IFT 2125 Automne 2018 **Pierre McKenzie**
Introduction à l'algorithmique

Devoir 4

Remise : le vendredi 13 avril. *Due on Friday 13th.*

1. Une fonction f est fournie, qui associe à chaque suite de lettres de $\{a, b, c, d, e, f, g, \ldots, y, z\}$, 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 $w = x_1 \cdot x_2 \cdot \cdots \cdot x_n$

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

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

4. Votre voisine de classe n'a pas compris le calcul des $highest(v)$ servant à identifier les points d'articulation d'un graphe parce que le professeur explique mal. Elle propose plutôt un algorithme de Monte Carlo pour décider si un graphe connexe $G = (\{1, 2, \ldots, m\}, A)$ possède un point d'articulation :

   tirer au hasard $v \in \{1, 2, \ldots, m\}$
   effectuer une fouille en profondeur de G en partant du sommet v

```
si v possède deux fils distincts dans l'arbre de la fouille alors
      retourner VRAI
sinon
      retourner FAUX
```

(a) Identifiez un graphe de 4 sommets sur lequel cet algorithme ne fait jamais d'erreur.

(b) L'algorithme est-il faux-biaisé ? vrai-biaisé ? ni l'un ni l'autre ?

(c) Calculez en fonction de m la probabilité d'erreur de l'algorithme.

(d) En répétant l'algorithme, est-il possible de réduire la probabilité d'erreur à moins de 1% lorsque $m = 4$? $m = 10$? $m = 100$? (Si oui, combien faut-il de répétitions dans chacun des cas ?)

---

1. A function f assigns to every sequence of letters from $\{a, b, c, d, e, f, g, \ldots, y, z\}$, 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., $\max\left\{\sum_{i=1}^{k} f(w_i) \quad : \quad w = w_1 \cdot w_2 \cdot \cdots \cdot w_k\right\}$, where "$\cdot$" represents concatenation.

   For example, if w is *helloworld* then the optimal segmentation would likely be $w_1 =$ hello and $w_2 =$ 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 ith 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 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 sollicit the battery in order to maximize the total number of chefs that will able to take part in your competition.

   Hint. At the very last hour, that is, at the kth 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.

4. Your friend in class did not understand the *highest*(v) computation business used to locate the articulation points in a graph because the professor is a dud. She suggests instead a Monte Carlo algorithm to decide whether a connected graph $G = (\{1, 2, \ldots, m\}, A)$ contains an articulation point :

```
pick v ∈ {1, 2, ..., m} at random
perform a depth-first search of G starting at node v
if v has two distinct children in the depth-first search tree then
      return True
else
      return False
```

   (a) Give a 4-node graph on which the algorithm never errs.

   (b) Is the algorithm false-biased ? true-biased ? neither ?

   (c) Compute as a function of m the error probability of the algorithm.

   (d) By repeating the algorithm, is it possible to reduce the error probability to less than 1% when $m = 4$? $m = 10$? $m = 100$? (If so, how many repetitions are needed in each case ?)