enunciado.pdf

user_3100197

Fundamentos de Algoritmia

2º Grado en Ingeniería Informática

Facultad de Informática
Universidad Complutense de Madrid

Fundamentos de Algoritmia

Grados en Ingeniería Informática

Convocatoria ordinaria, 19 de enero de 2024. Grupos B, G y H

1. (3.5 puntos) Dado un vector de $n \geq 0$ enteros no negativos y un entero $k > 0$, diseña un algoritmo **iterativo eficiente** que determine la longitud del tramo de valores consecutivos más corto cuyos elementos sumen *exactamente* k. En caso de que no exista ningún tramo cuyos valores sumen k, el algoritmo deberá devolver $n + 1$. Debes, asimismo, justificar la corrección (precondición, postcondición, invariante, cota) y el orden de complejidad del algoritmo.

Tu algoritmo se probará mediante casos de prueba que constarán de tres líneas:

- En la primera línea aparecerá el número n de elementos del vector ($0 \leq n \leq 1000000$).
- En la segunda línea aparecerán, en orden, los elementos del vector.
- En la tercera línea aparecerá el valor k.

La lista de casos de prueba finalizará con una línea con `-1`. Para cada caso de prueba, el programa imprimirá la longitud del tramo de valores consecutivos más corto cuyos elementos suman exactamente k. En caso de no existir ninguno, se escribirá $n + 1$.

A continuación se muestra un ejemplo de entrada/salida:

Entrada	Salida
6 2 4 4 2 2 0 8 6 2 4 4 2 2 0 9 6 0 4 4 2 8 2 8 -1	2 7 1

2. (2.5 puntos) Un patrón de *punto y cruz* es una secuencia de caracteres en la que se repite una y otra vez la cadena "`.x`". Una máquina ha generado patrones de este tipo y los ha almacenado en vectores de caracteres. Lamentablemente la máquina tiene un defecto: siempre intercala un "`*`" en el patrón.

Desarrolla un algoritmo de **divide y vencerás eficiente** que, dado un vector generado por nuestra máquina, determine la posición del defecto (es decir, la posición en la que se encuentra el "`*`"). Determina razonadamente, además, el coste del algoritmo planteando y resolviendo las recurrencias adecuadas.

Tu algoritmo se probará mediante casos de prueba que constarán de dos líneas:

- En la primera línea aparecerá el número n de elementos del vector ($0 < n \leq 1000000$).
- En la segunda línea aparecerá la cadena de n caracteres generada por la máquina.

La lista de casos de prueba finalizará con una línea con `-1`. Para cada caso de prueba, el programa imprimirá la posición del vector en la que aparece el defecto (recuerda que la primera posición es la 0).

A continuación se muestra un ejemplo de entrada/salida:

Entrada	Salida
15 .x.x.x.*x.x.x.x 15 *.x.x.x.x.x.x.x -1	7 0

3. (4 puntos) La red social `Triki-Tok` permite formar *grupos de amistad*. Estos grupos deben integrar un mínimo de K usuarios, todos ellos afines entre sí. Dos usuarios i y j son afines entre sí cuando se siguen mutuamente (es decir, cuando i sigue a j y j sigue a i).

En `Triki-Tok` los usuarios tienen, además, un *grado de actividad*. Eso hace posible determinar el grado de actividad de un grupo de amistad calculado como la suma de los grados de actividad de todos sus integrantes.

Los diseñadores de `Triki-Tok` quieren saber cuál es el grado de actividad más grande que puede alcanzar un grupo de amistad (que, como ya se ha comentado, debe estar formado por K o más integrantes). Además quieren saber cuántos grupos distintos con ese grado de actividad máximo pueden formarse.

Diseña un algoritmo de **vuelta atrás** que determine (i) el número de grupos de amistad más activos (o con mayor grado de actividad) que pueden formarse en `Triki-Tok`; (ii) el grado de actividad máximo que comparten todos estos grupos de amistad más activos. Para ello, el algoritmo podrá utilizar la siguiente información:

1. El mínimo número de usuarios K que deben integrar un grupo de amistad (no se permiten grupos de amistad con menos de K usuarios).
2. Una matriz booleana de seguimiento S que determina qué usuarios siguen a qué usuarios: $S_{i,j}$ será 'true'cuando i siga a j, y 'false'cuando i no siga a j (los valores de la diagonal principal de la matriz no tienen utilidad).
3. Un vector A de enteros que determina el grado de actividad A_i de cada usuario i. Es importante observar que A_i puede ser negativo (será el caso de aquellos usuarios especialmente pasivos, que, en lugar de sumar, restan actividad).

Se valorará la incorporación de mecanismos al algoritmo que reduzcan el número de potenciales soluciones a explorar.

Tu algoritmo se probará mediante distintos casos de prueba:

- La primera de línea de cada uno de ellos contendrá el número n de usuarios de la red ($0 \leq n \leq 100$)
- La segunda el mínimo número K de usuarios que debe haber en los grupos de amistad ($K \geq 0$).
- A continuación aparecerán n líneas, una por cada usuario, con n 0's o 1's que indican la correspondiente fila de la matriz de seguimiento S (0 significa 'false', y 1 significa 'true').
- La última línea de cada caso de prueba contiene n enteros con los grados de actividad de los usuarios.

La entrada terminará con una línea con un -1 que no debe ser procesada.

Por cada caso de prueba se escribirá una línea con dos números: el número de grupos más activos que pueden formarse en la red seguido de su actividad máxima. Si no se puede formar ningún grupo, se escribirá `NO HAY GRUPOS`.

A continuación se muestra un ejemplo de entrada/salida:

Entrada	**Salida**
5 2 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 0 1 1 0 1 1 0 10 20 0 -7 20 3 2 1 1 1 1 1 1 1 1 1 100 -10 -20 4 3 0 1 1 0 0 0 1 1 1 1 0 0 1 1 1 0 6 4 3 1 -1	3 30 1 90 NO HAY GRUPOS

En el primer caso de prueba la actividad máxima alcanzable por grupos de dos o más integrantes en el que todos sus integrantes son afines es 30 y hay tres formas de conseguirlo: con los grupos {0, 1}, {0, 1, 2} y {0, 4}. Un grupo como {0, 1, 4} no es posible pues el último usuario no es afin con el segundo. Desde el punto de vista de la afinidad, a los grupos anteriores se les podría añadir el usuario 3 pero eso haría que el grado de actividad del grupo se redujera en 7 unidades pues ese usuario tiene una actividad de -7.

En el segundo caso de prueba todos los usuarios son afines entre ellos pero dado que hay dos con actividad negativa, el grupo de máxima actividad de al menos dos integrantes que puede conseguirse es $\{0, 1\}$ con actividad total 90.

En el tercer caso de prueba no se puede crear ningún grupo de tres o más usuarios con todos ellos afines.