Descripción

Estás trabajando en una agencia de espionaje, y tu trabajo es descifrar los mensajes interceptados a las agencias rivales. Tu estrategia consiste en buscar las palabras más repetidas en los mensajes interceptados, e intentar descifrarlas por su cuenta ya que asumes que son términos reconocibles. Para esta tarea, tu primer objetivo es diseñar un programa que revise el mensaje interceptado y devuelva la palabra que aparezca más veces.

Entrada

- En la primera línea, recibes un entero n, que representa la cantidad de palabras que contiene el mensaje (incluyendo repeticiones).
- A partir de la segunda línea, cada línea contiene una palabra del mensaje interceptado.

Salida

La palabra que más veces se repite en la entrada. Puedes asumir que esta palabra es única.

Subtarea 1 [60 puntos]

$n < 100$

Subtarea 2 [40 puntos]

$n < 100000$

[Subtarea 1] *Solución:*
Para una primera solución, podemos contar cuántas veces aparece cada palabra del arreglo. Para cada una de las palabras en el arreglo (sin importar repeticiones), iteramos sobre el arreglo contando cuántas veces aparece. Guardamos el máximo número de veces y la respectiva palabra hasta ese entonces. Si la palabra que estemos considerando aparece más veces que el máximo actual, entonces esta se convierte en el máximo. Esta solución es claramente cuadrática $O(n^2)$ en tiempo pero utiliza espacio constante $O(1)$.

[Subtarea 2] *Solución:*
Podemos ordenar el arreglo sabiendo que una vez ordenado, las palabras idénticas aparecen en posiciones adyacentes. Con esta propiedad podemos iterar solamente una vez sobre el arreglo, recordando la palabra que hasta ese entonces se repita más veces. Ya que esta solución ordena el arreglo, si se usa la librería estándar tomaría tiempo $O(n \log n)$ y espacio constante $O(1)$.

Podemos resolver el problema aún de forma más eficiente si usamos espacio extra. Si usamos una estructura de datos asociativa, como un mapa no-ordenado (por ejemplo std::unordered_map), podemos guardar las palabras distintas como clave y su valor siendo la cantidad de veces que aparecen. Al final, simplemente iteramos sobre el mapa hallando la palabra que se repita más. Esta solución toma tiempo esperado lineal $O(n)$, ya que la inserción en un unordered_map es valor esperado constante $O(1)$. El espacio extra usado es lineal $O(n)$, ya que en el peor de los casos todas las palabras son distintas y necesitamos una entrada en el mapa para cada una.