Gran Premio México 2018 Fecha 2

Descripción de soluciones

A - Amusement Park

Se modela un espacio de búsqueda en el que cada estado está dado por la existencia del imán para refrigerador , le existencia del llavero y la posición actual de John.

Entonces, el estado inicial sería *(0,0 (X,Y))* donde *X*, *Y* son la posición inicial de John y el estado final es *(1,1, (X',Y'))* donde *X'* y *Y'* son la posición de la salida.

En este espacio de búsqueda el camino más corto para que John obtenga los dos regalos es el que se da haciendo una BFS del estado inicial al final, considerando que cada que se llega a una tienda al salir de ella se tendrá el objeto que la tienda vende y que la primera vez que se llegue al estado *(1,1, (X', Y'))* se habrá encontrado el camino más corto para llegar a la salida con los dos objetos.

B - Bakers Gang

Para que sobre un nugget, el valor de K debe ser una unidad mayor a un múltiplo de N esto es K = xN + 1.
Sea K_i el número de nuggets que hay cuando el i-ésimo gato toma su parte entonces el gato tomará
$K_i/N + 1$ nuggets dejando en total $K_{i+1} = (N-1)(K_i-1)/N$ nuggets.
Para que cada que un gato tome su parte el resultado también tenga un nugget como excedente se debe cumplir que:
$K_i/N + 1$ (la cantidad que toma) sea múltiplo de N. Esto sucede cuando el valor x es yN-1.
Así sabemos que $K_0 = (y_0N-1)(N)+1$ y $K_i = (y_iN-1)(N) + 1$. Resolviendo para calcular el valor de y_i
$K_i = (y_iN-1)(N) +1 = y_iN^2-N+1$.
$K_{i+1} = (N-1)(K_i-1)/N$ sustituyendo K_i $K_{i+1} = (N-1)(y_iN^2-N+1-1)/N = (N-1)(y_iN^2-N)/N = (N-1)y_iN^2/N - N +1$
Sea $y_{i+1} = (N-1)y_i/N$ entonces $K_{i+1} = y_{i+1}N^2 -N + 1$.
Resolviendo la recurrencia se puede observar que $K_i = (N-1)^i N^2y_0/N^i - N + 1$. El valor y_0 más pequeño con el que se pueden obtener N+1 elementos en la serie de los valores de K tales que su residuo es 1 es cuando $y_0 = N^{N-1}$. Entonces

$$K_0 = (N-1)^0N^2N^{N-1}/N^i - N + 1 = N^{N-1+2} - N + 1 = N^{N+1} - N + 1$$

Es importante para que se ejecute en el tiempo límite el que se utilice exponenciación binaria para calcular el término N^{N+1} , esto es, usar la siguiente recursión para calcularlo:

$a^b = a^{b/2} * a^{b/2}$ si b es par
$a^b = a^{b/2} * a^{b/2} * a$ si b es impar

C - Cables

La idea de este problema se tomó del problema C del NEERC 2001 (Cable Master)

Supongamos que hay un L tal que $sum(floor(A_i/L)) >= K$ donde floor es la función piso y sum la suma de todos los valores y además A_i es la longitud de cada cable en el stock entonces posiblemente haya un L más grande que cumpla la restricción, en caso contrario L debe ser más pequeño.

El valor de L más grande que cumpla con la restricción mencionada es la solución. Por las características de esta función este valor se puede encontrar con una búsqueda binaria sobre el valor de L.

D - Divisible repunit

Hay que observar que por la cantidad de dígitos que puede tener el repunit se puede hacer overflow muy rápido si se intenta generar cada repunit y probar si es divisible por *N*. Sin embargo un método se puede aplicar si mantenemos solo el residuo del repunit de *K* cifras con respecto a *N*.

Supongamos que el repunit de *K* cifras módulo *N* es igual a *R* entonces el residuo R' del repunit de *K+1* cifras será:

$$R' = ((R*10) \ \%N + (1 \ \% \ N)) \ \% \ N$$

Se parte del repunit de *1* cifra donde *R=1* para cualquier *N > 1* o *0* si *N = 1*. Y se calcula *R'* hasta que *R' = 0*.

¿Cómo saber si no existe solución? En caso que una solución no existiera se llegará a un *R'* que ya se había calculado, en cuyo caso se generaría un ciclo en los residuos.

E - Examining groups

Este es un ejemplo clásico de range queries offline, offline se refiere a que no se hacen actualizaciones en la información una vez que se comienzan a hacer consultas. Hay varias soluciones que se pueden implementar para obtener el YES en este tipo de problemas:

- BIT
- Segment Tree
- Algoritmo de Moh.

Este problema en particular es muy similar al clásico D-Query.

F - Frogs secret meeting

La idea de este problema se tomó del problema B del NWERC 2007 (March of the penguins)

Se genera una red residual con las restricciones dadas. Para modelar esta red se usan dos nodos por piedra que se tiene en la entrada, llamamos a estos el nodo de entrada para la piedra *i* y el nodo de salida de la piedra *i* y la red se modela como sigue:

- Una arista con capacidad igual al número de ranas que hay en la piedra *i* de la fuente de la red al nodo de entrada de la piedra *i* .
- Una arista con capacidad igual al número de saltos que se pueden hacer para que las ranas no se hagan sospechosas del nodo de entrada de la piedra *i* al nodo de salida de la piedra *i*.
- Una arista con capacidad “infinita” del nodo de salida de la piedra *i* al nodo de entrada de la piedra *j* si la distancia de la piedra *i* a la piedra *j* es menor o igual que *D*.

Tomando cada una de las piedras que hay en el estanque como el desagüe de la red, si el flujo máximo en esta red de la fuente a esta piedra es igual al número total de ranas entonces en esa piedra se pueden reunir todas las ranas.

G - Generator strings

Hay que observar que si las cadenas que resultan de ordenar los caracteres de dos cadenas *S* y *S'* son iguales entonces *S* y *S'* entonces *S* es generadora de ambas y *S'* es generadora de ambas.

Entonces, para contar cuantos pares *A,B* hay tales que *A* genera a *B* se ordena cada una de las cadenas, y se cuentan el número de pares que es la suma de los cuadrados de cuántas veces aparece cada cadena diferente en la lista de las cadenas ordenadas.

H - Hiding messages

Un problema simple de rotación en candenas.

La solución es aplicar simplemente la operación que se pide para cada valor en la entrada que sea una letra del alfabeto, solo hay que tener cuidado con la rotación dado que después de la Z sigue la A y de la z la a.

Para no tener muchos problemas con esto se codifica la letra a un valor entero entre 0 y 26, se le suma K, se aplica modulo 26 al resultado y este residuo se traduce a la letra que corresponde de salida, asegurando que si la letra de entrada era minúscula regresar la del mensaje codificado en minúscula también.

I - Intergalactic tourism

Por el número de nodos tan pequeño es fácil intuir que generar todas las permutaciones y revisar las restricciones en cada permutación debe correr en tiempo.

Para no agregar un factor de complejidad al revisar las restricciones se pueden codificar las restricciones de cada planeta *i* en un número entero usando máscaras de bits, esto es, en número donde el bit *k* está encendido si *k* está en la lista de planetas que deben haber sido visitados antes que *i*. Llamemos esta lista *B* .

Ahora al evaluar cada permutación la permutación es válida y deberá imprimirse si y solo si K_i & *B[i] = B[i]* donde K_i es una máscara de bits con los bits encendidos de los planetas que están de la posición *0* a la *i* en la permutación.

J - Joy and misery

La solución trivial es usar contador *k* para cada rango *X, Y* dado en la entrada el cual inicia en *X* y termina en *Y* . Si k es impar se aumenta el número de impares que hay en el rango, si k es par se aumenta el número de pares que hay en el rango. Al final imprimir el número de impares y el número de pares contados.

Una solución más eficiente en tiempo es observar que se puede calcular solo revisando la paridad de X y Y.

- Si X es impar y Y es par o si X es par y Y es impar entonces la cantidad de impares y de pares es la misma y es *(Y - X + 1) / 2*
- Si X es impar y Y es impar habrá un impar más que pares entonces hay *(Y - X + 1)/2 + 1* impares y *(Y - X +1) / 2* pares. Lo mismo pasa si X es par y Y es par.
- Si X es par y Y es par habrá un par más que impares entonces hay *(Y - X + 1) / 2* impares y *(Y - X + 1)/2 + 1* pares