Gran Premio de México 2021 - Primera Fecha

August 28th, 2021

Libro de soluciones

Este documento contiene las soluciones esperadas para los 13 problemas usados durante la competencia.

Las siguientes personas apoyaron desarrollando el set de problemas ya sea creando o mejorando enunciados, soluciones, casos de prueba, verificadores de entradas y salidas:

Lina Rosales

Saraí Ramírez

Eddy Ramírez

Leonel Juarez

Alan Enrique Ontiveros

Abraham Macias

Emilio López

Fernando Fiori

Moroni Silverio

Juan Pablo Marín

Alien Crop Triangles

Por: Emilio López

Para solucionar el problema, primero es necesario calcular el área total a cubrir, esto se logra sumando las áreas de todos los triángulos de la entrada. Puede hacerse de forma sencilla utilizando la fórmula de Herón:

$$\text{area} = \sqrt{s\,(s-a)\,(s-b)\,(s-c)}$$

donde s es el semiperímetro:

$$s = \frac{a+b+c}{2}$$

Ya conociendo el área en m^2 a cubrir, pasamos a calcular los kg de semillas necesarios, redondeando para arriba. El enunciado indica que se necesita 1 kg cada 30 m^2, por lo que

$$\text{kg necesarios} = \text{ceil}\left(\frac{\text{area}}{30}\right)$$

Luego usamos una mochila (*knapsack*) para minimizar el costo de comprar los kg necesarios. Debemos elegir la opción más barata que tiene al menos los kg necesarios – puede ser que resulte más barato comprar kilos de más; no hace falta que sea exactamente la misma cantidad de kg calculada. Este costo será la solución que imprimiremos.

Para finalizar, debemos tener especial cuidado de tratar los casos bordes de la entrada correctamente, entre los que se destacan los siguientes dos:

- Si no hay ningún triángulo en la entrada, no es necesario comprar nada (salida 0).
- Si tenemos que comprar algo y las únicas bolsas disponibles están vacías (peso 0), no se podrá resolver el problema (salida -1).

Basel Problem

Por: Alan Enrique Ontiveros

Sea $f(z) = \dfrac{\pi \cot(\pi z)}{z^n}$ una función, donde $n \geq 2$. Esta función será útil para calcular $\zeta(n)$ usando el teorema del residuo, porque $f(z)$ tiene polos en cada entero.

Integrando $f(z)$ sobre un cuadrado muy grande centrado en el origen del plano complejo, con vertices en los puntos: $\{(N+1/2)(1+i), (N+1/2)(-1+i), (N+1/2)(-1-i), (N+1/2)(1-i)\}$, donde $N \to \infty$.

Se puede mostrar que: $\oint_\Gamma f(z)dz = 0$. Por el teorema del residuo esa integral es:

$$\oint_\Gamma f(z)dz = 2\pi i \sum_{k=-\infty}^{\infty} Res_{z=k} f(z)$$

Comparando ambas expresiones, tenemos que:

$$\sum_{k=-\infty}^{\infty} Res_{z=k} f(z) = 0$$

Vemos que $f(z)$ tiene polos de orden 1 en cada entero diferente de 0, esto es, si $k \neq 0$ es un entero, entonces:

$$\begin{aligned}
Res_{z=k} f(z) &= \lim_{z\to k} f(z)(z-k) \\
&= \lim_{z\to k} \frac{\pi \cot(\pi z)(z-k)}{z^n} \\
&= \lim_{z\to k} \frac{\pi \cos(\pi z)(z-k)}{\sin(\pi z) z^n} \\
&= \lim_{z\to k} \frac{\pi \cos(\pi z)}{z^n} \cdot \lim_{z\to k} \frac{z-k}{\sin(\pi z)} \\
&= \frac{\pi \cos(\pi k)}{k^n} \cdot \lim_{z\to k} \frac{1}{\pi \cos(\pi z)} \\
&= \frac{\pi \cos(\pi k)}{k^n} \cdot \frac{1}{\pi \cos(\pi k)} \\
&= \frac{1}{k^n}
\end{aligned}$$

Como n es un número par entero positivo, tenemos que:

$$2\sum_{k=1}^{\infty} \frac{1}{k^n} + Res_{z=0} f(z) = 0$$

$$\zeta(n) = -\frac{1}{2} Res_{z=0} f(z)$$

Entonces, encontrar $Res_{z=0} f(z)$ es lo mismo que encontrar el coeficiente de z^{-1} en la serie de Laurent para $f(z)$, o de manera equivalente, encontrar el término constante en $\dfrac{\pi z \cot(\pi z)}{z^n}$.

Finalmente para encontrar todos los valores de $\zeta(n)$ al mismo tiempo, podemos observar que $\zeta(n)$ es el coeficiente de z^n en la serie de Taylor: $-\dfrac{1}{2}\pi z \cot(\pi z)$.

Sabemos que $\zeta(n) = \dfrac{p_n}{q_n}\pi^n$, por lo que:

$$\sum_{\substack{n=0 \\ n \text{ even}}}^{\infty} \zeta(n) z^n = \sum_{\substack{n=0 \\ n \text{ even}}}^{\infty} \frac{p_n}{q_n} (\pi z)^n = -\frac{1}{2}\pi z \cot(\pi z)$$

Sustituyendo $x = \pi z$, tenemos que:

$$g(x) = \sum_{\substack{n=0 \\ n \text{ even}}}^{\infty} \frac{p_n}{q_n} x^n = -\frac{1}{2} x \cot(x)$$

Observemos que $g(x) = -\frac{1}{2}x\cot(x) = -\frac{1}{2}\cdot\frac{\cos(x)}{\sin(x)/x}$ es el cociente de dividir dos series de Taylor, y es la función que genera todas las respuestas que buscamos.

Podemos extraer los primeros n coeficientes de manera eficiente usando FFT (NTT con el módulo $119\times 2^{23}+1$) tomando la convolución de $\cos(x) = \frac{1}{0!} - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 - \frac{1}{6!}x^6 + \cdots$ con el inverso multiplicativo de $\sin(x)/x = \frac{1}{1!} - \frac{1}{3!}x^2 + \frac{1}{5!}x^4 - \frac{1}{7!}x^6 + \cdots$ (el inverso se puede calcular usando el método de Newton-Raphson) con una complejidad de $O(n\log n)$ si ignoramos los coeficientes después de x^n de ambas series.

Cypher Decypher

Por: Eddy Ramírez

Este es un problema clásico. Dado que el valor de los números nunca será mayor a 10^6, se puede usar una criba para calcular los números primos. Una vez que se tenga la criba C podemos saber si un número es primo si $C[x] = 1$ y no es primo si $C[x] = 0$. Ahora, si para cada valor i, j dado en la entrada, contamos cuantos valores $C[x] = 1$ tales que $i \leq x \leq j$ estaremos encontrando la respuesta correcta, pero, la iteración agregada hará que el envío sea rechazado por exceder el tiempo límite de ejecución.

Podemos crear un nuevo arreglo S tal que $S[x] = S[x-1] + C[x]$, de esta manera podemos observar que $S[x]$ tiene el valor acumulado de cuántos números primos hay entre 1 e i (la suma de cuántos $C[i] = 1$ hasta i). Usando este arreglo, podemos encontrar la cantidad de primos que hay entre i y j simplementa calculando $S[j] - S[i-1]$.

Delivering Pizza

Por: Moroni Silverio

La mejor manera de solucionarlo es un ingrediente a la vez ignorando los demás empezando por el primer ingrediente, al leer la entrada sabemos cuáles serán los únicos que pasarán por ahí y deben atenderse según el orden de la entrada.

Para los demás ingredientes, de igual manera enfocarse en el j-ésimo ingrediente si los ingredientes anteriores han sido resueltos, ya que no se agregarán más pedidos a ese ingrediente después. Lo que se debe tener presente es el momento en que la persona atendiendo el ingrediente J se desocupa, una vez que lo hace tiene a elegir a todos los pedidos que llegaron antes de que se desocupara. Si no llegaron pedidos antes de eso, debe escoger el primer pedido que llegará después de que se desocupe. Esto se hace con dos colas de prioridad, una para llevar el control de los pedidos que llegaron antes de que se desocupara y otra para llevar el control de los pedidos que llegan después de que se desocupe.

Escape Room

Por: Moroni Silverio

Podemos observar que se puede dar la respuesta a cualquiera de las preguntas tales que la coordenada no sea un '.' en el mapa con solo observar el mapa. Entonces, nos interesa encontrar cómo responder las preguntas que se hagan a estas coordenadas: El problema de encontrar el camino más corto entre dos puntos en un mapa como el entregado en la entrada es bien conocido que puede ser resuelto aplicando una búsqueda en amplitud, entonces, una posible solución al problema sería para cada consulta realizar la búsqueda en amplitur partiendo de esa coordenada para encontrar el camino más corto hasta la salida, en caso de que exista algún camino. Esta solución si bien es correcta, es muy lenta con la cantidad de consultas que se podrían dar como entrada, y sería rechazada por exceder el tiempo límite de ejecución.

Una característica de la búsqueda en amplitud es que no solo encuentra el camino más corto partiendo de un nodo origen a un destino, sino que encuentra el camino más corto del nodo origen a cualquier otro nodo que es alcanzable en el mapa partiendo del origen, podemos observar que el camino más corto de una posición del mapa a la salida, es el mismo que el de la salida hacia esa posición, entonces, podemos calcular el camino más corto de cualquier posición del mapa a la salida realizando una búsqueda en amplitud que tenga como origen la salida del mapa. Es importante tener en cuenta las prioridades con las que se debe responder en caso de que un punto tenga más de un camino con la misma distancia a la salida. Una vez realizado este precómputo, se puede calcular cada respuesta en $O(1)$.

Fixing Subtitles

Por: Moroni Silverio

Para solucionar el problema se tiene que "parsear" la entrada para separar los subtítulos del tiempo en el que comienzan. Teniendo esa información y el tiempo correctamente "parseado", solo hay que sumar para obtener la nueva hora. Todas las líneas que no tienen una hora deben ser impresas como se reciben.

Se deben considerar casos donde el delay sea 0.

Game of Baker

Por: Lina Rosales

HeatWave

Por: Fernando Fiori

Primero describo una solución para un tamaño de w fijo y luego se puede extender a cualquier tamaño < 200.

Observación 1) quizás no se pueden reemplazar todas las ocurrencias de una palabra porque luego puede no ser reversible la operación, se pueden reemplazar ocurrencias que no se superpongan. Ejemplo: T=000, w=00, la operación correcta de reemplazos devuelve T'=20 o T'=02, pero nunca T'=22.

Observación 2) Si conozco todas las ocurrencias de una palabra w en T, una forma óptima de elegir los reemplazos es elegir la primera ocurrencia de w (recorriendo T de izquierda a derecha), saltear $|w|$ caracteres, y repetir el proceso. No hay una manera que comprima más el texto que ésta. (Demostración, sea i el índice de la primera ocurrencia de w en T, y j el índice de la siguiente ocurrencia, $i < j$. Supongamos que no elijo i como reemplazo, pero sí j. Luego puedo seguir eligiendo ocurrencias de w en T que empiecen en $j + |w|$, o sea en $S = T_{j+|w|}...T_{|T|-1}$. Pero si hubiésemos elegido la ocurrencia de i, entonces podríamos seguir eligiendo ocurrencias en $R = T_{i+|w|}...T_{|T|-1}$. Pero S está contenido en R, y nunca vamos a encontrar más ocurrencias en un pedazo de texto que en el texto completo, ya que puedo omitir esos primeros j-i caracteres. Con lo que elegir j en vez de i no nos da una mejor solución.)

Ahora bien, si buscamos una $|w|$ de un tamaño dado que nos maximice la compresión de T, podemos ir recorriendo T hasheando la ventana de tamaño w. Podemos usar hash polinomial ya que el tamaño máximo de la cadena es mayor que 64 bits. Vamos manteniendo un hashmap con hash de cadena como key, y como value un par cantidad de ocurrencias occ, último índice donde se la vio $last$. De esta manera si $last < indice\_actual - |w|$, entonces hago $occ++$ y $last = indice\_actual$.

Una vez que termino de recorrer T reviso cada entrada del map y me quedo con la palabra de mayor cantidad de ocurrencias.

Realizo este proceso para cada tamaño posible de w (de 2 a b) y me quedo con la mejor.

Luego es solamente hacer una cuenta y mostrar el resultado.

La complejidad final es $O(|T| * b)$, sin contar el costo de acceder a valores del hashmap.

Introducing Teleporting Machine

Por: Moroni Silverio

Para obtener el timepo sin la maquina teletransportadora basta con sumar $ciudad[n] - ciudad[i]$ para toda i.

Ahora, para calcular el tiempo total con la maquina conectando las ciudades i y j necesitamos:

- El tiempo total sin maquina T.
- Lo que ahorramos en tiempo D: el numero de ciudades que usaran la maquina por la distancia que se ahorrara que es la distancia entre i y j
- El costo de usar la máquina C: El numero ciudades que hay hasta i por el costo de usar la maquina

Entonces, cuando la máquina se pone entre las ciudades i y j, el costo será $T_{i,j} = T - D + C$, se debe encontrar el i, j que minimiza el valor de $T_{i,j}$. Esto se puede lograr iterando sobre cada ciudad i donde se puede poner la máquina y establecer j como la ciudad más lejana que está a una distancia menor o igual a los K kilómetros máximos que puede conectar la máquina de transportación, encontrar la ciudad j para cada valor posible de i se puede lograr con una busqueda binaria, y entonces el algoritmo para encontrar los valores i, j que minimizan $T_{i,j}$ es $O(NlogN)$.

Just Send the Email

Por: Abraham Macías

Sean los empleados nodos y las relaciones manager-subordinado aristas, la estructura de la organización de la compañía forma un árbol enraizado en 1. En este problema se pide hallar el valor esperado de la distancia entre un nodo aleatorio u del árbol a la hoja más cercana. Notar que de un nodo u se puede mover a cualquiera de sus hijos o al padre, así que se podría considerar un árbol no dirigido.

Para encontrar la respuesta, se puede hallar la suma de las distancias más cortas a una hoja desde cada nodo. Para hallar la distancia más corta de un nodo a la hoja más cercana, se puede hacer una BFS (búsqueda en amplitud) multi-origen desde todas las hojas hacia los demás nodos (hay que meter al inicio todas las hojas a la cola y después ejecutar una BFS). La distancia inicial de cada hoja es 1.

Después de hallar la suma de las distancias, hay que multiplicarlo por el inverso multiplicativo modular de n. Para hallar el inverso modular se puede usar el pequeño teorema de Fermat o el algoritmo extendido de Euclides.

Kids at the Party

Por: Saraí Ramírez

Se puede observar que para que una cantidad M de niños asistan a la fiesta, es necesrio que $N \ mod \ M = 0$. Es decir que M sea un divisor de N. Dado que Jaime quiere que su amigo Churro siempre asista y que no puede haber más de 6 niños en la fiesta, entonces el valor de M cumple que $2 \leq M \leq 6$. Entonces, para encontrar las diferentes cantidades de niños que pueden asistir a la fiesta será necesario encontrar todos los números M con las restricciones dadas que sean divisores de N. Un inconveniente es que el valor N puede ser tan grande que no se puede almacenar en un entero, entonces, se debe encontrar una estrategia que nos permita identificar los posibles valores de M a pesar del tamaño del número. Una manera de realizar esto es aplicar criterios de divisibilidad:

Se pueden utilizar criterios de divisibilidad de los números 2,3,4,5,6 para identificar los posibles valores de M dado N:

- 2 es un divisor de N si el último dígito de N es par.
- 3 es un divisor de N si la suma de los dígitos de N es un múltiplo de 3
- 4 es un divisor de N si los últimos dos dígitos de N son un múltiplo de 4
- 5 es un divisor de N si el último dígito de N es 5 o 0.
- 6 es un divisor de N si 2 y 3 son divisores de N.

Leonel and the powers of two

Por: Leonel Juárez

La notación que se le pide a Leonel, no es en realidad inventada por su profesor. Es como tal la función recursiva que se utiliza para hacer realizar el cálculo de un exponente con el algoritmo conocido como exponenciación binaria: `https://es.wikipedia.org/wiki/Exponenciaci%C3%B3n_binaria`.

El problema entonces se reduce a implementar la función recursiva que define a la notación que se le pidió a Leonel, teniendo cuidado de no dejar espacios en blanco y de que se sigue la impresión según las 3 reglas definidas.

Moon Dancers

Por: Juan Pablo Marín

Una característica importante a notar en el problema, es que, debido a que todos los danzantes se sentarán siempre en una posición tal que el ángulo es entero, entonces hay exactamente 359 posibles posiciones para un danzante que se levanta para rotar hacia otro danzante, además como todos los danzantes que se levantan rotarán el mismo ángulo, podemos pensar en una estrategia para resolver el problema en el que encontremos el máximo número de parejas que se pueden hacer si los danzantes rotaran una cantidad fija R, y encontrar el máximo entre los 359 valores que puede tomar R.

Dado R, podemos observar que el danzante i podrá ser pareja con el danzante j, si $(i + R)\ mod\ 360 = j$. Definamos un grafo dirigido G, donde los vertices son cada una de las posiciones donde hay un danzante, y existe una arista que sale de la posición i inicial de un danzante a la posición j si y solo si hay un danzante en la posición j y el danzante de la posición i puede llegar al danzante j al hacer la rotación R. Dado G, podemos visualizar el problema como colorear la máxima cantidad de nodos que nos sea posible usando dos colores A y B , de modo que si dos nodos u, y v están coloreados y hay una arista que los une, entonces u está coloreado del color A y v es de color B. Para realizar esta tarea observemos que por las características que tienen las rotaciones, cada vertice tendrá a lo más una arista de salida y una arista de entrada para cada una de las rotaciones posibles, entonces, podremos verificar que no habrá cadenas ni ciclos que se intersecten en G, así, si tomamos una componente conexa de G podríamos "estirarla"de modo que se forme una línea, debido a esta característica, podemos ver que si la componente conexa tiene C_v nodos de G, entonces, se pueden hacer $\lfloor \frac{C_v}{2} \rfloor$ parejas de esa componente. Entonces, para contar cuántas parejas se pueden hacer en una rotación, basta con sumar la cantidad de parejas que se pueden realizar por cada componente del grafo. La respuesta al problema será el valor más grande que se obtenga después de obtener la respuesta para cada una de las 359 rotaciónes posibles.