Competencia Peruana de Informática Online 2018 - Solucionario

Federación Olímpica Peruana de Informática

26 de mayo de 2018

Tutorial del problema: "Subarreglos"

Autor:

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

Para resolver los casos de prueba de este grupo podemos obtener los valores del arreglo en $O(n)$ como se explica en el enunciado y luego obtener el máximo de cada rango posible en $O(k)$. Ya que hay $O(n)$ rangos posibles, la complejidad es de $O(kn)$.

Grupo 2 (Complejidad Esperada: $O(n \log n)$)

Para resolver los casos de prueba de este grupo debemos usar la técnica de *sliding window*, pues si definimos v_i como el conjunto de elementos en el rango $[i, i+k-1]$ tendremos la siguiente igualdad:

$$v_{i+1} = v_i - a_i + a_{i+k}$$

Donde $+x$ es insertar el elemento x y $-x$ es eliminar una ocurrencia de x.

De esta forma, podemos usar un `multiset<int>` para almacenar los v_i y cambiar de v_i a v_{i+1} en $O(\log k)$ (una inserción y una eliminación). Además, consultar el máximo toma $O(1)$ con la función `rbegin()`.

Finalmente, la complejidad será de $O(n \log n)$.

Bonus: Se puede resolver el problema en $O(n)$ usando *Monotonous Queue*.

Tutorial del problema: "Gusto por los Números"

Autor:

Grupo 1 (Complejidad Esperada: $O(b-a)$)

Para resolver los casos de prueba de este grupo podemos simplemente iterar desde a hasta b, verificar si su dígito de las unidades es 0 o 5 (Esto se puede verificar sacando el residuo respecto a 10) y llevar el conteo en una variable. La complejidad será de $O(b-a)$.

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

Para resolver los casos de prueba de este grupo debemos notar que los números que terminan en 0 o 5 son los múltiplos de 5, así que debemos contar la cantidad de múltiplos de 5 en el rango $[a, b]$. Para ello, uno puede usar razones aritméticas y conteo de términos, pero plantearemos una forma diferente de analizar el conteo:

Sea $f(x)$ una función que nos da la cantidad de múltiplos de 5 en el rango $[1, x]$, entonces la respuesta es $f(b) - f(a-1)$. Ahora, la cantidad de múltiplos de b en el rango $[1, n]$ es $\left\lfloor \frac{n}{b} \right\rfloor$. Finalmente podemos responder en $O(1)$.

Tutorial del problema: "Kiko"

Autor: Walter Erquinigo

Grupo 1 y 2 (Complejidad Esperada: $O(n^2)$)

Para resolver los casos de prueba de este grupo podemos probar el convencer a cada una de las provincias posibles y analizar si se puede o no ganar las elecciones. Para verificar podemos usar un algoritmo lineal para obtener la suma de las provincias que voten por Kiko y así obtener una solución $O(n^2)$.

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

Para resolver los casos de prueba de este grupo podemos probar todas las posibilidades calculando previamente la suma de todos los votantes por Kiko en la variable *suma*. De esta manera podemos probar

convencer a la provincia i y la cantidad de votantes será:

$$R = suma - k_i + p_i$$

Si hay alguno de estos valores tal que $2R > total$, entonces la respuesta es que sí se puede. En particular, la provincia con máximo $p_i - k_i$ nos define si se puede o no (si puede, entonces es una posible respuesta, mientras que si no puede, el resto no podrá por tener una menor diferencia).

Tutorial del problema: "Trabajo de verano"

Autor: Aldo Culquicondor

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

Para resolver los casos de prueba de este grupo debemos plantear un balance sobre los billetes de 10:

- Si un cliente paga con 10, la cantidad de billetes de 10 aumenta en 1.
- Si un cliente paga con 20, la cantidad de billetes de 10 disminuye en 1.

Lo anterior nos da la idea de que la variación siempre es ± 1, así que necesitamos que n sea par para que el balance final pueda ser 0.

Otra restricción es que no podemos tener una cantidad negativa de billetes, así que debemos plantear una recursión $f(pos, balance)$ que nos dé la cantidad de formas en las que podemos cumplir con las condiciones dado que ya hemos procesado los primeros *pos* clientes y hemos obtenido un balance de *balance* billetes de 10. Las transiciones serán de la siguiente forma:

- Si el balance es positivo, el siguiente cliente puede pagar con 20, así que nos iremos a $f(pos + 1, balance - 1)$.
- El siguiente cliente puede pagar con 10, así que nos iremos a $f(pos + 1, balance + 1)$.

Cuando $pos = n$, debemos devolver 1 si el balance es 0 y 0 en caso contrario. La complejidad final será de $O(2^n)$, pues en el peor de los casos cada posición tiene 2 opciones.

Grupo 4 (Complejidad Esperada: $O\left(n^2\right)$)

Para resolver los casos de prueba de este grupo nos basta transformar la recursión anterior a programación dinámica y obtener un algoritmo $O(n^2)$. Sin embargo, el problema no es tan directo como parece, dado el límite de memoria, debemos plantear la solución con ahorro de memoria. Para lograr lo anterior, debemos notar que solo necesitamos la fila *pos* para obtener la fila $pos + 1$, así que en realidad nos basta $O(n)$ de memoria representada por dos arreglos *last* (fila *pos*) y *memo* (fila $pos + 1$). Con una implementación adecuada la solución nos dará AC.

Bonus: Se puede resolver el problema en $O(n)$ notando que la respuesta es igual al $\left(\frac{n}{2}\right)$-ésimo número de Catalán (el problema es equivalente a expresiones de paréntesis correctamente balanceadas) y usando pequeño teorema de fermat para hallar inversa módulo $10^9 + 7$ (pues debemos evitar el uso excesivo de memoria).