

Algoritmos y Estructuras de Datos III

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Abril 2017

Trabajo Práctico 1

| Alumno | LU | Correo electrónico |
|------------------------|--------|---------------------|
| Seijo, Jonathan Adrián | 592/15 | jon.seijo@gmail.com |

Índice

| | |
|---|----------|
| 1. Introducción | 3 |
| 1.1. Explicación del problema | 3 |
| 1.2. Ejemplo | 3 |
| 2. Backtracking | 4 |
| 2.0.1. Pseudocódigo | 4 |
| 3. Programación Dinámica | 5 |

1. Introducción

1.1. Explicación del problema

Dada una secuencia A de números, se quieren pintar cada uno de ellos con rojo, azul o dejarlos sin pintar. Una aclaración importante es que los elementos de A no pueden modificarse, ni tampoco cambiarse su orden inicial. Lo unico que puede hacerse con ellos es colorearlos (o no).

Para que una secuencia de colores se considere **válida** es necesario que se cumplan ciertas condiciones:

1. Todos los elementos de color **rojo** están ordenados por valor de forma estrictamente creciente
2. Todos los elementos de color **azul** están ordenados por valor de forma estrictamente decreciente

(Estrictamente significa que no hay numeros consecutivos iguales)

Las secuencias de colores válidas pueden tener diferentes cantidades de elementos sin pintar. El objetivo del problema es encontrar la **mínima cantidad de elementos sin pintar** de todas las secuencias válidas que pueden formarse a partir de A .

1.2. Ejemplo

Supongamos que $A = [0, 7, 1, 2, 2, 1, 5, 0]$. Veamos *algunas* de las posibles secuencias de colores válidas:



Consideremos los colores del tercer caso para ver que es una secuencia válida.

1. **Rojos:** $[0, 1, 2, 5]$ (estrictamente crecientes)
2. **Azules:** $[7, 2, 1, 0]$ (estrictamente decrecientes)

Podemos ver que diferentes formas de pintar de rojo y azul nos obligan a dejar algunos elementos sin pintar para que la secuencia sea válida. En el caso de este ejemplo la **mínima cantidad de elementos sin pintar** que puede obtenerse de A es **0**, como puede verse en la tercer combinación.

2. Backtracking

2.0.1. Pseudocódigo

```

procedure BACKTRACK(sequencia(Colores) colores, int actual)
  if actual = n then
    if EsValido(colores) then
      return CantSinPintar(colores)
    else
      return  $\infty$ 
  else
    colores[actual]  $\leftarrow$  Rojo
    minimoConRojo  $\leftarrow$  backtrack(colores, actual + 1)

    colores[actual]  $\leftarrow$  Azul
    minimoConAzul  $\leftarrow$  backtrack(colores, actual + 1)

    colores[actual]  $\leftarrow$  Ninguno
    minimoSinPintar  $\leftarrow$  backtrack(colores, actual + 1)

    return Min(minimoConRojo, minimoConAzul, minimoSinPintar)

```

Auxiliares:

```

procedure ESVALIDA(sequencia(Colores) colores)
  bool rojoValido  $\leftarrow$  EsCreciente(DameRojos(colores))  $\triangleright O(n)$ 
  bool azulValido  $\leftarrow$  EsDecreciente(DameAzules(colores))  $\triangleright O(n)$ 
  return (rojoValido  $\wedge$  azulValido)

```

```

procedure CANTSINPINTAR(sequencia(Colores) colores)  $\triangleright O(n)$ 
  return Tamaño(DameSinPintar(colores))

```

3. Programación Dinámica