Gran Premio México 2018 Fecha 1

Descripción de soluciones

A - Permuting and adding up

- Para resolver la primera parte del problema (la cantidad de números distintos con los mismos dígitos que N)sea c_k la cantidad de veces que aparece el dígito k en N, entonces hay :
  $(c_0 + c_1 + c_2 + c_3 + ... + c_9)!/(c_0! * c_1! * c_2! * ... * c_9!)$ - Permutaciones con repetición
- Para la segunda parte (la suma de todos los diferentes números) . Llamemos V_k a la cantidad de veces que aparece el dígito k en un lugar específico dentro de las permutaciones de la cifras del número N. Llamemos S_k la suma con que contribuye el dígito k con el resultado. Con estas definiciones el resultado final debe ser : $\sum^{9}_{k=0} S_k$.
- Calculando S_k El dígito k aparece V_k veces en la cifra de las unidades, V_k veces en las decenas, V_k veces en las centenas, … Por lo tanto: $S_k = kV_k + 10kV_k + 100kV_k + ... = kV_k(111...)$ Donde la cantidad de unos es igual a la cantidad de cifras de N.
- V_k es igual a la cantidad de permutaciones que se pueden hacer con los dígitos del número N quitándole una cifra de valor de k. Entonces por la parte 1 del problema tenemos que:
  $V_k = (c_0 + c_1 + c_2 + ... + (c_k - 1) ... + c_9)!/(c_0! * c_1! * c_2! * ... * (c_k - 1)! * ... * c_9!)$

B - Sleeping Baker

Sean *i, j* un par de filas en la matriz dada *M* tal que $i \neq j$. La cantidad de rectángulos que se pueden hacer usando estas dos filas son $(x) * (x-1) / 2$ .Donde *x* es el número de elementos *k* que satisfacen $M(i,k) = M(j,k) = 1$. El total de rectángulos entonces es la suma de $(x) * (x-1) / 2$ de cada par de filas *i,j* tales que $i < j$.

Una implementación trivial en $O(n^3)$ revisar cada valor k en cada par de filas recibiría un TLE. Para obtener el YES se hace una amortización a O(n^2) utilizando cada fila como un array de enteros codificando cada fila en $M/64$ valores donde *M* es el número de columnas. O en su defecto utilizar una implementación existente en el API del lenguaje como es el caso de BitSets.

C - Counting weak RNA sequences

- Se hace un hash de cada subcadena de longitud 10 siempre y cuando solo contenga las letras A y T y que aparezca al menos una vez cada una de ellas. El hash se forma como la representación binaria de la cadena donde A = 0 y T = 1, así la cadena AAAAATTTTT se codifíca en el número 0000011111. En un arreglo *C* que pueda almacenar los posibles 2^10 hashes diferentes se cuenta cuántas veces aparece cada cadena.
- Utilizando el hash descrito se puede calcular la cadena complementaria de otra cadena haciendo la inversión de los bits, lo cual se puede lograr negando el número y aplicando la operación lógica AND a la negación con el número binario 1111111111 ((~x) & 1023).
- Así el número de hashes totales es la suma de *C[i]* para todo i de 1 a 1023 tal que *C[i] > 0* y *C[complemento[i]] > 0*

D - Dividing hexadecimal numbers

- 2 Opciones O(D) donde D es el número de cifras del número dado.
- Opción 1:
  - Iniciar con un contador en 0.
  - En cada paso agregar el siguiente dígito

    cont = cont*16 + (cont + A[i])
  - Calcular el residuo:

    cont = cont % 17;
  - Si al final el residuo es 0 entonces el número es múltiplo de 17.
- Opción 2: Similar a base 10 donde los múltiplos de 11 se pueden identificar cuando la resta de la suma de los dígitos en posiciones impares y la suma de los dígitos en posiciones pares es múltiplo de 11, en base 16 los múltiplos de 17 cumplen con la misma propiedad.

E - Exact sum of squares

- (1) Según el teorema de fermat sobre la suma de cuadrados se tiene que cualquier primo p tal que $p = 2$ o $p = 1 \mod 4$ entonces p puede ser representado como la suma de dos cuadrados.
- (2) Supongamos que un número primo p de la forma $4k + 3$ en la factorización de N y supongamos que $N = a^2 + b^2$ entonces p aparece una cantidad par de veces en la factorización de N(se omite la demostración por espacio, se invita a que el lector las realice)
- (3) Si dos números x, y se pueden escribir como suma de dos cuadrados, entonces su producto también se puede escribir de esta forma. Esto es por la identidad:
  $(a^2 + b^2)(c^2 + d^2) = (ac-bd)^2 + (ab+bc)^2$
- Se tiene entonces que, si la factorización de N todos los números primos de la forma $4k+3$ tienen un exponente par o el número solo tiene primos de la forma $4k+1$ y el 2 por la identidad en (3) el número se puede expresar como suma de dos números cuadrados.

F - Finding the train

El problema se remonta a encontrar la cadena palíndromo más grande de longitud par que existe en S . Para esto cualquier algoritmo que lo encuentr en O(n) es suficiente, con los límites establecidos incluso soluciones en orden O(NlogN) son aceptados.

Se presentan tres posibles opciones de solución en O(N):

- Usar hash y encontrar la subcadena común más grande entre S y S' donde S' es S revertida que inician en el mismo indice.
- Crear un Suffix tree generalizado con S y S' y encontrar la subcadena común más grande que inician en el mismo índice.
- Usar el algoritmo de Manacher.

G - Gambusines

- Crear una matriz M binarizada donde M(i,j) = 1 si la matriz dada en la entrada tiene el valor 1 en la fila i y columna j. M(i,j) = 0 en caso contrario.
- Encontrar todas las componentes conexas de M usando una técnica de flood fill donde M(i,j) = 1 y contar el número de celdas de cada componente.
- Imprimir el número de componentes tales que su número de celdas es mayor a T y la suma del beneficio de explotar estas componentes.

H - Harder sokoban

Se modela un espacio de búsqueda donde cada uno de los estados son las coordenadas en donde está cada una de la ubicaciones de las cajas. Así el estado inicial es el que está dado por las posiciones iniciales de las cajas, el estado final es cualquiera de los estados en el que cada caja ‘B’ se encuentra en un ‘G’, al no haber una correspondencia directa de ‘B’ y ‘G’ hay dos alternativas para encontrar el mínimo número de movimientos:

- Hacer un etiquetado de que G correspondería a cada B y hacer una BFS del estado inicial al del etiquetado que se haya hecho, la menor cantidad de movimientos será el del etiquetado que minimiza el número de movimientos de la BFS.
- Hacer una BFS sin un etiquetado previo, en cada estado al que se llegue, revisar si cada una de las cajas está en algún G, el número de pasos dados en el árbol de búsqueda del primer estado al que se llegue que satisface esta condición será el número mínimo de movimientos para acomodar las cajas.

En las alternativas BFS se refiere a una búsqueda en amplitud, un par de estados *X*, *Y* son adyacentes para continuar la búsqueda si *Y* contiene la misma configuración de *X* excepto que una caja ha sido movida con un movimiento válido según las restricciones descritas en el problema

I - Is a triangle

Un número *N* es triangular si existe un número *x* tal que la suma de los números enteros del *1* al *x* es igual a N.

La suma de los números de 1 a x se puede expresar como *(x)(x+1)/2* . Entonces *N* será triangular si existe un x tal que *(x)(x+1)/2 = N* .

Elaborando un poco la igualdad anterior:

- (1) . $(x^2 + x)/2 = N$
- (2) . $x^2 + x = 2N$
- (3) . $X^2 + x - 2N = 0$

Si al resolver la ecuación (3) es una solución entera para *x* entonces *N* es triangular. Otro método es usar búsqueda binaria para encontrar si existe un valor de *x* que satisface a (2).

J - Joining cells

- Se precomputan la suma de todos los rectángulos dados que parten de la coordenada *0,0* y terminan en *i,j* usando programación dinámica y el teorema de inclusión y exclusión, llamemos a la matriz que lo precomputa *DP*, y a la matriz de entrada *M* :

  $DP(i,j) = DP(i-1,j) + DP(i,j-1) - DP(i-1,j-1) + M(i,j)$
- Usando *DP* se puede encontrar la suma de cualquier rectángulo dado por las coordenadas (L_x, L_y) y (U_x, U_y):

  $S(L_x,L_y,U_x,U_y) = D(U_x,U_y)-D(U_{x-1},U_y)-D(U_x,U_{y-1})+D(U_{x-1},U_{y-1})$
- Con esta notación se busca cuántos cuadrados que inician en su esquina superior izquierda L_x, L_y que terminen en su esquina superior derecha U_x, U_y tales que $U_x - L_x = U_y - L_y$ y $S(L_x,L_y,U_x,U_y) >= X$.
- Para encontrar este valor, fijemos un valor de U_x, U_y se puede distinguir que la función S aumenta a medida que U_x, U_y tengan el mismo valor y L_x, L_y disminuyen ambas en en el mismo valor(moverse sobre la diagonal que contiene a U_x,U_y) Siendo entonces *S* una función monótona creciente al tener los valores U_x, U_y fijos, entonces se pueden encontrar el valor *D* más pequeño tal que $Lx = Ux - D$ y $Ly = Uy - D$ y $S(Lx,Ly,Ux,Uy) >= X$ usando búsqueda binaria y la cantidad de máquinas que se pueden poner que terminan en *Ux, Uy* es $min(Lx,Ly)$
- Entonces el número de lugares donde se puede poner la máquina es la suma del paso anterior para cada *Ux,Uy* posible.