Introducción a la programación dinámica

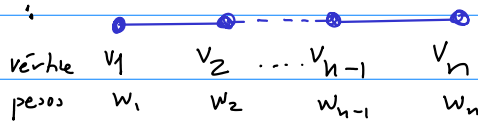
La PROGRAMACIÓN DINÁMICA es un paradigma para resolver problemas de optimización en familias, la explicaremos mediante un ejemplo

PROBLEMA: [Weighted independent set] (WIS) Sea G un grafo no dirigido con pesos $w \geq 0$ en los vértices.

$$WIS(G) = \max \{ \bar{w}(S) : S \subseteq V(G), S \text{ independiente} \}$$

donde $\bar{w}(S) := \sum_{s \in S} w(s)$.

Para explicar programación dinámica nos enfocaremos en "cadenas" G_n:



usando programación dinámica construiremos un algoritmo muy eficiente (de tiempo $O(m+n)$) para resolver WIS en cadenas (aunque G_n tiene una cantidad exponencial de conjuntos independientes)

IDEA CLAVE: Preguntamos: Si tuviéramos una solución óptima, ¿qué propiedades tendría? ¿Cómo se relaciona con soluciones óptimas de problemas más pequeños?

Suponga que S^* es una solución óptima, es decir $\bar{w}(S^*) = WIS(G_n)$ y nos preguntamos si el último vértice v_n pertenece o no a S^*

$S^* \subseteq G_n$ independiente con $\bar{w}(S^*) = WIS(G_n)$.

¿ $v_n \in S^*$?

NO:

Todos los vértices de S^* pertenecen a G_{n-1} así que S^* es óptimo para G_{n-1}

(de lo contrario, si hubiera $T \subseteq G_{n-1}$ T independiente con $\bar{w}(T) > \bar{w}(S^*)$

$T \subseteq G_n$ lo que contradice la optimalidad de S^*)

$\bar{w}(S^*) = WIS(G_{n-1})$

SI

$v_{n-1} \notin S^*$ (porque S^* es indep.)

luego $S' := S^* \setminus \{v_n\} \subseteq G_{n-2}$

y S' es óptimo en G_{n-2}

(de lo contrario $\exists\ T \subseteq G_{n-2}$ independiente con $\bar{w}(T) > \bar{w}(S')$

luego $T \cup \{v_n\}$ es independiente en G_n y $\bar{w}(T \cup \{v_n\}) = w(v_n) + \bar{w}(T)$ (V) $w(v_n) + \bar{w}(S')$

contradice la optimalidad de S^* " $\bar{w}(S)$

$\bar{w}(S^*) = w(v_n) + WIS(G_{n-2})$

Como $\bar{w}(S^*) = WIS(G_n)$ concluimos que los valores óptimos de los problemas WIS en cadenas tienen la siguiente relación, llamada **ECUACIÓN DE BELLMAN**: (*)

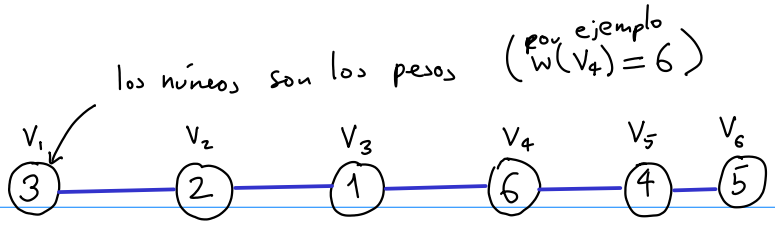
Lema: Si $n \geq 2$ entonces

(*) $WIS(G_n) = \max \{ WIS(G_{n-1}),\ w(v_n) + WIS(G_{n-2}) \}$

Este lema es muy útil porque, si sabemos $WIS(G_1)$ y $WIS(G_2)$ (fácil), la fórmula (*) nos dice como calcular $WIS(G_3)$, $WIS(G_4)$, ...

Veámoslo en un ejemplo concreto (página siguiente ...

G_6 :



$WIS(G_1) = 3$

$WIS(G_2) = 3$

$wis(G_3) = \max(WIS(G_2),\ \overset{w(v_3)}{\downarrow}\ 1 + WIS(G_1)) = \max(3, 4) = 4$

$wis(G_4) = \max(4,\ 6 + wis(G_2)) = \max(4, 9) = 9$

$wis(G_5) = \max(9,\ 4 + 4) = 9$

$wis(G_6) = \max(9,\ 5 + 9) = 14$

Concluimos que el máximo peso de un subconjunto independiente es 14. ¿Cómo encontr un conjunto indep con ese peso?

Marcamos en — arriba en cuál de las dos alternativas se alcanza el máximo y eso nos dice qué hacer, empezando por V_n y caminando hacia atrás...

Concretamente: S^*:

G_6 $\{V_6\}$

G_5 $\{V_6\}$

G_4 $\{V_6, V_4\}$

G_3 $\{V_6, V_4, V_3, V_1\}$

"Adicionamos un vértice cuando el máximo se alcanza a la derecha."
(el caso verde en nuestro análisis)

En general, tenemos la siguiente conclusión:

Para cualquier grafo cadena con pesos positivos el siguiente algoritmo produce un conjunto independiente de máximo peso: WIS-Cadena es fácil!

(1) Usamos la recursión de Bellman para calcular un vector

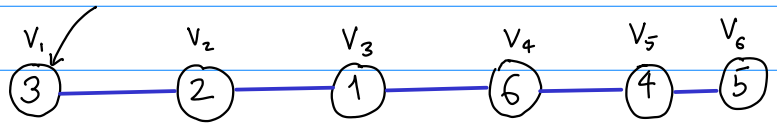
$$A[i] = WIS(G_i) \quad i = 1, \ldots, n$$

(2) Recorremos el arreglo A de adelante hacia atrás seleccionando los vértices del conjunto independiente maximal.

Tiempo $O(n)$, MUY RÁPIDO.

Pensemos ahora en una implementación.

los números son los pesos (por ejemplo $w(v_4) = 6$)



Sabemos los valores iniciales y la recursión:

$WIS(③) = 2$

$WIS(③—②) = 3$

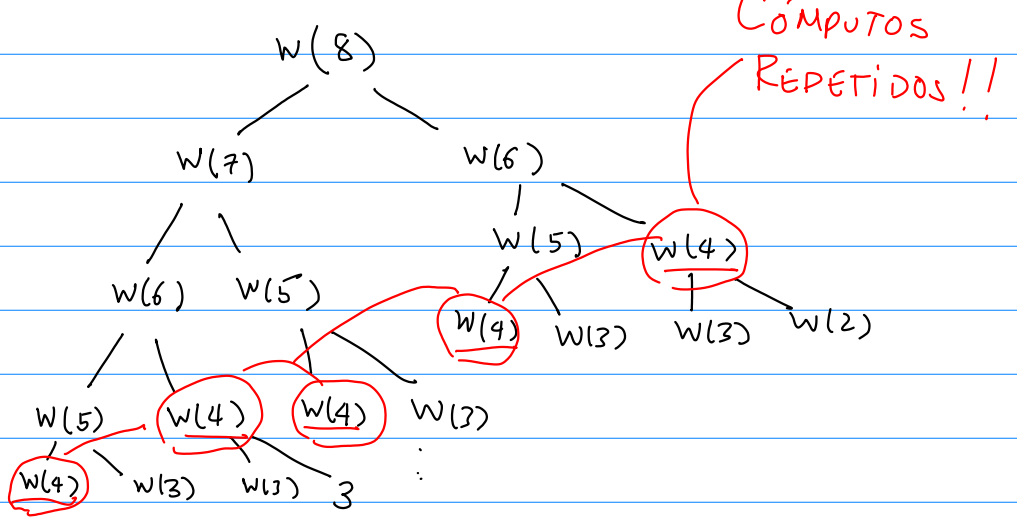
$WIS(G_n) = \max\left(WIS(G_{n-1}),\ w(v_n) + WIS(G_{n-2})\right)$

Idea 1: [MALA IDEA, CUIDADO!!]

Hagamos una recursión

```
def WIS(n):
    if n==1:
        return 2
    if n==2:
        return 3
    else
        return max(WIS(n-1), w(v_n) + WIS(n-2))
```

POR QUÉ ES MALA IDEA:



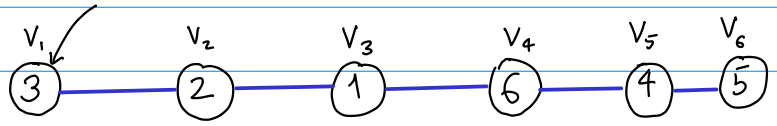
IDEA 2: [BUENA MEMOIZATION]

Lo que deberíamos hacer es calcular cada $WIS(G_t)$ UNA VEZ y reutilizar ese cálculo todas las veces. ¿Cómo hacer esto? Fácil, debemos responder a la pregunta

* PREGUNTA CLAVE: Dado el grafo con pesos G

(por ejemplo $w(v_4) = 6$)

los números son los pesos

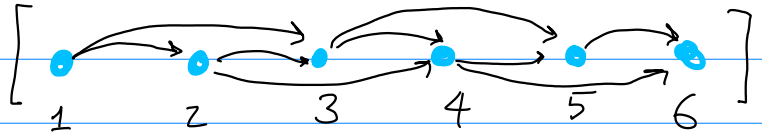


qué información necesito para saber qué subproblema estoy resolviendo?

$$WIS(G_n) = WIS(n)$$

Conociendo G todo el problema queda especificado mediante el ENTERO n

n determina completamente el "estado" de nuestra búsqueda



Este primer ejemplo funcionó MUY BIEN porque los subproblemas venían ordenados, lo que simplificó fuertemente la aplicación del método

$$G_1 \subseteq G_2 \subseteq \dots \subseteq G_n$$

La programación dinámica se puede aplicar a muchos, muchos otros problemas.

PROBLEMA 2: [Knapsack o "de la maleta"]

Tenemos una tabla de ítems que tienen un cierto volumen y un **valor**

item	volúmen	valor	
1	h_1	p_1	
2	h_2	p_2	
⋮	⋮	⋮	
n	h_n	p_n	

y tenemos una maleta con capacidad total C.

¿Cuál es la colección de ítems más valiosa que cabe en la maleta?

$$\text{Knapsack}(n, C) = \max\left\{ \bar{p}(S) : S \subseteq [n] \;\middle|\; \sum_{s \in S} h_s \leq C \right\}$$

($[n] = \{1, 2, \ldots, n\}$!!)

con $\bar{p}(S) = \sum_{s \in S} p_s$.

los ítems caben en la maleta

Aplicamos la idea clave:

¿Si $S^* \subseteq [n]$ es una selección óptima contiene al ítem n-ésimo?

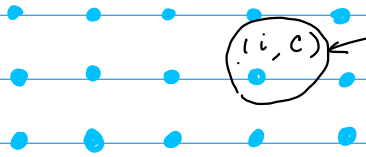
NO SI

Lema: [Ecuación de Bellman para Knapsack]

$$K(n, C) = \max\left(K(n-1, C),\ p_n + K(n-1, C - h_n)\right)$$

Los "estados" forman un conjunto mucho más interesante (n, C)

IMPLEMENTACIÓN



Hemos visto los i primeros ítems y tenemos disponible capacidad C.