Editorial de los problemas del Club de Algoritmia CUCEI Intermedios 2019-B (STL avanzada)

Concurso: https://vjudge.net/contest/323783

Lunes 2 de Septiembre del 2019

A. Boxes Packing

El problema nos pide calcular la menor cantidad de cajas que serán visibles después de empacar una dentro de otra. Observemos los casos de ejemplo.

La caja azul entra en la caja verde y esta en la roja, dejándonos con una caja roja al final.

La caja verde entra en la caja roja y esta en la caja amarilla, dejándonos al final con sólo dos cajas amarillas

Esto no nos muestra tanto, pero si hacemos otro caso de ejemplo podemos ver de qué se trata la solución. La solución sería la cuenta más grande de cajas de tamaños iguales, en este caso la cuenta de las cajas azules.

Para esto podemos utilizar un `map<int, int>` que tenga las cuentas de los objetos y mantener cuál es la cuenta más grande para poderle dar respuesta al problema.

C. Tom Riddle's Diary

El problema nos pide decir si una string ya ha aparecido antes o no. Para esto haremos uso de un `set<string>` en donde preguntaremos si existe o no una `string s` mediante la función `find(s)` y dependiendo del resultado imprimiremos “YES” o “NO”.

D. Indian Summer

El problema es similar al anterior sólo que ahora nos pide la cantidad de elementos que tenemos, para esto podemos hacer uso de un `set< pair<string, string> >` y finalmente imprimir el tamaño de nuestro set que es lo que nos pide, alternativamente podemos hacer uso de un `set<string>` donde cada par de string los concatenaremos mediante un separador, o varios… (`ab = a + "#" + b + "$"`).

F. Physical Education

El problema nos pide dado un `a[]` convertirlo en `b[]`, es decir que cada `a[i]` coincida con todo `b[i]`, es decir `(a[i] == b[i])`, mediante operaciones de intercambio entre **elementos adyacentes**, e indicar los intercambios realizados.

Para resolverlo, en todo `a[i]`, en orden `i ∈ [1..n]`, buscamos una `j >= i` que cumpla `a[i] == b[j]`, entonces lo que tenemos que hacer es llevar `b[j]` a `b[i]` para que se cumpla con la condición `a[i] == b[i]`.

arreglo / posición	1	2	3	4
a[]	1	2	3	2
b[]	3	2	1	2

Para i: 1 que es 1 (a[1]), buscamos en b[] un 1, que se encuentra en j: 3 (a[3]), y lo que procedemos a hacer es intercambiarlo hasta que j sea i.

arreglo / posición	1	2	3	4
a[]	1	2	3	2
b[]	3	1	2	2

arreglo / posición	1	2	3	4
a[]	1	2	3	2
b[]	1	3	2	2

Como se puede notar los intercambios hechos fueron (2, 3) y (1, 2) y ahora `a[1] == b[1]`, por lo que podemos seguir continuando con las siguientes i's buscando nuevas j's hasta terminar con el proceso descrito anteriormente.

B. Okabe and Boxes

El problema nos pide la cantidad mínima de acomodos de la pila (hacer un sort sobre ella) de tal manera que podamos sacar la permutación ordenada de la misma {1, 2, 3, … n - 1, n}.
Para esto sólo necesitamos saber quién tenía que ser el tope de la pila al momento de sacar un elemento y si **no** coincide con el tope que tenemos nosotros entonces tendremos que hacer un ordenamiento para garantizar sacar el tope que nosotros requerimos.
Para esto lo simularemos con una pila, donde cada operación de “add” agregaremos a la pila, pero cada operación de “remove” checaremos lo descrito anteriormente para ver si debemos de “ordenar” o no.

E. History

El problema nos pide la cantidad de parejas de eventos que cumplan que `a[j] < a[i] & b[i] < b[j]`, lo que nos indica que el evento `j` contiene al evento `i`, para esto ordenaremos un arreglo de eventos `{a, b}`, con lo cual tendremos cumplida siempre la primera condición `a[j] < a[i]`, pero la segunda tendremos que checar si se cumple, esto lo hacemos teniendo un evento previo que tiene como `b`, el número más grande, para así verificar que `b[i] < previo.b` y cumpla con la condición que se nos pide.

Referencias

http://www.cplusplus.com/reference/map/map/
http://www.cplusplus.com/reference/utility/pair/
http://www.cplusplus.com/reference/set/set/
http://www.cplusplus.com/reference/algorithm/sort/
http://www.cplusplus.com/reference/stack/stack/