

**Devoir 4 (partiel)**

Remise : le vendredi 13 avril

Notez : le numéro 4 (sur les algos probabilistes) de ce devoir viendra plus tard ; nous verrons le retour arrière en cours le mardi 3 avril

*Due Friday April 13th ; a fourth problem (out of the material from Chapter 10) will come later ; we will discuss backtracking on Tuesday April 3rd*

1. Une fonction  $f$  est fournie, qui associe à chaque suite de lettres de  $\{a, b, c, d, e, f, g, \dots, y, z\}$ , par exemple à *summer*, un naturel qui estime la fréquence à laquelle cette suite risque d'apparaître comme un mot dans un texte en anglais. Décrivez en pseudo-code de votre choix un algorithme de programmation dynamique qui résout

SEGMENTATION

**DONNÉE:** suite de lettres  $w = x_1 \cdot x_2 \cdot \dots \cdot x_n$

**CALCULER:** valeur d'une segmentation optimale de  $w$  obtenue par insertion d'espaces, i.e.,  $\max \left\{ \sum_{i=1}^k f(w_i) : w = w_1 \cdot w_2 \cdot \dots \cdot w_k \right\}$ , où “.” représente la concaténation.

Par exemple, si  $w$  était *helloworld* alors la segmentation optimale serait vraisemblablement  $w_1 = \text{hello}$  et  $w_2 = \text{world}$ .

2. Vous organisez une compétition culinaire dans une salle gigantesque alimentée en électricité par une pile solaire. Chaque chef participant requiert une heure d'éclairage. Un chef est disponible à une heure fixe et à cette heure seulement. Vous connaissez à l'avance le nombre  $n_i$  de chefs disponibles à l'heure  $i$ ,  $1 \leq i \leq k$ . (Plusieurs chefs peuvent accéder à la salle en même temps.)

À l'heure 0, votre pile est vide. Son fabricant Soleillo-Québec vous assure toutefois que  $c(j)$  chefs peuvent être éclairés pendant une heure complète dès que la pile a eu droit à  $j$  heures de recharge ininterrompue (i.e., sans utilisation),  $1 \leq j \leq k$ . Après une heure d'utilisation, la pile est déchargée complètement, quel que soit son niveau de charge au début de l'heure.

Donnez un algorithme efficace qui prend en entrée les  $n_i$  et qui fait appel à la fonction  $c$  dans le but de calculer les moments (par exemple, aux heures 1, 4 et 12) où solliciter la pile de manière à maximiser le nombre total de chefs qui pourront participer à la compétition.

*Indice.* À la toute dernière heure, c'est à dire à l'heure  $k$ , quel est le nombre de chefs admis si la charge de la pile au début de l'heure est  $k$  ? Si cette charge est  $k - 1$  ? Si elle est 1 ? Si elle est 0 ? Que pouvez-vous calculer par la suite ?

3. Appelons *abaque* un tableau  $A \in \mathbb{N}^{2 \times m}$  tel que pour  $1 \leq i \leq m$ ,  $A(1, i) < i$  et  $A(2, i) < i$ . La *taille* de l'abaque est  $m$  et son *poids* est  $w(m)$ , défini comme suit :

$$w(i) = \begin{cases} 1 & \text{si } i = 0 \\ w(A[1, i]) + w(A[2, i]) & \text{si } 0 < i \leq m. \end{cases}$$

- (a) Donnez en Python un algorithme qui prend en entrée un tableau  $2 \times m$  et qui retourne le poids de l'abaque, ou  $-1$  si le tableau n'est pas un abaque.
  - (b) Donnez en Python un algorithme à retour arrière qui, étant donné  $w \in \mathbb{N}^{\geq 2}$ , retourne un abaque de poids  $w$  et de taille minimale.
-

1. A function  $f$  assigns to every sequence of letters from  $\{a, b, c, d, e, f, g, \dots, y, z\}$ , such as *summer*, a natural number that is roughly proportional to the frequency at which this sequence actually shows up as a word in a some large English body of text. Describe in a pseudo-code of your choice a dynamic programming algorithm that solves

SEGMENTATION

**DONNÉE:** a sequence  $w$  as above

**CALCULER:** value of an optimal segmentation of  $w$  obtained by the insertion of blanks,

i.e.,  $\max \left\{ \sum_{i=1}^k f(w_i) : w = w_1 \cdot w_2 \cdot \dots \cdot w_k \right\}$ , where “.” represents concatenation.

For example, if  $w$  is *helloworld* then the optimal segmentation would likely be  $w_1 = \text{hello}$  and  $w_2 = \text{world}$ .

2. You are planning a culinary competition in a gigantic solar-powered room. Each chef requires one hour of lighting. A chef is available for one hour at a fixed hour and at no other time. You know ahead of time the number  $n_i$  of chefs that are available at the  $i$ th hour,  $1 \leq i \leq k$ . (Any number of chefs may access the room simultaneously.)

At the 0th hour, your solar battery is empty. Its manufacturer Soleillo-Québec guarantees however that the lights for  $c(j)$  chefs can be powered during a complete hour once the battery has received  $j$  hours of uninterrupted recharge (i.e., during which it is not being used),  $1 \leq j \leq k$ . After one hour of use, the battery is fully discharged, regardless of its charge level at the beginning of the hour.

Give an efficient algorithm that takes as inputs the  $n_i$  and that calls the function  $c$  in order to compute the times (for example, hours 1, 4 and 12) at which to solicit the battery in order to maximize the total number of chefs that will be able to take part in your competition.

*Hint.* At the very last hour, that is, at the  $k$ th hour, how many chefs are admitted if the charge level of the battery at the beginning of that hour is  $k$ ? What if the level is  $k - 1$ ? Or 1? Or 0? What do these numbers allow you to compute next?

3. Say an *abacus* is an array  $A \in \mathbb{N}^{2 \times m}$  having the property that for  $1 \leq i \leq m$ ,  $A(1, i) < i$  and  $A(2, i) < i$ . The *size* of the abacus is  $m$  and its *weight* is  $w(m)$ , defined as follows :

$$w(i) = \begin{cases} 1 & \text{if } i = 0 \\ w(A[1, i]) + w(A[2, i]) & \text{if } 0 < i \leq m. \end{cases}$$

- (a) Give, and upload on Studium, a Python method that takes as input a  $2 \times m$  array and returns the weight of the abacus, or else  $-1$  if the array is not an abacus.
- (b) Give, and upload on Studium, a backtracking algorithm in Python that, given  $w \in \mathbb{N}^{\geq 2}$ , returns a minimum size abacus of weight  $w$ .