Ejercicios de Programación Dinámica

Francisco Vicente Suárez Bellón

Septiembre de 2024

Contents

1 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.

2 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 $O(2^n)$

2.1 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}")
```

2.2 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

2.3 Complejidad

2.3.1 Temporal

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

2.3.2 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.

3 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.

4 Notación

- $W[j]$ Conjunto de subcadenas que terminan en j.
- $W^*[j]$ Conjunto de subcadenas que terminan en j con valor máximo.
- w_j^* subcadena representante del conjunto $W^*[j]$.
- $Val[w]$ valor de la cadena w
- $Col[i]$ color de la bola i
- $Col[w]$ color de la última bola de la cadena w.
- $W_\alpha^*[i]$ Conjunto de cadenas de valor máximo de color α hasta la bola i
- $w_\alpha^*[i]$ subcadena que es representante del conjunto $W_\alpha^*[i]$
- $End[i]$ Conjunto de subcadenas que donde terminan con la $bola[i]$
- $End^*[i]$ Conjunto de subcadenas de valor máximo que terminan con la $bola[i]$

.....	$bola_i$

Table 1: Ejemplo de como se debe unir la bola

5 Optimizar la Solución

Podemos ver el problema como un problema de decisión en el cual que subcadena W (puede ser la subcadena vacía) unimos la bola i por tanto para ello tenemos que ver por cada bola j ($j < i$) cual es la subcadena con la cual podemos unir i y que maximize el valor de esta. Inicialmente vemos que para toda $w_j^* \in W^*[j]$, $w^{`} = w_j^* + bola[i]$ el $Val[w^{`}]$ por tanto cualquier w_j^* añadir $bola[i]$ será el mismo valor

5.1 Lema 1:

Para todo $w_j^* \in W^*[j]$ $w^{`} = w_j^* + bola[i]$ el $Val[w^{`}]$ sera el mismo.

5.2 Nota

Dado lo anterior es evidente que una vez computado el $W^*[j]$ no es necesario recomputarlo en el resto de los llamados. Por lo tanto estamos ante una subestrutura óptimam, de las cuales una vez calculado se puede guardar en una caché de aquí partirá el resto de nuestras Ideas

6 Algoritmo en $O(n * c)$

Dado que anteriormente vimos que existe una subestructura óptima con respecto a guardar en cada bola cual es la mejor subcadena en la cual ella es la última bola. Partiendo de esta idea, en el problema no tiene importancia en si cual es la bola anterior, solo nos importa cual es la cadena $w_\alpha^*[i]$ por cada color.

6.1 Lema 2:

Para todas subcadenas hasta i con igual color la que se elegirá para unir a $bola[i]$ es la de mayor valor. Sean h, q bolas; $h, q < i$ y $Col[h] = Col[q]$ Si $Val[w_h^*] = Val[w_q^*]$ entonces al momento de elegir en la bola[i] cualquiera de las dos cadenas son iguales Si $Val[w_h^*] < Val[w_q^*]$ entonces siempre elegiremos a la w_q^* dado que al poner a la $bola[i]$ al final de estas la de mayor valor será óptima para las cadenas que terminan en el Col[h].

6.1.1 Algoritmo

Ahora conociendo que para cada color α la subcadena que se seleccionará para unir a la $bola[i]$. Osea $w_\alpha^*[i] \in End[i]$ por lo tanto $End^*[i]$ están las subcadenas

de los colores a los cuales añadir la $bola[i]$ hacen que su valor sea óptimo para esta.

Dado lo anterior podemos elaborar un algoritmo de programación dinámica donde iteraremos por cada bola , en ella iteramos por cada color, para ir actualizando el valor de la cadena máxima contando con la $bola[i]$ con una chaché $dp[\alpha]$ donde guarda el valor de la cadena de valor máximo en el color α de la siguiente forma:

- $dp[Col[bola[i]]] = max(dp[Col[bola[i]]], dp[\beta] + (b * v[i]), dp[Col[bola[i]]] + a(a * v[i]), b * v[i])$ para todo color $\beta \neq Col[bola[i]]$

6.2 Subestructura Óptima

Como se ha visto la función de asignacion de $w^*_\alpha[i]$ depende de los valores ya precomputados de las bolas que se han visto con anterioridad. Por tanto supongamos que ya computado la subcadena de valor máximo del color de la $bola[i]$ hasta la $bolai$ (Ya se computo esta) es s^* será el máximo entre lo ya computado entre todos los colores (también puede ser que lo subóptimo sea empezar en esta bola para este color en específico) Por tanto si obtengo una solución optima global del problema > 0 es porque la subcadena es distinta del vacío.

6.3 Correctitud de la DP

Para demostrar que la solución por DP funciona vamos a realizar una inducción sobre la cantidad de elementos **Caso Base :**

- Para $n = 1$ Se reduce a $max(b * v[i], 0)$
- Para $n = 2$ Vemos que se cumple la propiedad dado que la primera bola y su color respectivo se basan en caso base anterior por lo tanto el color que representa la primera bola tiene el subóptimo hasta dicha bola, por tanto la 2da bola dada la propia función solo puede mejorar o igualar la solución de los colores precomputados

6.3.1 Hipótesis :

Supongamos que para todo array con k bolas se cumple la propiedad de dp

6.3.2 Tesis:

Supongamos que tenemos un array de tamaño k+1 bolas, y lo separamos en 2 secciones, x, y donde x tendrá k elementos y y un elemento por hipótesis x tiene precomputado correctamente el dp. Hasta x se tienen precomputado correctamente por cada color cual es su subsecuencia máxima,

como anteriormente se ha demostrado que al añadir una bola para maximizar la subcadena máxima es suficiente con el valor de las subcadenas óptimas para cada color precomputadas hasta ese momento. por tanto para unir a y buscamos los valores de dp por cada color (incluido el vacío) y computamos la función de matcheo y elegimos la de mayor valor.

```
for _ in range(q):

    dp=[-INF]*(N)
    color_can=[False]*len(dp) # Llevamos la

    for i in range(n): # Por cada elemento
        ci=c[i]
        last=dp[ci]

        for col in range(1,len(dp)):

            if col == ci and color_can[ci]:

                s=last if last>-INF else 0
                dp[ci]=max(dp[ci],s+(a*v[i]))
            else:
                s=dp[col] if dp[col]>-INF else 0
                dp[ci]=max(dp[ci],s+(b*v[i]))

        dp[ci]=max(dp[ci],b*v[i])
        color_can[ci]=True

    print(max(max(dp),0))
```

6.4 Demostración de la correctitud del algoritmo

En este algoritmo por cada bola se recorre cada uno de los colores actualizando la dp, es evidente que para la primera ocurrencia de un color se rellena por primera vez el valor de cada color con el factor $b * v[i]$. Para cada bola hay 3 opciones.

1. Que exista una subcadena que termine en mi mismo color al cual al yo ponerme al final de este mejores la subcadena.
2. Que añadirme en el final de otra subcadena un color distinto al mio mejore con respecto al subóptimo que tenía
3. Que comenzar la subcadena pueda ser mas ventajoso

6.5 Complejidad

6.5.1 Temporal:

Como se itera por cada bola $O(n)$ y por cada una se itera por todos los colores $O(C)$ y todas las operaciones dentro de este bucle son constantes en tiempo. El algoritmo es $O(c * n)$.

6.5.2 Espacial:

Dado que la creación de las caches se suguiere hacerse con arreglos (los cuales permiten indexar en $O(1)$) El coste de estos son $O(C)$ siendo C la cantidad de colores distintos.

Algoritmo en $O(n)$

Basandonos en las mismas ideas demostradas en el anterior algoritmo de programación dinámica. Para poder eliminar cómputo innecesario podemos fijarnos en el caso que la bola a añadir se quiere poner al final de una subcadena que termina en un color distinto a este por lo tanto el valor de crear esa nueva subcadena es de valor subcadena color distinto $+ (b * (valordeesabola))$ por lo tanto intuitivamente si maximizamos el valor de la subcadena a la cual nos unimos entonces será la mejor entre todas las posibles.

Entonces si llevamos la cuenta de cual es el valor de las subcadenas máximas hasta cada bola, solo nos quería tocar el caso que la bola tenga un color igual al del max, por tanto debemos llevar el max_1 (máximo total) y max_2(segundo mayor máximo) Dado que se compara en este caso a añadir esa bola con una subcadena con valor max_2 (Aquí también se tiene en cuenta el empezar la cadena desde esa bola).
Nota: max_1 y max_2 son ≥ 0

6.6 Lema 3:

Demostrar que dado una$bola[i]$ de color $Col[bola[i]]$ de todas las subcadenas que se pueden formar tal que termine con esta bola y la penúltima bola tenga un color distinto a la $bola[i]$ la de mayor valor, s^* es la $s^* - bola[i]$ con mayor valor.

Supongamos que tenemos s^* que esta formada por $s_1 + i$, s_1, $Col[w_1] \neq Col[bola[i]]$ no es la de mayor valor entre las que tienen distinto color. Sea s_2, $Col[s_2] \neq Col[bola[i]]$ y s_2 tiene valor máximo entre las subcadenas que no terminan en $Col[bola[i]]$

Entonces si $Val[s^*] = Val[s_1] + (b * v[bola[i]])$ y $Val[s_m] = Val[s_2] + (b * v[bola[i]])$, $Val[s_1] < Val[s_2]$ Contradiccion. s_m máxima.

Dado que ahora no es necesario computar por las cadenas terminadas en distinto color para conocer si añadir la bola mejorará la dp entonces nos hay que analizar el caso en que la mejor solución sea mi propio color por tanto debo

comprobar con las 2das mejores soluciones para verificar que elección es más conveniente.

6.7 Subestructura Óptima

La subestrutura óptima es análoga a la propuesta anteriormente, aunque en este caso aplicamos en Lema 3, por tanto no hay afectación en la estructura de esta

6.8 Correctitud de la DP

Como en este algoritmo continuamos utilizando la misma idea dinámica anterior aplicando el Lema 3, nos queda añadir que no es necesario iterar por todos los colores dado que la aplicacion antes mencionada hacer que los colores distintos a la bola no sean necesarios. Y el caso de que el color de esa bola sea igual al del máximo se tiene el max_2 para comparar si es mejor.

6.9 Código

```
for _ in range(q):

    mx1 = 0
    mx2 = 0

    ans = [-INF] * N

    for i in range(n):
        if mx1 == ans[c[i]]:
            ans[c[i]] = max(ans[c[i]], ans[c[i]] + a * v[i])
            ans[c[i]] = max(ans[c[i]], mx2 + b * v[i])
            mx1 = max(mx1, ans[c[i]])
        else:
            ans[c[i]] = max(ans[c[i]], ans[c[i]] + a * v[i])
            ans[c[i]] = max(ans[c[i]], mx1 + b * v[i])
            mx2 = max(mx2, ans[c[i]])

            if mx2 > mx1:
                mx1, mx2 = mx2, mx1

    print(mx1)
```

6.10 Correctitud del algoritmo

Nota: $max_1, max_2 \geq 0$ durante todo el algoritmo por tanto nunca abrá en un color soluciones subóptimas negativas por lo cual las operaciones que hagamos

dependen del factor a o b y del valor de la bola a añadir para determinar si debe actualizar la caché.
El algoritmo itera por todas las posibles bolas, tratando a las bolas su color tienen a una subsecuencia con valor subóptimo de forma tal que comparan cual de las dos situaciones es mas beneficioso:

1. Alargar una subsecuencia que terminaba en mi mismo color
2. Alargar una subsecuencia que tenia el 2do mejor valor.

En el 2do caso el max_2 puede ser igual al max_1 en el caso que existan más de un color en ese momento cuyas subsecuencias que teminan en este son las mejores en terminos de valor. **Nota:** El max_1 nunca disminuye en esta opción.

En el caso que color de la bola a añadir no tenga una subsecuencia máxima entonces compruebo entre:

1. Añadir a la subsecuencia de mayor valor que termina en igual color
2. Añadirme a la subsecuencia con mejor valor y distinto color final (Aplicar lema 3).

Aca se actualiza el max_2 con el objetivo de continuar con las propiedades necesarias demostradas anteriormente. Para mantener las invariantes de los máximos se realiza el swap si es necesario.

6.11 Complejidad

6.11.1 Temporal

La inialización de la caché del dp es $O(C)$ pero $C \leq N$ en este caso El algoritmo hace un bucle por todos n elementos y las operaciones son en $O(1)$ por lo tanto el coste es $O(n)$

6.11.2 Espacial

La inialización de la caché del dp es $O(C)$ pero $C \leq N$.