $merge$ es la complejidad de unir los conjuntos.

2. Si agrupamos las consultas realizadas en bloques de tamaño B y coincidentemente tenemos una estructura que nos permita unir conjuntos de tamaño B en $O(1)$, podríamos resolver el problema en $O(\frac{nq}{B})$.

3. La estructura que nos permite unir conjuntos en $O(1)$ es la máscara de bits, así que tomaremos $B = 64$ y usaremos `unsigned long long` para representar los conjuntos de nodos que son alcanzables en cada bloque.

4. Para las B consultas a considerar en cada bloque, los únicos nodos alcanzables que nos interesan son los B_i correspondientes, así que el conjunto de $DP(u)$ tendrá el i-ésimo bit prendido si el B_j de la i-ésima consulta del bloque es alcanzable por u.

Con las observaciones anteriores y una implementación adecuada podremos pasar el TL y el ML sin problemas.