Olimpiada Peruana de Informática 2022
Fase 1 - Solucionario

Federación Olímpica Peruana de Informática

13 de noviembre de 2021

Tutorial del problema: "Parientes"

Autor(es): Alfredo de la Fuente

Supongamos que x es la suma de enteros positivos consecutivos.

$$x = \alpha + (\alpha + 1) + (\alpha + 2) + \cdots + (\alpha + k)$$

Agrupando terminos y simplificando,

$$x = \frac{1}{2}(k + 1)(2a + k)$$

para $k \geq 0$. Notamos que la suma de ambos numeros: $k+1$ y $2a+k$, resulta un numero impar de la forma $2(a + k) + 1$. Por lo tanto, uno de los numeros debera ser par y el otro impar. Luego podemos concluir que el numero de maneras de expresar x como la suma de uno o mas enteros positivos consecutivos, es igual al numero de divisores impares de x.

En la practica se podria calcular a partir de las decomposion por primos, o iterando para cada posible divisor impar.

Tutorial del problema: "Hamburguesas"

Autor(es): Alfredo De la Fuente

Analizamos los distintos casos:

1. Para números de la forma $9n$, son posibles para todo n.
2. Para números de la forma $9n + 1 = 9(n - 11) + 99 + 1 = 9(n - 11) = 25 \times 4$, $n \geq 11$.
3. Para números de la forma $9n + 2 = 9(n - 10) + 90 + 2 = 17 + 25 \times 3$, para $n \geq 10$.
4. Para números de la forma $9n + 3 = 9(n - 8) + 72 + 3 = 9(n - 8) + 25 \times 3$, para $n \geq 8$.
5. Para números de la forma $9n + 4 = 9(n - 7) + 63 + 4 = 9(n - 7) + 17 + 25 \times 2$, para $n \geq 7$.
6. Para números de la forma $9n + 5 = 9(n - 5) + 45 + 5 = 9(n - 5) + 25 \times 2$, para $n \geq 5$.
7. Para números de la forma $9n + 6 = 9(n - 4) + 36 + 6 = 9(n - 5) + 17 + 25$, para $n \geq 4$.
8. Para números de la forma $9n + 7 = 9(n - 2) + 18 + 7 = 9(n - 2) + 25$, para $n \geq 2$.
9. Para números de la forma $9n + 8 = 9(n - 1) + 9 + 8 = 9(n - 1) + 17$, para $n \geq 1$.

Por lo tanto, el mayor número ocurre en $9n + 1$ cuando $n = 10$. La respuesta es 91.

Tutorial del problema: "Caballo"

Autor(es): Alfredo de la Fuente

Notamos, en principio, que desde la casilla inicial $(1, 1)$ solo hay dos opciones (por medio de $(3, 2)$ y $(2, 3)$) para llegar al destino en caminos de dos saltos.

Por simetría, para los caminos mas largos podemos considerar solo un caso y multiplicar el resultado final por 2. Sin perdida de generalidad, tomamos como punto inicial la casilla $(2, 3)$, desde ahi debemos lograr un camino válido hasta $(4, 4)$ de distintas longitudes. Notamos que el camino deberá pasar

necesariamente por $(3, 2)$ antes de llegar a $(4, 4)$, lo que simplifica el problema a contar el número de caminos de $(2, 3)$ a $(3, 2)$, sin pasar por $(4, 4)$ o $(1, 1)$. Volvemos a notar simetría para este caso, siendo las únicas casillas siguientes a saltar $(3, 1)$ y $(4, 2)$. Simplificamos a contar el número de caminos de longitud a lo mucho 5 desde $(3, 1)$ hasta $(3, 2)$, sin pasar por $(2, 3)$, $(4, 4)$ y $(1, 1)$, el resultado será multiplicado por 4.

Una vez, en este punto, se puede evidenciar que solo hay caminos válidos de $(3, 1)$ a $(3, 2)$ por un número impar de saltos; es decir, para nuestro caso, 3 y 5. Para el caso más sencillo de longitud 3, solo hay dos posibles caminos $(3, 1) \to (1, 2) \to (2, 4) \to (3, 2)$ y $(3, 1) \to (4, 3) \to (2, 4) \to (3, 2)$. Similarmente, observamos para el caso de longitud 5 hay dos posibles caminos $(3, 1) \to (1, 2) \to (3, 3) \to (2, 1) \to (1, 3) \to (3, 2)$ y $(3, 1) \to (4, 3) \to (2, 2) \to (3, 4) \to (1, 3) \to (3, 2)$.

Finalmente, tenemos que el número total de caminos válidos es $4 \times (2 + 2) + 2 = 18$.

Tutorial del problema: "Carretera"

Autor(es): Walter Erquinigo y Alfredo de la Fuente

Grupo 1 (Complejidad Esperada: $O(1)$)

En este caso, F es 0, por lo tanto podemos poner cuantos puntos de acceso como queramos. Podemos, en efecto, poner uno en cada asentamiento y darle un radio de 0 a cada uno de ellos. Así logramos que todos los asentamientos estén cubiertos y el costo total sea 0. En conclusión, basta imprimir 0 en la respuesta.

Grupo 2 (Complejidad Esperada: $O(1)$)

Al ser V igual a 0, entonces podemos comprar un solo punto de accesso y darle un radio infinito, cubriendo así todos los asentamientos. En este caso, basta imprimir F como respuesta, que es el costo de un solo punto de acceso.

Grupo 3 (Complejidad Esperada: $O(2^n)$)

$n \leq 10$ nos dice que intentemos un backtracking. Pero antes tenemos que determinar cómo podemos elegir los intervalos. Una conclusión interesante a la que debemos llegar es que siempre existirá una solución en la que cada intervalo necesariamente comience y termine exactamente en asentamientos. En este tipo de soluciones, no se pueden encontrar intervalos que terminen en medio de dos asentamientos adyacentes. Podemos probarlo:

Imaginemos que tenemos una solución óptima en la cual el intervalo $[a, b]$ cubre al asentamiento en el punto x. Además, no hay ningún asentamiento en $]x, b]$. En otras palabras, x es máximo y $b <> x$. Si reducimos el intervalo a $[a, x]$, efectivamente cubrimos el mismo número de asentamientos y hemos reducido el costo en $(b-x)V$. Por lo tanto, esa solución inicial no era solución (contradicción). Finalmente, podemos hacer lo mismo para el lado izquierdo (punto a), y así demostramos que todos los intervalos deben comenzar y terminar en asentamientos.

Ahora lo que tenemos que hallar es una descomposición de los puntos en intervalos disjuntos con costo mínimo. Se deja como ejercicio para el lector realizar un backtracking para esto.

Grupo 4, 5 (Complejidad Esperada: $O(N^2)$)

Aquí se puede usar la programación dinámica clásica de intervalos. Esta se basa en hallar la solución comenzando en el asentamiento i y terminando en el $N-1$. Así, podemos tener subproblemas y la formula siguiente nos puede dar las soluciones parciales:

$dp(i) = \min_{j>=i} dp(j+1) + F + (X_j - X_i)V$ (nótese que el paso recursivo equivale a comprar un nuevo punto de acceso)

donde X_j es la posición del j-ésimo asentamiento.

Finalmente, la respuesta es $dp(0)$.

Grupo 6 (Complejidad Esperada: $O(Nlog(n))$)

Una solución se basa en el hecho de que si fijamos el numero de puntos de acceso a usar, digamos A, entonces quedarán $A-1$ espacios entre asentamientos adyacentes que no han sido cubiertos por los puntos de acceso. Por ejemplo, si $A = 2$, tenemos que vamos a cubrir puntos por la izquierda y por la derecha y dejaremos un espacio vacío. Si no dejaramos este espacio vacío, solo tendríamos un único intervalo, lo cual es incorrecto.

Ahora podemos reformular el problema a, dado un A, hallar el subconjunto de intervalos de asentamientos adyacentes de tamaño $A-1$ cuya suma es máxima. ¿Por qué se cumple esto? Veamos que la parte de la fórmula $2RV$ aplica a todos los asentamientos. O sea, si el total de espacio cubierto por todos los asentamientos es de tamaño T, el costo total es $AF+TV$. O sea lo que nos importa es el total de espacio cubierto, no cómo está distribuido. Si queremos minimizar el costo, tenemos que minimizar el espacio cubierto, que es lo mismo que maximizar el espacio no cubierto.

Para maximizar el espacio no cubierto, podemos crear un segundo arreglo

$D[i] = X[i+1] - X[i]$

que son las diferencias de asentamientos adyancentes. Si dedicidimos usar $A = 1$, entonces el costo total es $F - (X[N-1] - X[0])V$. Si decidimos usar $A = 2$, entonces solo necesitamos usar una de estas diferencias, que debería ser la máxima, y substraerla de $2F - (X[N-1] - X[0])V$. O sea, $2F - (X[N-1] - X[0] - D_{max})V$. Si $A = 3$, entonces debemos usar las dos diferencias máximas y substraerlas de la solución. Nótese que F siempre va multiplicado por A.

Así, podemos ordenar D de mayor a menor y probar con todos los posibles valores de A, quedándonos con el que minimiza el costo.

Tutorial del problema: "El laberinto más divertido"

Autor(es): Enrique Junchaya

Grupo 1 (Complejidad Esperada: $O(nm)$)

Debido a que hay un camino único entre cada par de casilleros, si recorremos el tablero por los casilleros vacíos comenzando desde la posición (0, 1), podemos hallar la distancia a cualquier otro casillero vacío del tablero. Esto es suficiente para escoger la mejor salida del laberinto, pues basta recorrer el borde del tablero y escoger el casillero lexicográficamente menor que es adyacente a un casillero vacío con distancia máxima.

Nota que nunca es posible escoger el casillero (0, 2) como salida porque eso haría que el laberinto deje de ser perfecto.

Grupo 2 (Complejidad Esperada: $O(nm(n+m))$)

La solución es idéntica a la del grupo 1, pero ahora debemos probar todos las $O(n+m)$ posibles entradas diferentes y manejar el caso borde en el que el tablero deja de ser perfecto de forma más general.

Grupo 3 (Complejidad Esperada: $O(nm)$)

El hecho de que haya exactamente un camino entre cada par de casilleros implica que podemos modelar el laberinto como un árbol. Entonces, es posible aplicar el algoritmo goloso estándar para hallar el diámetro del árbol, pero debemos tener dos consideraciones para que este sea un camino más largo válido en el laberinto:

- Ambos extremos del diámetro deben ser adyacentes a un borde del laberinto. Para conseguir esto, podemos preprocesar el laberinto y "tapar" todo los caminos que no llegan a un borde o modificar el algoritmo goloso para manejar bien esta restricción.

- Los extremos del diámetro deben estar junto a bordes lexicográficamente menores. Para conseguir esto, basta modificar el algoritmo goloso para que encuentre no solo el nodo a mayor distancia de un origen, sino, de entre estos, el que sea adyacente a un borde lexicográficamente menor.

Tutorial del problema: "Menor secuencia lexicográfica"

Autor(es): Racsó Galván

Grupo 1 (Complejidad Esperada: $O\left(\binom{n+m-1}{n-1} \cdot (n+m)\right)$)

Para resolver los casos de prueba de este grupo nos basta usar backtracking para considerar todos los caminos posibles y cada vez que encontremos un nuevo camino actualizamos la respuesta al comparar.

Ya que hay $O\left(\binom{n+m-1}{n-1}\right)$ caminos y cada uno de ellos tiene longitud $n+m-1$, entonces la complejidad será de $O\left(\binom{n+m-1}{n-1} \cdot (n+m)\right)$.

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

Para resolver los casos de prueba de este grupo bastaba con proponer un algoritmo que use programación dinámica definiendo la función:

$$memo(i,j) = \text{Menor secuencia lexicografica para llegar a la celda } (i,j) \text{ desde la } (1,1)$$

De esta manera, solo tenemos dos candidatos para una celda (i,j):

- Haber llegado desde la celda superior (si existe), en cuyo caso la secuencia será la menor posible de llegar a $(i-1,j)$ y luego pasar por (i,j) $(memo(i-1,j)+a_{i,j})$.
- Haber llegado desde la celda izquierda (si existe), en cuyo caso la secuencia será la menor posible de llegar a $(i,j-1)$ y luego pasar por (i,j) $(memo(i,j-1)+a_{i,j})$.

Entonces usamos la recursión:

$$memo(i,j) = \min\{memo(i-1,j), memo(i,j-1)\} + a_{i,j}$$

Donde $+$ es concatenación de vectores y consideraremos que $memo(0,x)$ y $memo(x,0)$ son una secuencia que consta solo del infinito ((∞), para que sea mayor lexicográfica que cualquier candidato).

Ya que hay $O(nm)$ posiciones a calcular y la comparación lexicográfica nativa de los lenguajes toma $O(Longitud)$, tendremos una complejidad de $O(nm(n+m))$.

Grupo 3 (Complejidad Esperada: $O\left(nm\right)$)

Para resolver los casos de prueba de este grupo debíamos notar que las posiciones que están a una misma cantidad de pasos de distancia desde la celda $(1,1)$ compiten entre ellos para ser el k-ésimo punto de paso en la secuencia óptima. Esto nos permite separar los candidatos por distancia a la celda $(1,1)$, la cual será $|x-1|+|y-1|$ pasos para la celda (x,y).

Luego, analizaremos el k-ésimo nivel de la siguiente forma:

- Comparamos todas las celdas marcadas como válidas de este nivel y obtenemos aquella celda (x,y) con $a_{x,y}$ menor. Agregamos $a_{x,y}$ al final del vector con la respuesta.
- Para todas las celdas válidas con $a_{x,y}$ igual al menor, marcamos las celdas a su derecha y abajo (si existen) como válidas para el siguiente nivel.

Al final del procedimiento se tendrá la respuesta correctamente procesada.

Ya que el separar por niveles a las celdas forma una partición de toda la matriz, visitaremos cada celda una sola vez, por lo cual tendremos una complejidad de $O(nm)$.