8. Debemos cortar una vara de madera en varios lugares predeterminados. Sabemos que el costo de realizar un corte en una madera de longitud ℓ es ℓ (y luego de realizar ese corte quedarán 2 varas de longitudes que sumarán ℓ). Por ejemplo, si tenemos una vara de longitud 10 metros que debe ser cortada a los 2, 4 y 7 metros desde un extremo, entonces los cortes se pueden realizar, entre otras maneras, de las siguientes formas:

   - Primero cortar en la posición 2, después en la 4 y después en la 7. Esta resulta en un costo de $10 + 8 + 6 = 24$ porque el primer corte se hizo en una vara de longitud 10 metros, el segundo en una de 8 metros y el último en una de 6 metros.

   - Cortar primero donde dice 4, después donde dice 2, y finalmente donde dice 7, con un costo de $10 + 4 + 6 = 20$, que es menor.

Queremos encontrar el mínimo costo posible de cortar una vara de longitud ℓ.

---

a) Convencerse de que el mínimo costo de cortar una vara que abarca desde i hasta j con el conjunto C de lugares de corte es $j - i$ mas el mínimo, para todo lugar de corte c entre i y j, de la suma entre el mínimo costo desde i hasta c y el mínimo costo desde c hasta j.

✓

---

b) Escribir matemáticamente una formulación recursiva basada en *a*). Explicar su semántica e indicar cuáles serían los parámetros para resolver el problema.

$C = \{c_1 ... c_n\}, c_1 < ... < c_n$

$$\mathrm{CE}_C(i,j) = \begin{cases} 0 & \text{si } \nexists c \in C \mid i < c < j \\ j - i + \min_{(c \in C) \xrightarrow[L]{} (i<c<j)} \begin{cases} \mathrm{CE}_C(i,c) \\ \mathrm{CE}_C(c,j) \end{cases} & \text{sino} \end{cases}$$

Se resuelve con $\mathrm{CE}_C(0, |C| + 1)$

---

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

```
f solve(C):
  memo = []
  f CE(i,j):
    si no existe C[k] :: i<k<j: //O(1)
      ret 0

    si existe memo[i][j]:
      ret memo[i][j]

    min_costo = inf

    para cada k in [1...n]: //O(n)
      si i < C[k] < j:
        min_costo = min(min_costo, j-i + min(CE(i,C[k]),CE(C[k],j)))
```

```
    memo[i][j] = min_costo
    ret memo[i][j]
```

Complejidad espacial, $(0 \leq i, j \leq n) \Rightarrow O(n^2)$

Cada nodo del árbol de recursión es $O(n)$, y por `memo`, hay máximo n^2 llamados, luego es $O(n^3)$

Bottom-up

```
f solve(C):
  para cada l en [1...n]:
    para cada i en [0...n-l]:
      j = i+l
      memo[i][j] = -inf
```

---

d) Supongamos que se ordenan los elementos de C en un vector *cortes* y se agrega un 0 al principio y un ℓ al final. Luego, se considera que el mínimo costo para cortar desde el i-ésimo punto de corte en *cortes* hasta el j-ésimo punto de corte será el resultado buscado si $i = 1$ y $j = |C| + 2$.

I) Escribir una formulación recursiva con dos parámetros que esté basada en *d*) y explicar su semántica.

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