10. Tenemos cajas numeradas de 1 a N, todas de iguales dimensiones. Queremos encontrar la máxima cantidad de cajas que pueden apilarse en una única pila cumpliendo que:

- sólo puede haber una caja apoyada directamente sobre otra;
- las cajas de la pila deben estar ordenadas crecientemente por número, de abajo para arriba;
- cada caja i tiene un peso w_i y un soporte s_i, y el peso total de las cajas que están arriba de otra no debe exceder el soporte de esa otra.

Si tenemos los pesos $w = [19, 7, 5, 6, 1]$ y los soportes $s = [15, 13, 7, 8, 2]$ (la caja 1 tiene peso 19 y soporte 15, la caja 2 tiene peso 7 y soporte 13, etc.), entonces la respuesta es 4. Por ejemplo, pueden apilarse de la forma 1-2-3-5 o 1-2-4-5 (donde la izquierda es más abajo), entre otras opciones.

---

a) Pensar la idea de un algoritmo de *backtracking* (no hace falta escribirlo).

✓

---

b) Escribir una formulación recursiva que sea la base de un algoritmo de PD. Explicar su semántica e indicar cuáles serían los parámetros para resolver el problema.

$$\mathrm{PC}_{ws}(i, \mathrm{acc}) = \begin{cases} 0 & \text{si } i < 1 \\ \mathrm{PC}_{ws}(i-1, \mathrm{acc}) & \text{si } i \geq 1 \land s_i < \mathrm{acc} \\ \max\{\mathrm{PC}_{ws}(i-1, \mathrm{acc}), 1 + \mathrm{PC}_{ws}(i-1, \mathrm{acc} + w_i)\} & \text{si } i \geq 1 \land s_i \geq \mathrm{acc} \end{cases}$$

Recorremos "de adelante para atrás"

El caso base no suma nada, es solo para definir un final, ya pasamos todos los $i \in \{1...N\}$.

El primer caso recursivo define si superamos el peso que la caja i puede cargar, luego, no suma nada y sigue sin tener en cuenta esa caja.

El segundo caso recursivo es el que nos suma 1 si i puede cargarlo, entonces suma w_i al acumulador y continúa la recursión.

El problema se resuelve con $\mathrm{PC}_{ws}(N, 0)$

---

c) Diseñar un algoritmo de PD y dar su complejidad temporal y espacial auxiliar. Comparar cómo resultaría un enfoque *top-down* con uno *bottom-up*.

Top-down

Primero definimos $\mathrm{acc}_{\max} = \sum_{i=1}^{N} w_i$, luego

```
f solve(w,s,N):- Sea $k$ el resultado, las cajas están apiladas de la forma ${i_1 ...
i_k} | i_1 < ... < i_k$
- $ forall j in {1...k-1}: (sum_(t=j+1)^k w_i_t ) <= s_i_j $

  memo = matriz N*acc_max con todo -1

  f pc(i,acc):
    if i = 0:
      ret 0

    if memo[i][acc] >= 0:
```

```
      ret memo[i][acc]

    if s[i] < acc:
      memo[i][acc] = pc(i-1,acc)
    if s[i] >= acc:
      memo[i][acc] = max(pc(i-1,acc), pc(i-1,acc+w[i])+1)

    return memo[i][acc]

  ret pc(N,0)
```

Su complejidad espacial y temporal están limitadas por la cantidad de estados, $O\left(N \cdot \sum_{i=1}^{N} w_i\right)$

Bottom-up

```
f solve(w,s,N):

  memo = [-1] de N*acc_max
  for acc en 0...acc_max:
    memo[0][acc] = 0

  for i in 1...N:
    for acc in acc_max...0:

      if acc+w[i] <= acc_max:
        use = 1 + memo[i-1][acc+w[i]]

      if acc+w[i] > acc_max:
        use = 0

      memo[i][acc] = max(memo[i-1][acc], use)
```

---

d) (Opcional) Formalizar el problema y demostrar que la función recursiva es correcta.

Tenemos:

$N \in \mathbb{N}$ cantidad de cajas.

Sea $i \in \{1...N\}$ la notación de la i-ésima caja, hay 2 vectores:

$w = \{w_1...w_N\}$ donde $w_i \in \mathbb{N}$ es el peso de la i-ésima caja.

$s = \{s_1...s_N\}$ donde $s_i \in \mathbb{N}$ es el aguante de la i-ésima caja.

Nuestro objetivo es encontrar el máximo número de cajas que podemos apilar tal que:

- Sea k el resultado, las cajas están apiladas de la forma $\{i_1...i_k\} \mid i_1 < ... < i_k$
- $$\forall j \in \{1...k-1\} : \left(\sum_{t=j+1}^{k} w_{i_t}\right) \leq s_{i_j}$$

Queremos ver que $\text{PC}_{ws}(i, \text{acc})$ es correcta

$$\text{PC}_{ws}(i, \text{acc}) = \begin{cases} 0 & \text{si } i < 1 \leftarrow \{B\} \\ \text{PC}_{ws}(i-1, \text{acc}) & \text{si } i \geq 1 \land s_i < \text{acc} \leftarrow \{R1\} \\ \max\{\text{PC}_{ws}(i-1, \text{acc}), 1 + \text{PC}_{ws}(i-1, \text{acc} + w_i)\} & \text{si } i \geq 1 \land s_i \geq \text{acc} \leftarrow \{R2\} \end{cases}$$

Donde i es la caja actual y acc es el peso acumulado de las cajas sobre i

Vamos a demostrarlo por inducción en i

Caso base: $i < 1$

Tenemos 0 cajas

$\mathrm{PC}_{ws}(0, \mathrm{acc}) \underset{\{B\}}{=} 0$

Paso inductivo:

HI: Existe un $i = k$ tal que $\mathrm{PC}_{ws}(k-1, \mathrm{acc})$ es el máximo de cajas apilables hasta la k-ésima caja

Queremos ver que $\forall$ acc $\in \mathbb{N}.\mathrm{PC}_{ws}(k, \mathrm{acc})$

Caso R1: $s_k < \mathrm{acc}$

Si $s_k < \mathrm{acc}$ no puedo apilar las cajas acumuladas sobre k, por lo que no sumamos nada y seguimos a la siguiente caja.

$\mathrm{PC}_{ws}(k, \mathrm{acc}) = \mathrm{PC}_{ws}(k-1, \mathrm{acc}) \underset{\mathrm{HI}}{=} V$

Caso R2 $s_k \geq \mathrm{acc}$

subcaso 1, no apilamos k:

Mantenemos el mismo número de cajas apiladas como en el caso **R1**

$\mathrm{PC}_{ws}(k, \mathrm{acc}) = \mathrm{PC}_{ws}(k-1, \mathrm{acc}) = \mathrm{S1}$

subcaso 2, apilamos k:

Apilamos la k-ésima caja

$\mathrm{PC}_{ws}(k, \mathrm{acc}) = \mathrm{PC}_{ws}(k-1, \mathrm{acc} + w_k) + 1 = \mathrm{S2}$

Justificación

R2 devuelve el máximo entre S1 y S2, por HI sabemos que hasta $k-1$ teníamos el máximo acumulable, se evalúan todas las chances, para todo k. Luego $\mathrm{PC}(k, \mathrm{acc})$ computa máximo número de cajas apiladas hasta la k-ésima caja.

$\square$