Exercice 1. Arbre binaire de recherche optimal

Étant donnés des éléments $e_0 < e_1 < ... < e_{n-1}$ de probabilités d'apparitions p_0, ..., p_{n-1}, on veut construire un arbre binaire de recherche (ABR) a contenant e_0, ..., e_{n-1} et minimisant la complexité moyenne de recherche d'un élément (le *coût* $C(a)$ de a), qui est :

$$C(a) = \sum_{i=0}^{n-1} p_i(1 + prof(e_i))$$

où $prof(e_i)$ est la profondeur de e_i dans a

1. Écrire une fonction `cout` ayant un ABR a en argument et renvoyant $C(a)$ en complexité linéaire en le nombre de sommets de a. On supposera définie, dans cette question, une fonction `proba` renvoyant en O(1) la probabilité d'apparition d'un élément de l'ABR.

Dans le reste de l'exercice, on veut calculer l'ABR optimal (de coût minimum) par programmation dynamique. Pour cela on note:

- $w_{i,l} = p_i + ... + p_{i+l-1}$
- $c_{i,l}$ le coût de l'ABR optimal $a_{i,l}$ contenant e_i, ..., e_{i+l-1}

2. Donner une relation de récurrence « simple » sur $c_{i,l}$.
   Indice : que vaut $c_{i,l}$ si $a_{i,l}$ a pour racine e_{i+k}?
3. Écrire une fonction `opt` ayant un tableau des probabilités d'apparitions en entrée et renvoyant le coût de l'ABR optimal correspondant.
4. Quelle est la complexité de `opt`?
5. Modifier `opt` de façon à renvoyer aussi l'ABR optimal.

Exercice 2. Chemin de poids maximum dans une matrice

Étant donnée une matrice d'entiers $A = (a_{i,j})$ de taille $n \times k$, on veut connaître un chemin (n'utilisant que des déplacements $\rightarrow$ ou $\downarrow$) de la case en haut à gauche (d'indice $(0, 0)$) à la case en bas à droite (d'indice $(n-1, k-1)$) maximisant la somme des entiers rencontrés (le **poids** du chemin).
Voici un exemple avec un chemin de poids maximum en gras:

$$\begin{pmatrix} \mathbf{2} & \mathbf{39} & \mathbf{12} & \mathbf{49} & \mathbf{47} & 18 & 22 & 19 \\ 37 & 21 & 34 & 26 & \mathbf{10} & 2 & 35 & 39 \\ 31 & 21 & 12 & 26 & \mathbf{34} & \mathbf{27} & 7 & 22 \\ 20 & 46 & 16 & 2 & 11 & \mathbf{40} & \mathbf{36} & \mathbf{13} \\ 18 & 30 & 32 & 37 & 28 & 24 & 9 & \mathbf{6} \end{pmatrix}$$

1. Quelle serait la complexité d'un algorithme de recherche exhaustive, énumérant tous les chemins possibles de $(0, 0)$ à $(n-1, n-1)$? (on suppose pour simplifier que $n = k$, dans cette question)
2. Supposons qu'un chemin C de poids maximum de $(0, 0)$ à $(n-1, k-1)$ passe par la case (i, j). Montrer que le sous-chemin de C de $(0, 0)$ à (i, j) est de poids maximum (c'est une propriété de **sous-optimalité**).
3. Soit $p_{i,j}$ le poids maximum d'un chemin de $(0, 0)$ à (i, j). Donner une formule de récurrence sur $p_{i,j}$.
4. Écrire un algorithme récursif simple `p : int array array -> int * int -> int` tel que `p a (i, j)` renvoie le poids maximum d'un chemin de $(0, 0)$ vers (i, j) dans `a`. Que dire de sa complexité?
5. Écrire une fonction `pmax` donnant le poids maximum d'un chemin de la case en haut à gauche à la case en bas à droite, en utilisant une méthode par programmation dynamique. Comparer sa complexité avec la méthode précédente.
6. La fonction précédente ne donne que le poids maximum d'un chemin... Comment ferait-on pour trouver un chemin de poids maximum?

Exercice 3. Multiplication de matrices

1. Quel est le nombre de multiplications nécessaires pour calculer AB où A est une matrice de taille (n, p) et B de taille (p, q) (en utilisant la définition du produit matriciel) ?
2. Soit A, B, C de tailles $(2, 3)$, $(3, 4)$, $(4, 5)$.
   Combien de multiplications demande le calcul de $(AB)C$? (on fait d'abord le produit de A par B puis on multiplie par C). Combien de multiplications demande le calcul de $A(BC)$?

On veut calculer le produit $A_0 \times A_1 \times ... \times A_{n-1}$, où $A_0, A_1, ..., A_{n-1}$ sont de tailles $(t_0, t_1), (t_1, t_2), ..., (t_{n-1}, t_n)$.

3. Soit $m_{i,j}$ le nombre minimum de multiplications nécessaires pour calculer $A_i \times A_{i+1} \times ... \times A_j$.
   Donner une équation de récurrence sur $m_{i,j}$.
4. En déduire une fonction efficace `produit t` renvoyant $m_{0,n-1}$, où `t.(i)` contient t_i.

Exercice 4. Point fixe, de M. Bricout
Quelle est la meilleure complexité que vous pouvez donner pour trouver un point fixe (c'est-à-dire un indice `i` tel que `t.(i) = i`) dans un tableau `t` trié ?
Écrire l'algorithme correspondant.