# Primer Problema de DAA

Carla Sunami Pérez Valera

Septiembre 2024

## 1 Enunciado del Problema

**Dificultad:** 1900

# B1. Painting the Array I

**time limit per test:** 2 seconds
**memory limit per test:** 512 seconds

**The only difference between the two versions is that this version asks the maximal possible answer.**

Homer likes arrays a lot. Today he is painting an array $a_1, a_2, \ldots, a_n$ with two kinds of colors, white and black. A painting assignment for $a_1, a_2, \ldots, a_n$ is described by an array $b_1, b_2, \ldots, b_n$ that $b_i$ indicates the color of $a_i$ (0 for white and 1 for black).

According to a painting assignment $b_1, b_2, \ldots, b_n$, the array $a$ is split into two new arrays $a(0)$ and $a(1)$, where $a(0)$ is the sub-sequence of all white elements in $a$ and $a(1)$ is the sub-sequence of all black elements in $a$. For example, if $a = [1, 2, 3, 4, 5, 6]$ and $b = [0, 1, 0, 1, 0, 0]$, then $a(0) = [1, 3, 5, 6]$ and $a(1) = [2, 4]$.

The number of segments in an array $c_1, c_2, \ldots, c_k$, denoted $seg(c)$, is the number of elements if we merge all adjacent elements with the same value in $c$. For example, the number of segments in $[1, 1, 2, 2, 3, 3, 3, 2]$ is 4, because the array will become $[1, 2, 3, 2]$ after merging adjacent elements with the same value. Especially, the number of segments in an empty array is 0.

Homer wants to find a painting assignment $b$, according to which the number of segments in both $a(0)$ and $a(1)$, i.e. $seg(a(0)) + seg(a(1))$, is as large as possible. Find this number.

**Input**

The first line contains an integer $n$ ($1 \leq n \leq 10^5$).

La segunda línea contiene $n$ enteros $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq n$).

**Output**

Output a single integer, indicating the maximal possible total number of segments.

**Examples**

**input:**

```
7
1 1 2 2 3 3 3
```

**output:**

```
6
```

**input:**

```
7
1 2 3 4 5 6 7
```

**output:**

```
7
```

**Note:**

In the first example, we can choose $a(0) = [1, 2, 3, 3]$, $a(1) = [1, 2, 3]$ and $seg(a(0)) = seg(a(1)) = 3$.So the answer is $3 + 3 = 6$.

In the second example, we can choose $a(0) = [1, 2, 3, 4, 5, 6, 7]$ and $a(1)$ is empty. We can see that $seg(a(0)) = 7$ y $seg(a(1)) = 0$. So the answer is $7 + 0 = 7$.

## 2 Solución

Formalmente, para cada secuencia $a_1, a_2, \ldots, a_n$, y suponiendo que $a_1, a_2, \ldots, a_n$ son enteros positivos, el número de segmentos en $a$ se define como:

$$seg(a) = \sum_{i=1}^{n} [a_{i-1} \neq a_i],$$

donde $a_0 = 0$, y $[expresiónbooleana] = 1$ si la expresión booleana es verdadera y 0 en caso contrario.

Reformulemos el problema como sigue:

**Problema.** Dada una secuencia $a_1, a_2, \ldots, a_n$, divídela en dos subsecuencias disjuntas $s$ y $t$ de tal manera que $seg(s) + seg(t)$ sea lo más grande posible.

**Solución.** Construiremos dos subsecuencias disjuntas escaneando la secuencia $a_1, a_2, \ldots, a_n$.

Configuración inicial: $s$ y $t$ son dos secuencias vacías, y $a_1, a_2, \ldots, a_n$ aún no se han escaneado.

Procedemos: Supongamos que los últimos elementos de $s$ y $t$ son $x$ e $y$, respectivamente, y $x = 0$ (resp. $y = 0$) si $s$ (resp. $t$) está vacío. Sea $z$ el elemento actual que estamos escaneando en $a_1, a_2, \ldots, a_n$. Nuestra estrategia codiciosa se describe en dos casos:

**Estrategia Greedy I:** Si $z$ es igual a uno de $x$ o $y$, entonces asignamos $z$ a la subsecuencia opuesta. Es decir, si $z = x$, entonces añadimos $z$ después de $y$; y si $z = y$, entonces añadimos $z$ después de $x$. En particular, si $z$ es igual a ambos $x$ y $y$, la asignación podría ser arbitraria.

**Estrategia Greedy II:** Si $z$ es diferente de ambos $x$ y $y$, entonces añadimos $z$ después del que tenga el valor más cercano. Es decir, sea $next(x)$ la próxima posición donde $x$ aparece en $a_1, a_2, \ldots, a_n$ después de $z$, entonces añadimos $z$ después de $x$ si $next(x) < next(y)$, y después de $y$ en caso contrario.

La estrategia greedy es intuitiva, y con esta estrategia, se obtiene inmediatamente un algoritmo de $O(n)$. Sin embargo, su demostración resulta complicada. La adjuntamos para comprobar su completitud.

### 2.1 Prueba Intuitiva

Consideremos cualquier asignación óptima $b_1, b_2, \ldots, b_n$, y mostraremos que nuestra estrategia no es peor que ella. Sea $a[l \ldots r] = a_l, a_{l+1}, \ldots, a_r$ la subarray de $a$.

Ahora supongamos que estamos en alguna posición $p$, donde la asignación óptima entra en conflicto con nuestra estrategia. Asumimos que $s = (a[1 \ldots p])(0) = s'_x$ termina en $x$, y $t = (a[1 \ldots p])(1) = t'_y$ termina en $y$, y $a_{p+1} = z$.

**Estrategia Greedy I:** Si la asignación $b$ entra en conflicto con la Estrategia Greedy I, entonces debemos tener $x \neq y$ y, sin pérdida de generalidad, asumimos que $x = z$. La Estrategia Greedy I sugiere que debemos añadir $z$ después de $y$, pero $b$ sugiere que añadamos $z$ después de $x$. Supongamos que $b$ resulta en los dos subarreglos $s'xzs''|t'yt''$, mientras que hay otra asignación óptima que coincide con nuestra estrategia y resulta en $s'xt''|t'yzs''$.

**Estrategia Greedy II:** Si la asignación $b$ entra en conflicto con la Estrategia Greedy II, entonces debemos tener $x$, $y$ y $z$ distintos y, sin pérdida de generalidad, asumimos que la próxima ocurrencia de $x$ va delante de la de $y$. La Estrategia Greedy II sugiere que

añadimos $z$ después de $x$, pero $b$ sugiere que añadimos $z$ después de $y$. Supongamos que $b$ resulta en los dos subarreglos $s'xs''|t'yzt''$. Consideremos dos casos.

**Caso 1.** Si $s''$ no comienza con $y$, entonces hay otra asignación óptima que coincide con nuestra estrategia y resulta en $s'xzt''|t'ys''$.

**Caso 2.** Si $s''$ comienza con $y$, es decir, $s'' = ys_1$, entonces, dado que la primera ocurrencia de $x$ está delante de la de $y$, debemos tener que $x$ está en $t''$, y asumimos que $t'' = t_1xt_2$. El resultado de $b$ se reescribe como $s'xys_1|t'yzt_1xt_2$. Encontramos que hay otra asignación óptima que coincide con nuestra estrategia y resulta en $s'xzt_1ys_1|t'yxt_2$ (Nota que $t_1$ no contiene ningún $x$ o $y$ en él)

## 3 Prueba Formal

El número de alternaciones en una secuencia $a$ comenzando con $x$ se define como:

$$seg_x(a) = \sum_{i=1}^{n} [a_{i-1} \neq a_i],$$

donde $a_0 = x$. Notemos que $seg_0(a) = seg(a)$.

Sea $f_{x,y}(a)$ el máximo posible de la suma de números de alternaciones en dos subsecuencias disjuntas $s$ y $t$ de $a$, es decir:

$$f_{x,y}(a) = \max_{s,t}\{seg_x(s) + seg_y(t)\},$$

donde $s$ y $t$ recorren todas las posibles parejas de subsecuencias disjuntas de $a$. Es obvio que el orden de $x$ e $y$ no importa, es decir, $f_{x,y}(a) = f_{y,x}(a)$. Notemos que nuestro objetivo es computar $f_{0,0}(a)$.

Sea $next(x)$ el menor índice $k$ tal que $a_k = x$, es decir, $next(x) = \min\{k \in N : a_k = x\}$. En caso de que no exista tal índice $k$, $next(x)$ se define como $\infty$.

De hecho, nuestro problema puede resolverse mediante programación dinámica independientemente de la complejidad temporal.

**Proposition1**[Programación Dinámica] Para $n \geq 1$ y todo $x, y \in N$:

$$f_{x,y}(a_1, a_2, \ldots, a_n) = \max\left\{f_{a_1,y}(a_2, \ldots, a_n) + [a_1 \neq x], f_{x,a_1}(a_2, \ldots, a_n) + [a_1 \neq y]\right\}.$$

En particular, para la secuencia vacía $\epsilon$, tenemos $f_{x,y}(\epsilon) = 0$.

Podemos obtener algunas propiedades inmediatas de $f_{x,y}(a)$ a partir de la recurrencia de programación dinámica anterior.

**Proposition 2** Para todo $x, y \in N$, $f_{x,0}(a) \geq f_{x,y}(a) \geq f_{x,x}(a)$. Además, si $next(y) = \infty$, entonces $f_{x,0}(a) = f_{x,y}(a)$.

Después de algunas observaciones, tenemos:

**Proposition 3** Para todo $x, y, z \in N$ y secuencia $a$, $f_{z,x}(a)+1 \geq f_{z,y}(a)$.

**Prueba** Por inducción sobre la longitud $n$ de la secuencia $a$.

**Base**. Es trivial para el caso $n = 0$ ya que el lado izquierdo siempre es 1 y el lado derecho siempre es 0.

**Hipótesis inductiva**. Supongamos cierto para el caso $n = k(k \geq 0)$, es decir, $f_{z,x}(a)+1 \geq f_{z,y}(a)$ se cumple para toda secuencia $a$ de longitud $k$. Ahora consideremos una secuencia $a_1, a_2, \ldots, a_{k+1}$ de longitud $k+1$.

**Caso 1**. $x = y$. Es trivial que $f_{z,x}(a) + 1 \geq f_{z,x}(a)$.

**Caso 2**. $z = x \neq y$. Debemos probar que $f_{x,x}(a) + 1 \geq f_{x,y}(a)$. Por la Proposición 1, necesitamos probar que:

$\mathrm{f}_{a_1,x}(a_2, \ldots, a_{k+1}) + [a_1 \neq x] + 1 \geq f_{a_1,y}(a_2, \ldots, a_{k+1}) + [a_1 \neq x]$, $f_{a_1,x}(a_2, \ldots, a_{k+1}) + [a_1 \neq x] + 1 \geq f_{x,a_1}(a_2, \ldots, a_{k+1}) + [a_1 \neq y]$.

La segunda desigualdad es obvia. La primera desigualdad se convierte en:

$$f_{a_1,x}(a_2, \ldots, a_{k+1}) + 1 \geq f_{a_1,y}(a_2, \ldots, a_{k+1}),$$

que se cumple por hipótesis inductiva.

**Caso 3**. $x \neq y = z$. Debemos probar que $f_{x,y}(a) + 1 \geq f_{x,x}(a)$. Por la Proposición 1, solo necesitamos probar que:

$$f_{x,a_1}(a_2, \ldots, a_{k+1}) + [a_1 \neq y] + 1 \geq f_{a_1,x}(a_2, \ldots, a_{k+1}) + [a_1 \neq x],$$

lo cual es obvio.

**Caso 4**. $x \neq y$, $z \neq x$ y $z \neq y$. Por la Proposición 1, $f_{z,x}(a)+1 \geq f_{z,y}(a)$ es equivalente a:

$\max\{f_{a_1,x}(a_2, \ldots, a_{k+1})+[a_1 \neq z], f_{z,a_1}(a_2, \ldots, a_{k+1})+[a_1 \neq x]\}+1$

$\geq \max\{f_{a_1,y}(a_2, \ldots, a_{k+1}) + [a_1 \neq z], f_{z,a_1}(a_2, \ldots, a_{k+1}) + [a_1 \neq y]\}$.

Procedemos por casos:

**Caso 4.1**. $a_1 = z$. El lado izquierdo se convierte en:

$$\max\{f_{z,x}(a_2, \ldots, a_{k+1}), f_{z,z}(a_2, \ldots, a_{k+1})+1\}+1 = f_{z,z}(a_2, \ldots, a_{k+1})+2$$

por hipótesis inductiva de que $f_{z,z}(a_2,\ldots,a_{k+1})+1 \geq f_{z,x}(a_2,\ldots,a_{k+1})$. El lado derecho se convierte en:

$$\max\{f_{z,y}(a_2,\ldots,a_{k+1}), f_{z,z}(a_2,\ldots,a_{k+1})+1\} = f_{z,z}(a_2,\ldots,a_{k+1})+1$$

por hipótesis inductiva de que $f_{z,z}(a_2,\ldots,a_{k+1})+1 \geq f_{z,y}(a_2,\ldots,a_{k+1})$. La desigualdad se cumple de inmediato.

**Caso 4.2**. $a_1 = x$. El lado izquierdo se convierte en:

$$\max\{f_{x,x}(a_2,\ldots,a_{k+1})+1, f_{z,x}(a_2,\ldots,a_{k+1})\}+1 = f_{x,x}(a_2,\ldots,a_{k+1})+2$$

por hipótesis inductiva de que $f_{x,x}(a_2,\ldots,a_{k+1})+1 \geq f_{z,x}(a_2,\ldots,a_{k+1})$. El lado derecho se convierte en:

$$\max\{f_{x,y}(a_2,\ldots,a_{k+1})+1, f_{z,x}(a_2,\ldots,a_{k+1})+1\}.$$

Por hipótesis inductiva de que $f_{x,x}(a_2,\ldots,a_{k+1})+1 \geq f_{x,y}(a_2,\ldots,a_{k+1})+1$ y $f_{x,x}(a_2,\ldots,a_{k+1})+1 \geq f_{x,z}(a_2,\ldots,a_{k+1})+1$, la desigualdad se cumple.

**Caso 4.3**. $a_1 = y$. El lado izquierdo se convierte en:

$$\max\{f_{y,x}(a_2,\ldots,a_{k+1})+1, f_{z,y}(a_2,\ldots,a_{k+1})+1\}+1.$$

El lado derecho se convierte en:

$$\max\{f_{y,y}(a_2,\ldots,a_{k+1}), f_{z,y}(a_2,\ldots,a_{k+1})+1\} = f_{z,y}(a_2,\ldots,a_{k+1})+1$$

por hipótesis inductiva de que $f_{y,y}(a_2,\ldots,a_{k+1})+1 \geq f_{z,y}(a_2,\ldots,a_{k+1})+1$. La desigualdad se cumple de inmediato ya que $f_{z,y}(a_2,\ldots,a_{k+1})$ aparece en ambos lados (y puede eliminarse juntos).

**Caso 4.4**. $a_1 \notin \{x,y,z\}$. El lado izquierdo se convierte en:

$$\max\{f_{a_1,x}(a_2,\ldots,a_{k+1})+1, f_{z,a_1}(a_2,\ldots,a_{k+1})+1\}+1.$$

El lado derecho se convierte en:

$$\max\{f_{a_1,y}(a_2,\ldots,a_{k+1})+1, f_{z,a_1}(a_2,\ldots,a_{k+1})+1\}.$$

Por hipótesis inductiva de que $f_{a_1,x}(a_2,\ldots,a_{k+1})+1 \geq \max\{f_{a_1,y}(a_2,\ldots,a_{k+1}), f_{a_1,z}(a_2,\ldots$ la desigualdad se cumple.

La desigualdad se cumple para todos los casos. Por lo tanto, la desigualdad se cumple para $n = k + 1$.

**Conclusión**. La desigualdad se cumple para todo $n \geq 0$.

**Proposition 4** Supongamos que $a_1, a_2, \ldots, a_n$ es una secuencia. Para todo $x, y, z \in N$ distintos, es decir, $x \neq y$, $y \neq z$ y $z \neq x$, si $next(x) < next(y)$, entonces $f_{z,y}(a) \geq f_{z,x}(a)$.

Por inducción sobre la longitud $n$ de la secuencia $a$.

**Base**. Es trivial para el caso $n = 0$ ya que ambos lados son 0.

**Hipótesis inductiva**. Supongamos cierto para el caso $n = k(k \geq 0)$, es decir, $f_{z,y}(a) \geq f_{z,x}(a)$ se cumple para toda secuencia $a$ de longitud $k$. Ahora consideremos una secuencia $a_1, a_2, \ldots, a_{k+1}$ de longitud $k + 1$.

**Caso 1**. $a_1 = z$. Por las Proposiciones 1 y 3, el lado izquierdo se convierte en:

$$\max\{f_{z,y}(a_2,\ldots,a_{k+1}), f_{z,z}(a_2,\ldots,a_{k+1})+1\} = f_{z,z}(a_2,\ldots,a_{k+1})+1,$$

y el lado derecho se convierte en:

$$\max\{f_{z,x}(a_2,\ldots,a_{k+1}), f_{z,z}(a_2,\ldots,a_{k+1})+1\} = f_{z,z}(a_2,\ldots,a_{k+1})+1.$$

La desigualdad se cumple de inmediato.

**Caso 2**. $a_1 = x$. Por la Proposición 1, el lado izquierdo se convierte en:

$$\max\{f_{x,y}(a_2,\ldots,a_{k+1}) + 1, f_{z,x}(a_2,\ldots,a_{k+1}) + 1\},$$

y el lado derecho se convierte en:

$$\max\{f_{x,x}(a_2,\ldots,a_{k+1}) + 1, f_{z,x}(a_2,\ldots,a_{k+1})\}.$$

Por la Proposición 2, tenemos:

$$f_{z,x}(a_2,\ldots,a_{k+1}) \geq f_{x,x}(a_2,\ldots,a_{k+1}),$$

y por lo tanto, la desigualdad se cumple.

**Caso 3**. $a_1 = y$. Esto es imposible porque $next(x) < next(y)$, es decir, hay un elemento de valor $x$ delante del primer elemento de valor $y$.

**Caso 4**. $a_1 \notin \{x, y, z\}$. El lado izquierdo se convierte en:

$$\max\{f_{a_1,y}(a_2,\ldots,a_{k+1}) + 1, f_{a_1,z}(a_2,\ldots,a_{k+1}) + 1\}.$$

El lado derecho se convierte en:

$$\max\{f_{a_1,x}(a_2,\ldots,a_{k+1})+1, f_{a_1,z}(a_2,\ldots,a_{k+1})+1\}.$$

**Caso 4.1**. Si $next(y) > next(z)$, entonces por hipótesis inductiva tenemos:

$$f_{a_1,y}(a_2,\ldots,a_{k+1}) \geq f_{a_1,z}(a_2,\ldots,a_{k+1}),$$

y (porque $next(y) > next(x)$, )

$$f_{a_1,y}(a_2,\ldots,a_{k+1}) \geq f_{a_1,x}(a_2,\ldots,a_{k+1}).$$

La desigualdad se cumple.

**Caso 4.2**. Si $next(y) < next(z)$, entonces por hipótesis inductiva tenemos:

$$f_{a_1,z}(a_2,\ldots,a_{k+1}) \geq f_{a_1,y}(a_2,\ldots,a_{k+1}),$$

y (porque $next(z) > next(y) > next(x)$, )

$$f_{a_1,z}(a_2,\ldots,a_{k+1}) \geq f_{a_1,x}(a_2,\ldots,a_{k+1}).$$

La desigualdad se cumple.

La desigualdad se cumple para todos los casos. Por lo tanto, la desigualdad se cumple para $n = k+1$.

**Conclusión**. La desigualdad se cumple para todo $n \geq 0$.

[Estrategia Codiciosa I] Supongamos que $a_1, a_2, \ldots, a_n$ es una secuencia. Para todo $x, y \in N$, si $a_1 = x$, entonces:

$$f_{x,y}(a_1,\ldots,a_n) = f_{x,x}(a_2,\ldots,a_n)+1.$$

Por la Proposición 1, tenemos:

$$f_{x,y}(a_1,\ldots,a_n) = \max\{f_{x,y}(a_2,\ldots,a_n), f_{x,x}(a_2,\ldots,a_n)+1\}.$$

Por la Proposición 3, tenemos:

$$f_{x,x}(a_2,\ldots,a_n)+1 \geq f_{x,y}(a_2,\ldots,a_n).$$

Combinando ambas desigualdades se obtiene la prueba.

**Proposition5**[Estrategia Greedy II] Supongamos que $a_1, a_2, \ldots, a_n$ es una secuencia. Para todo $x, y \in N$ con $x \neq y$, si $a_1 \notin \{x, y\}$, entonces:

$$f_{x,y}(a_1,\ldots,a_n) = \{\, f_{a_1,y}(a_2,\ldots,a_n)+1 si next(a_1) < next(y) f_{x,a_1}(a_2,\ldots,a_n)+1 si next(x) <$$

Si $next(a_1) < next(y)$, por la Proposición 4, tenemos:

$$f_{a_1,y}(a_2, \ldots, a_n) \geq f_{x,a_1}(a_2, \ldots, a_n).$$

Por lo tanto, por la Proposición 1, tenemos:

$$f_{x,y}(a_1, \ldots, a_n) = f_{a_1,y}(a_2, \ldots, a_n) + 1.$$

La misma afirmación se mantiene para $next(x) > next(a_1)$.

# 4 Complejidad de la Estrategia Codiciosa

La complejidad de la estrategia codiciosa presentada en el contexto del problema de dividir una secuencia en subsecuencias disjuntas para maximizar el número de segmentos se puede analizar de la siguiente manera:

## 4.1 Complejidad del Algoritmo

La estrategia codiciosa se basa en escanear la secuencia de entrada y construir dos subsecuencias disjuntas $s$ y $t$ a medida que se procesan los elementos de la secuencia. La complejidad del algoritmo se determina principalmente por el número de elementos en la secuencia y las operaciones realizadas en cada paso.

### 4.1.1 Operaciones en Cada Paso:

Para cada elemento $z$ en la secuencia, se determina si se debe agregar a la subsecuencia $s$ o $t$ basándose en las últimas ocurrencias de los elementos $x$ y $y$. Esto implica verificar las posiciones de las próximas ocurrencias de $x$ y $y$ y decidir en qué subsecuencia agregar $z$.

## 4.2 Complejidad Temporal:

La estrategia codiciosa se implementa en un solo recorrido a través de la secuencia, lo que da como resultado una complejidad temporal de $O(n)$, donde $n$ es la longitud de la secuencia. Esto se debe a que cada elemento se procesa una vez y las decisiones se toman en tiempo constante.

### 4.3 Complejidad Espacial:

La complejidad espacial del algoritmo depende de la cantidad de espacio adicional utilizado para almacenar las subsecuencias $s$ y $t$. En el peor de los casos, se podrían almacenar todos los elementos en una de las subsecuencias, lo que llevaría a una complejidad espacial de $O(n)$.

### 4.4 Conclusión

La estrategia codiciosa presentada es eficiente, con una complejidad temporal de $O(n)$ y una complejidad espacial de $O(n)$. Esto la convierte en una solución adecuada para el problema, permitiendo procesar secuencias de longitud considerable en un tiempo razonable. Sin embargo, la prueba de la validez de esta estrategia puede ser más compleja, como se ha mencionado en las secciones anteriores.