Matemática Discreta

HOJA 2 RESUELTA

En todos los ejercicios poner un ejemplo de funcionamiento del algoritmo y estudiar la complejidad.

1) Diseñar un algoritmo que tenga como entrada tres números enteros $a_1, a_2 \neq 0$ y $n > 0$ y su salida sea el resultado del cociente de a_1 y a_2 con n decimales exactos.

Consideramos que podemos calcular el valor absoluto, llamando a un algoritmo antes diseñado, y denotado con la notación habitual $|X|$.

Entrada: a_1, a_2; n.
$c_0 := |a_1|/|a_2|$
$r_0 := |a_1| \ mod \ |a_2|$
For $i = 1$ to n
  $c_i := 10r_{i-1}/|a_2|$
  $r_i := 10r_{i-1} \ mod \ |a_2|$
If $a_1a_2 \geq 0$ then $r := c_0, c_1...c_n$ else $r := -c_0, c_1...c_n$
Salida: r

Consideramos n el tamaño de la entrada. Es un algoritmo de complejidad lineal ya que hay un sólo bucle que se repite n veces y, cada vez que el contador del bucle toma un cierto valor, efectúa una cantidad constante de operaciones.

Veamos un ejemplo:

Entrada: 10, -3; 3.
$c_0 := |10|/|-3| = 3$
$r_0 := |10| \ mod \ |-3| = 1$
$i = 1$ entonces $c_1 := (10 \times 1)/3 = 3$ y $r_1 := 10 \ mod \ 3 = 1$
$i = 2$ entonces $c_2 := (10 \times 1)/3 = 3$ y $r_2 := 10 \ mod \ 3 = 1$
$i = 3$ entonces $c_3 := (10 \times 1)/3 = 3$ y $r_3 := 10 \ mod \ 3 = 1$
Como $10 \times (-3) \leq 0$ entonces
Salida: -3,333.

2) En este ejercicio trabajamos en el modelo computacional Real RAM. Las entradas están formadas por números reales y las operaciones básicas son las asignaciones, comparaciones y la suma, resta, producto y división de reales. Diseñar un algoritmo que tenga como entrada tres puntos distintos del plano real, esto es, tres pares ordenados de números reales, digamos (a_1, a_2), (b_1, b_2) y (c_1, c_2), y la salida determine si están o no alineados. Usando este primer algoritmo, diseñar otro algoritmo cuya entrada sea una lista de puntos del plano distintos entre sí y determine si hay entre ellos tres alineados o no.

Los tres puntos estarán alineados si el determinante siguiente es cero:

$$\begin{vmatrix} b_1 - a_1 & b_2 - a_2 \\ c_1 - a_1 & c_2 - a_2 \end{vmatrix}$$

Por tanto el algoritmo es:

Entrada: (a_1, a_2), (b_1, b_2), (c_1, c_2).
If $(b_1 - a_1)(c_2 - a_2) - (c_1 - a_1)(b_2 - a_2) = 0$ then $r := 1$ else $r := 0$.
Salida: r

La salida debe interpretarse: 1 si están alineados, 0 si no lo están. Lo hacemos así para poder usar la salida de este algoritmo en la segunda parte del ejercicio.

Es un algoritmo de complejidad constante pues siempre realiza el mismo número de operaciones.

Tomando los puntos $(1, 1)$, $(2, 2)$ y $(3, 3)$ el algoritmo computa el determinante:

$$\begin{vmatrix} b_1 - a_1 = 1 & b_2 - a_2 = 1 \\ c_1 - a_1 = 2 & c_2 - a_2 = 2 \end{vmatrix}$$

verifica que es 0 y por tanto la salida es $r = 1$, que siginifica que están alineados.

La segunda parte del algoritmo debe llamar a esta primera parte. De este modo consideramos el algoritmo *alineados* que se aplica a una terna de puntos del plano y da como salida r. Se trata entonces de construir todas las posibles ternas de puntos y verificar, usando el algoritmo anterior, si están o no alineadas.

Entrada: $p_1 = (a_{11}, a_{21})$, ..., $p_n = (a_{1n}, a_{2n})$
$s := 0$
For $i = 1$ to $n - 2$, while $s = 0$
  For $j = i + 1$ to $n - 1$, while $s = 0$
    For $k = j + 1$ to n, while $s = 0$
      $s :=$ *alineados*(p_i, p_j, p_k)
If $s = 0$ then $t :=$*no hay tres alineados* else $t :=$*sí hay tres alineados*
Salida: t

Es un algoritmo de complejidad cúbica puesto que hay tres bucles anidados.

Tomamos cuatro puntos: $(1, 1)$, $(0, 1)$, $(2, 2)$ y $(0, 0)$.
$s := 0$
$i = 1$, $j = 2$, $k = 3$: como *alineados*$((1, 1), (0, 1), (2, 2)) = 0$ entonces $k = 4$
como $k = 4 \leq 4$ entonces
$i = 1$, $j = 2$, $k = 4$: como *alineados*$((1, 1), (0, 1), (0, 0)) = 0$ entonces $k = 5$
como $k = 5 > 4$ entonces
$i = 1$, $j = 3$, $k = 4$: como *alineados*$((1, 1), (2, 2), (0, 0)) = 1$ entonces $s := 1$ y se sale de todos los bucles
como $s := 1$ entonces la salida es *sí hay tres alineados*

3) Volvemos al modelo computacional RAM. Diseñar un algoritmo que tenga, como entrada, una lista de números enteros: a_1, ..., a_n y un conjunto de números naturales (no repetidos) $\{b_1, ..., b_m\}$ con $1 \leq b_i \leq n$ y, como salida, una nueva lista construida a partir de la inicial borrando los elementos a_{b_1}, ..., a_{b_m}.

Comenzamos diseñando un algoritmo de complejidad lineal que borra el elemento de una lista a_1, ..., a_n que ocupa el lugar b-ésimo, lo llamaremos *borra*:

Entrada: a_1, ..., a_n; b (donde $1 \leq b \leq n$)
For $i = b$ to $n - 1$
  $a_i := a_{i+1}$
Salida: a_1, ..., a_{n-1}

Consideramos un algoritmo de ordenación llamado *ordena*, cuya entrada es un conjunto $b := \{b_1, ..., b_m\}$ y cuya salida es la lista de los elementos de b ordenada de menor a mayor.

Notación. A una lista $a_1, ..., a_n$ la denotamos a. Si escribimos $b := a$ queremos decir que a la lista b le asignamos los valores de la lista a.

Entrada: $a := a_1$, ..., a_n; $b := b_1$, ..., b_m.
$b := ordena(b_1, ..., b_m)$
for $i = 1$ to m
  $a := borra(a, b_i - i + 1)$
Salida: a

Si el tamaño de la entrada es n, el peor de los casos es cuando hay que borrar todos sus elementos. Es un algoritmo de complejidad cuadrática pues cada vez que se entra en el bucle i (como máximo n veces ya que $m \leq n$) se llama al algoritmo *borra* que tiene complejidad lineal. La ordenación no tiene complejidad mayor por lo que no aumenta la complejidad del algoritmo total. A este algoritmo lo denominaremos *borrar*.

Veamos un ejemplo:
Entrada: 7, 5, -1, -4, 2; 3,2.
$a := 7, 5, -1, -4, 2$
$b := ordena(3, 2) = 2, 3$
$i = 1$ $a := borra(a, 2) = 7, -1, -4, 2$
$i = 2$ $a := borra(7, -1, -4, 2; 3 - 2 + 1 = 2) = 7, -4, 2$
$i = 3$ sale del bucle
Salida: 7, -4, 2.

4) Diseñar un algoritmo que tenga como entrada dos listas de números enteros, digamos, a_1, ..., a_n y c_1, ..., c_m y un conjunto de números naturales (no repetidos) $\{b_1, ..., b_m\}$ con $1 \leq b_i \leq n$ y como salida una nueva lista donde se han sustituido los elementos correspondientes a_{b_1}, ..., a_{b_m} de la lista inicial por c_1, ..., c_m respectivamente.

Se construye un sencillo algoritmo lineal (el peor de los casos es $m = n$) donde se redefinen las entradas de la lista primera:

Entrada: a_1, ..., a_n; c_1, ..., c_m; b_1, ..., b_m.
For $i = 1$ to m
  $a_{b_i} := c_i$
Salida: a_1, ..., a_n.

Ejemplo: 7, -1, 2, 2; 1,-1; 1, 4.
$i = 1$ $a_1 := c_1 = 1$
$i = 2$ $a_4 := c_2 = -1$
Como $3 > m = 2$ entonces sale del bucle.
Salida: 1, -1, 2, -1.

5) Diseñar un algoritmo que tenga como entrada dos listas de números enteros: a_1, ..., a_n y b_1, ..., b_m y como salida una nueva lista de longitud $n + m$ resultado de concatenar ambas listas.

Sea m el tamaño de la entrada. Se construye un algoritmo de complejidad lineal, ya que hay un solo bucle y cada vez que se entra en dicho bucle se realiza una cantidad constante de operaciones.

Entrada: a_1, ..., a_n; b_1, ..., b_m.
For $i = n + 1$ to $n + m$
  $a_i := b_{i-n}$
Salida: a_1, ..., a_{n+m}.

Veamos un ejemplo:

Entrada: 3,-1,2; 5,5,-2.
$i = 4$ entonces $a_4 = b_{4-3} = b_1 = 5$
$i = 5$ entonces $a_5 = b_{5-3} = b_2 = 5$
$i = 6$ entonces $a_6 = b_{6-3} = b_6 = -2$
Salida: 3, -1, 2, 5, 5, 2.

6) Diseñar un algoritmo que tenga como entrada una lista de números enteros a_1, ..., a_n y como salida la lista a_{i_1}, .., a_{i_m} $(1 \leq i_1 < ... < i_m \leq n)$ de elementos de la lista inicial que son positivos.

Entrada: a_1, ..., a_n
$j := 1$
For $i = 1$ to n
  if $a_i \leq 0$ then $b_j := i$, $j := j + 1$
$s := borrar(a_1, ..., a_n; b_1, ..., b_{j-1})$
Salida: s

Es un algoritmo de complejidad cuadrática. Las operaciones en el bucle asociado al contador i se repiten n veces. En cada repetición realiza una cantidad constante de operaciones, por tanto esta parte del algoritmo es de complejidad lineal. Por otro lado, el algoritmo *borrar* es de complejidad cuadrática. Por tanto el algoritmo tiene complejidad cuadrática.

Entrada: 5, -2, 1, -1, 7.
$j = 1$
$i = 1$ como $a_1 > 0$ no se hace nada
$i = 2$ como $a_2 < 0$ entonces $b_1 = 2$ y $j = 2$
$i = 3$ como $a_3 > 0$ no se hace nada
$i = 4$ como $a_4 < 0$ entonces $b_2 = 4$ y $j = 3$
$i = 5$ como $a_5 > 0$ no se hace nada
$i = 6$ nos saca del bucle
$borrar(5, -2, 1, -1, 7; 2, 4) = 5, 1, 7.$