Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2283 - Diseño y Análisis de Algoritmos - 2do semestre 2022

TAREA 3

Publicación:	Martes 4 de octubre.
GitHub Classroom:	https://classroom.github.com/a/rDBmM-rw
Entrega:	**Viernes 14 de octubre 23:59 horas**.

Indicaciones

- La tarea es estrictamente individual.
- La solución puede ser entregada en el archivo `t3.py` del repositorio privado asignado mediante GitHub Classroom para esta tarea. Se revisará el último *commit* subido antes de la entrega al repositorio. Se usará Python 3.10.X para la revisión.
- El *input* para el programa debe ser obtenido desde *standard input*. El *output* debe ser entregado mediante *standard output*.
- La corrección se realizará mediante *tests* automatizados acordes al formato de *input* y *output* especificado. Cada *test* tendrá un *timeout* según lo que se especifica como tiempo esperado.
- Un *test* se considerará **reprobado** en caso de que 1) dado el *input* el *output* sea incorrecto, 2) exista un error de *runtime* durante su ejecución, o 3) el *timeout* se cumpla durante su ejecución. En otro caso, el *test* se considerará **aprobado**.
- No se permite el uso de librerías externas a la librería estándar de Python *a priori*. Consultar en las *issues* del repositorio oficial del curso en caso de requerir una.
- **Para esta tarea sí aplica la política de atrasos descrita en el programa del curso.**

Problema

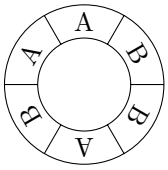
Una inteligencia artificial llamada IA-n con mucho tiempo libre recorre el planeta Tierra 100.000 años luego de la extinción de la sociedad humana. IA-n fue programada para leer palabras y actualmente solo le interesa encontrar alguna palabra escrita en algún vestigio humano. Un día se topa con una tira de metal oxidada con una palabra que sólo contiene letras "A" y "B". Sin embargo, el óxido ha corroído algunas de las letras y ahora resulta imposible saber que letras estaban escritas en esas posiciones.

Resulta que IA-n se ha olvidado de todos los alfabetos humanos, por lo que esta es la primera vez que puede expandir su alfabeto interno a $\Sigma = \{A, B\}$. En la infinitud de tiempo que posee IA-n a su dispoción, se plantea la siguiente pregunta: Al reeemplazar con "A" o "B" en cada posición ilegible en la palabra ¿A qué valores puede equivaler el **período** del palabra resultante?

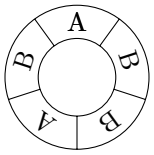
Para $w \in \Sigma^*$ con $|w| = n$ decimos que $k \in \{1, \ldots, n\}$ es un **período** de w si al desplazar w en k posiciones a la derecha sobre sí misma, todas las letras superpuestas coinciden. Formalmente, si $w = a_1 \cdots a_n$ entonces

w tiene período k sí, y solo sí, $a_i = a_{i+k}$ para todo $i \in \{1, \ldots, n-k\}$. Por ejemplo, 3 y 5 son períodos de "$ABBAB$"; y 2, 4 y 6 son períodos de "$ABABAB$".

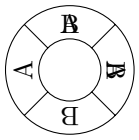
En este problema, dada una palabra w con letras ilegibles se debe entregar todos sus períodos posibles.



$k = 6$



$k = 5$



$k = 4$

Figure 1: Se puede visualizar el problema al colocar la palabra en un anillo. Si la palabra es más larga que el anillo, esta se solapa sobre sí misma. En el ejemplo, la palabra "$ABBABA$" tiene período 6 y 5, pero NO tiene período 4 (las letras solapadas no coinciden).

Input

La primera línea contiene un único número entero: el número de casos de prueba.

Cada caso de prueba se describe en dos líneas: la primera línea contiene un solo entero n ($1 \leq n \leq 10^4$) que corresponde a la longitud de la palabra. La segunda línea contiene un palabra de longitud n que consiste en letras "A", "B" y caracteres "?". Este último significa que la letra en su posición es ilegible.

Puede asumir que la suma de las longitudes entre todos los casos de prueba no excede 10^4.

Output

Para cada caso de prueba imprime dos líneas. En la primera línea imprime el número de períodos posibles después de sustituir cada letra ilegible por "A" o "B". En la siguiente línea imprime todos estos valores en orden creciente.

Tiempo esperado

Se espera que la solución se ejecute en un tiempo **menor o igual a 1 segundo**. Sin embargo posiblemente se espere más tiempo para las instancias más grandes[1].

Complejidad esperada

Se espera que la solución posea una complejidad de $\mathcal{O}(n \log(n))$ para cada caso de test.

Hint: Primero intente obtener una condición sobre pares de índices $i, j \in \{1, \ldots, n\}$ tal que $i \neq j$ para que la palabra tenga período k. Luego considere reducir su problema a una multiplicación de polinomios con tal de utilizar el algoritmo FFT para computar la repuesta correcta.

[1] Esto se debe a que implementar FFT en Python puede resultar muy ineficiente.

Ejemplo

Los siguientes tests están ya cargados a GitHub Classroom con corrección automática mediante GitHub Actions. **Los tests para la corrección serán distintos a estos.**

Input
3 5 A??AB 6 ?????? 4 ?AB?
Output
2 3 5 6 1 2 3 4 5 6 3 2 3 4

Nota: Para el primer caso del ejemplo podemos obtener, por ejemplo, la palabra "$ABBAB$" que tiene períodos 3 y 5.

Para el segundo caso podemos obtener "$AAAAAA$" que tiene todos los períodos del 1 al 6.

Para el tercer caso la palabra "$BABA$" tiene períodos 2 y 4, y la palabra "BABB" tiene períodos 3 y 4.