# Ejercicios de Programación Dinámica

Francisco Vicente Suárez Bellón

Septiembre de 2024

## Definición formal del problema

Hay $n$ bolas. Están dispuestas en una fila. Cada bola tiene un color (para conveniencia, un número entero) y un valor entero. El color de la $i$-ésima bola es $c_i$ y el valor de la $i$-ésima bola es $v_i$.
La ardilla Liss elige algunas bolas y forma una nueva secuencia sin cambiar el orden relativo de las bolas. Ella quiere maximizar el valor de esta secuencia. El valor de la secuencia se define como la suma de los siguientes valores para cada bola (donde *a* y *b* son constantes dadas):

1. Si la bola no está al principio de la secuencia y el color de la bola es el mismo que el de la bola anterior, suma (el valor de la bola ) x $a$ .

2. De lo contrario, suma(el valor de la bola) x $b$.

Se te dan $q$ consultas. Cada consulta contiene dos enteros $a_i$ y $b_i$ .

Para cada consulta, encuentre el valor máximo de la secuencia que puede crear cuando $a = a_i$ y $b = b_i$ .

**Nota:** Ten en cuenta que la nueva secuencia puede estar vacía, y el valor de una secuencia vacía se define como cero.

## Solución Fuerza Bruta

El problema puede tratarse como en cada bola tomar la decisión de añadirla a la subsecuencia. Si se generan todas las subsecuencias posibles entonces se puede seleccionar la de valor máximo.

Puede ser visto como generar todas las cadenas binarias donde en caso de la bola en la posicion i sea $True$ entonces se añadirá a la subsecuencia y en caso contrario no se añadirá aquí seleccionar el conjunto de estas cuyo valor sea máximo (Cualquier subsecuencia de este conjunto a efectos del problema son iguales dado que tiene igual valor).

- Esta solución es exponencial $O(2^n)$

## Podas

- Se puede pensar en no añadir una bola dado que el valor multiplicado por $a$ o $b$ lo que corresponda es $< 0$, lo que descartaría casos que no llevan al óptimo.
- Sin embargo, en ciertos casos es necesario, por ejemplo:

| $v$ | 1 | $-2$ | 1 | 4 | 0 | $-1$ |
|---|---|---|---|---|---|---|
| $c$ | 1 | 2 | 1 | 2 | 1 | 1 |
| $index$ | 0 | 1 | 2 | 3 | 4 | 5 |

Para $a = 2$ y $b = 1$, escoger los índices 0-1-8 es conveniente.

- Esto se debe a que el valor máximo si la subsecuencia termina con el color $c$ es $-\infty$, dado que nunca se ha escogido ese color.
- Si ese color ya tuvo representación en alguna subcadena explorada, poner un negativo después de multiplicar con $a$ o $b$ no puede llevar a una solución óptima.

```
def solver(n,q,v,c):
resultado = generar_combinaciones(n)

for _ in range(q):
    a, b = map(int,input().split())
    chains:list[Manager]=[]
    for combinacion in resultado:
        manager=Manager(a=a,b=b)
        for i in range(n):
            if not combinacion[i]:
                continue
            manager.add(index=i,value=v[i],color=c[i])
        chains.append(manager)
    chains=sorted(chains,reverse=True)
    #print(chains)
    print(f"El mayor valor es {chains[0].value}")
```

## 0.1 Correctitud del algoritmo

Como tenemos un algoritmo que genera todas las cadenas binarias de longitud $n$ entonces solo tenemos que recorrer esa combinación y tener esa subsecuencia, después ordenamos desde mayor a menor por lo cual tenemos las subsecuencias de mayor valor

## 0.2 Complejidad

### 0.2.1 Temporal

- Obtener todas las cadenas binarias tiene un costo de $O(2^n)$

### 0.2.2 Espacial

- Tener todas las cadenas binarias tiene un coste de $O(n * (2^n))$

## Reformular el problema

- En nuestro caso, cualquier subsecuencia óptima tiene el mismo valor, por lo que todo elemento del conjunto óptimo global es óptimo igual.
- Pueden existir múltiples óptimos:

Para $a = 5$ y $b = 1$:

| | | | | |
|---|---|---|---|---|
| $v$ | 1 | 1 | 1 | 1 |
| $c$ | 1 | 2 | 1 | 2 |
| $index$ | 0 | 1 | 2 | 3 |

El máximo puede ser con 0-2-3 o 0-1-3, que es 7.

## Observaciones

- Si tenemos una bola que, al multiplicar con $a$ o $b$, da un valor $< 0 \Rightarrow$ esa bola (si es que la última antes de añadir es del mismo color o no) no puede ser añadida.
- El valor máximo global requiere que todos los factores de las bolas sean positivos.
- Un valor de cero en una bola no afecta si está al final de la cadena máxima, pero puede influir si está en el medio, ya que la siguiente bola puede ser de color análogo, haciendo que esta esté en la cadena máxima.

## Notación

- $c[i]$ color de la bola $i$.
- $W_i$ cadena que termina con la bola $i$
- $W_i^*$ cadena de valor máximo que termina con la bola $i$
- $Col[W_i]$ color de la última bola.
- $W_{c_i}$ subcadena que termina en el color $c_i$
- $Val[W_{c_i}]$ Valor de la subcadena $W_{c_i}$
- $v[i]$ valor de la pelota $i$
- $W_{c_i}^*$ cadena de valor óptimo para el color $i$

| ..... | $bola_i$ |
|---|---|

Table 1: Ejemplo de como se debe unir la bola

# Ideas para Optimizar la Solución

Dado que el problema de decisión de poner o no la bola $i$ depende exclusivamente del valor de las cadenas de $Val[W_{c_i}]$ por cada color $c_i$.
En cada bola $i$ el problema se puede resumir a que subcadena $W_{c_i}$ (puede ser una subcadena vacía) se unirá esta bola para aumentar el valor.

Por tanto se debe tomar la decisión de tomar el máximo valor entre:

- $max(W_j^* + b * v[i], W_k^* + a * v[i], b * v[i])$ para todo $Col[W_j^*] \neq c[i]$ y $Col[W_k^*] = c[i]$

y al recorrer todas las bolas posibles en cada una de ellas tenemos el $max(Val[W_{c_i}])$ para cada bola $i$ por tanto el valor óptimo de la solución es $max(max(Val[W_{c_i}]), 0)$ para toda $i$.

Entonces podemos reducir este problema a que para cada bola $i$ tenemos que calcular para toda $(j < i)$ la $W_j^*$ tal que sea máximo el valor de poner a la bola $i$ como último valor de esa subcadena $W_j^*$.

# Algoritmos Propuestos para Optimizar en Tiempo Polinomial

Siguiendo la idea anterior no es necesario para cada bola $i$ cual es el valor con cada $W_j^*$ $(j < i)$ dado que si tomamos:

- $max(W_{c_j} + b * v[i], W_{c_i} + a * v[i], b * v[i])$ para

## Algoritmo en $O(n \cdot c)$

### Idea

Supongamos que tenemos, por cada color, el valor máximo de todas las cadenas hasta el momento que terminan con ese color. Si recorremos el array de bolas, por cada bola determinamos si el valor máximo de nuestro color es menor que unirnos con alguna subsecuencia que termine en otro color.

### Demostración

Cada vez que seleccionamos una bola, miramos cuál es la bola que debe precederla. Comparando todos los colores, si el valor de alguno es mayor que el de nuestro propio color, actualizamos.

## Algoritmo en $O(n)$

### Notación

- $Max_1$: Valor de las cadenas óptimas hasta el momento.
- $Max_2$: Segundo mejor valor de las cadenas óptimas hasta el momento.
- $v[i]$: Valor de la bola $i$.

### Idea

En cada iteración nos interesa conocer el valor de las subcadenas óptimas que terminan en diferentes colores. Si dos colores tienen el mismo valor óptimo hasta el momento, los tratamos como equivalentes, y guardamos el valor máximo de todas las subcadenas óptimas en la variable $Max_1$, y el segundo mejor valor en $Max_2$.