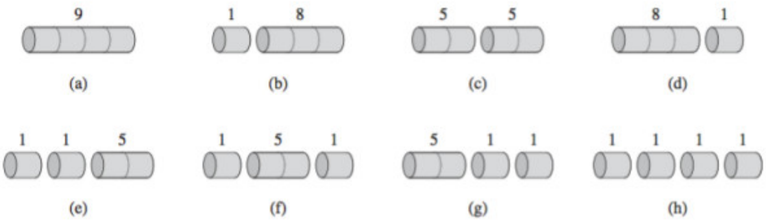
Exercici 2 (3 punts)

Els recursos naturals són cada vegada més escassos i hem de reduir-ne i optimitzar-ne l'ús. A la indústria metal·lúrgica això és especialment important perquè treballen principalment amb minerals com a matèria primera. Per aquesta raó hi ha molts processos d'optimització que es desenvolupen i s'apliquen en aquest àmbit. A nivell econòmic, sovint això implica haver de treure el màxim profit econòmic de la quantitat de recursos disponible en un determinat moment.

Tenim una gran barra d'acer de longitud n, i volem tallar-la en trossos per a destinar-los a diferents usos. Cada tros de barra d'acer, en funció de la seva llargària, té un cost al mercat. Un tros de llargària i, amb $i \in \mathbb{Z}^+$ i $1 \leq i \leq n$, val p_i euros. Observeu que les llargàries dels trossos són unitats enteres.

La figura següent en mostra un exemple amb les 8 possibles formes de tallar una barra d'acer de llargària 4. Sobre cadascun dels trossos s'indica el preu que se n'obté ($\{p_1 = 1, p_2 = 5, p_3 = 8, p_4 = 9\}$). La solució òptima és l'opció (c) –tallar la barra en dos trossos de longitud 2– que dóna un guany de 10€.



Es demana:

a) De quantes formes diferents es pot tallar una barra de llargària n?
Raoneu la resposta.

b) Dissenyeu un algorisme, el més eficient que pugueu, per a decidir com tallar una barra d'acer de longitud n en trossos, de manera que es maximitzi el guany total que se n'obté. El vostre algorisme ha de dir quants talls s'han de fer en total, i a on.

Assumirem que fer un tall a la barra no indueix cap cost afegeix. Observeu que no es demana cap requisit sobre el nombre de talls a fer; podeu fer qualsevol nombre de talls entre 0 i $n - 1$.

Una solució:

a) 2^{n-1}, perquè hi ha $n - 1$ llocs on podem triar fer talls, i a cada lloc, o fem un tall o no el fem.

b) Un algorisme naïf per a resoldre aquest problema exploraria les $O(2^n)$ formes de tallar la barra (apartat (a)) i retornaria la que maximizés el guany obtingut. Naturalment, aquesta solució no és opció per a nosaltres. Podem resoldre'l de manera més eficient si apliquem programació dinàmica. Podem fer-ho perquè el problema presenta les dues condicions necessàries: subestructura òptima,[2] i problemes superposats .[3]

Representem una solució òptima a un subproblema mitjançant la següent recurrència: sigui $S(i)$ el preu màxim que podem obtenir en tallar un tros de la barra de llargada i que comprèn l'inici de la mateixa (és a dir, el tros que va des de l'inici fins als i metres). Segons aquesta definició, la solució al problema queda representada pel terme $S(n)$ de la recurrència. Tenint en compte la subestructura òptima, la resta de solucions òptimes parcials es calculen de la següent forma:

[2]Fixat un punt de tall a la barra, la solució òptima que s'hi pot aconseguir requereix resoldre de manera òptima els dos nous trossos que aquest punt de tall produeix a la barra. Altrament sempre podríem crear una solució millor i, per tant, la considerada no seria òptima.

[3]En el plantejament recursiu del problema ens trobem que s'ha de resoldre diverses vegades el mateix subproblema.

$$S(i) = \begin{cases} 0, & \text{si } i = 0 \\ \max\limits_{0 \le j < i} \{S(j) + p_{i-j}\}, & \text{altrament} \end{cases}$$

Observeu que, al cas recursiu, el que es fa es cercar el millor *punt de tall* j on fer l'últim tall a la barra (tros des de $j+1$ a i, que té llargària $i-j$).

Per implementar aquesta solució i poder guardar les solucions dels subproblemes necessitem una taula de mida n, on la posició i enmagatzemarà el valor de $S(i)$. En total s'han de resoldre n subproblemes i cadascun d'ells requereix temps $O(n)$. El cost en espai és, doncs, $O(n)$ i el cost temporal $O(n^2)$.

Se'ns demana també reconstruir la solució, és a dir, explicitar quants talls s'han de fer i a quines posicions per obtenir el benefici màxim que calcula la recurrència. Això ho podem fàcilment amb una estructura auxiliar d'espai $O(n)$ que guardi memòria, per al càlcul de cadascun dels $S(i)$, de quina ha estat la j (punt de tall) que ha produït el màxim. Possible implementació:

```
funció BottomUpTallaBarraComplet (p, n)
    S[0..n] ← {0, 0, .., 0}          ▷ Vector que guarda els valors de la recurrència
    T[0..n]                          ▷ Vector on guardarem els talls
    per a i = 1 to n fer
        q = −∞
        per a j = 1 to i fer
            si q < S[j] + p[i − j] aleshores
                q = S[j] + p[i − j])                  ▷ Arrosseguem el màxim
                T[i] = j     ▷ En acabar el bucle, (T[i] = j)⟹(j = arg_max{S(j) + p_{i−j}})
        S[i] = q
    retornar S, T

funció PrintSolution (p, n)
    (S, T) ←BottomUpTallaBarraComplet(p, n)
    mentre n > 0 fer
        Print(T[n])
        n = n − T[n]
```

NOTA: De manera anàloga, es podria plantejar la solució simètrica, és a dir, que $S(i)$ fos el preu màxim que es pot obtenir en tallar un tros de la barra que va des de la posició i fins al final (tros de llargada $n-i$). En aquest cas, el punt de tall es buscaria entre i i n.

Una altra solució:

Podem també representar una solució òptima a un subproblema mitjançant la següent recurrència: sigui $S(i, j)$ el preu màxim que podem obtenir en tallar un tros de la barra comprès entre les posicions i i j, considerant que es talla *després* de la posició i. Aquesta representació és més genèrica i ens permet considerar com a subproblema qualsevol tros de la barra, des de qualsevol posició.[4] Com veurem, això no ens aporta cap benefici.

Segons aquesta definició, la solució al problema queda representada pel terme $S(0, n)$ de la recurrència. Tenint en compte la subestructura òptima, la resta de solucions òptimes parcials es calculen de la següent forma:

$$S(i,j) = \begin{cases} 0, & \text{si } i = j = 0 \\ p_j, & \text{si } j > i = 0 \\ \max\limits_{i \le k < j} \{S(i,k) + p_{j-k}\}, & \text{altrament} \end{cases}$$

Amb aquesta representació hem de resoldre un nombre de subproblemes d'ordre $O(n^2)$ i cadascun d'ells requereix temps $O(n)$. El cost total necessari és, doncs, $O(n^3)$. A més, en implementar-ho, necessitarem fer ús d'un espai també d'ordre $O(n^2)$. Aquesta solució és, doncs, pitjor que la primera proposada, tant en temps com en espai.

[4]En contraposició a la primera solució proposada, que sempre considerava un tros de la barra original però començant des del principi de la barra (els "prefixos" de la barra).