Programmation dynamique: exercices

Exercice 1. *Carré de zéros dans une matrice binaire.* On considère une matrice de taille $n \times m$, constituée de zéros et de uns. On cherche à déterminer la taille maximale d'un carré dans la matrice, constitué uniquement de zéros. Par exemple, la matrice suivante possède un carré de zéros de taille 3×3 :

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Pour $0 \leq i < n$ et $0 \leq j < m$ on note $t_{i,j}$ la taille maximale d'un carré de zéro dans la matrice, dont le coin en bas à droite est indexé par (i, j) (on a donc notamment $t_{i,j} = 0$ si le coefficient en case (i, j) de la matrice est 1).

1. Relier $t_{i,j}$ à $t_{i-1,j}$, $t_{i,j-1}$ et $t_{i-1,j-1}$ pour $i, j \geq 1$.
2. En déduire une méthode efficace pour calculer la taille maximale d'un carré de zéros dans la matrice.
3. L'implémenter en Caml et donner sa complexité.

```
#a ;;
- : int array array =
 [|[|1; 0; 0; 1; 1; 0; 1; 1; 0|]; [|0; 1; 1; 1; 0; 1; 1; 0; 0|];
   [|0; 0; 0; 1; 0; 1; 0; 0; 0|]; [|0; 1; 1; 1; 0; 0; 0; 0; 0|];
   [|0; 0; 0; 0; 0; 1; 0; 0; 0|]; [|1; 1; 0; 0; 1; 0; 1; 1; 0|]|]
#taille_zeros a ;;
- : int = 3
```

Exercice 2. *Le problème de la portion de somme maximale.* On se donne un tableau `t` de taille n contenant des entiers, qui peuvent être positifs ou négatifs. Le but de l'exercice est de rechercher une portion du tableau, délimitée par deux indices $i \leq j$, telle que la somme `t.(i)` + `t.(i+1)` + $\cdots$ + `t.(j)` est maximale.

1. Résoudre le problème naïvement en testant toutes les possibilités : écrire une fonction `max_somme_naif t` retournant un couple d'indices (i, j) qui convient (on pourra écrire une fonction `somme_portion t i j`, et faire usage de références ensuite). Quelle est la complexité de cette approche ?

```
# let t_ex = [|-1; 2; 1; 4; -3; -5; 6; 2; -1; 6; 0; -1; -2; 2|] in max_somme_naif t_ex ;;
- : int * int = (6, 9)
```

2. Le reste de l'exercice est dévolu à la recherche d'une solution faisant usage de programmation dynamique. On note s_j une somme maximale de la forme `t.(i)` + `t.(i+1)` + $\cdots$ + `t.(j)`. Exprimer s_j en fonction de s_{j-1}.
3. En déduire un algorithme `max_somme_dyn t` permettant le calcul de tous les s_j, et renvoyant $s = \max_j s_j$. Quelle est sa complexité ?
4. Modifier l'algorithme pour qu'il retourne en plus un couple d'indice (i, j) qui convient, sans changer la complexité.

Exercice 3. *Distance de Levenshtein.* On définit la distance de Levenshtein entre deux mots u et v comme le nombre minimal $d(u, v)$ d'opérations nécessaires pour obtenir v à partir de u, les opérations autorisées étant la suppression, l'insertion, ou la modification d'une lettre. Par exemple, *mathematique* est à distance 5 de *arithmetique*, comme on le voit par la suite de transformations :

mathematique $\to$ mathmatique $\to$ mathmetique $\to$ aathmetique $\to$ arthmetique $\to$ arithmetique

1. Montrer que d est bien une distance, c'est à dire qu'elle vérifie :
   $d(u, v) = 0 \Leftrightarrow u = v \qquad \forall u, v,\ d(u, v) = d(v, u) \qquad$ et $\qquad \forall u, v, w,\ d(u, v) \leq d(u, w) + d(w, v)$.
2. Pour $0 \leq i \leq |u|$ et $0 \leq j \leq |v|$, on note $d_{i,j}$ la distance de Levenshtein entre les préfixes de u et de v de longueurs respectives i et j. Déterminer une expression de $d_{i,j}$ faisant intervenir $d_{i,j-1}$, $d_{i-1,j-1}$ et $d_{i-1,j}$.

3. En déduire un algorithme de calcul de la distance de Levenshtein.

```
# dist_levenshtein "mathematique" "arithmetique" ;;
- : int = 5
```

4. Modifier l'algorithme pour obtenir un *alignement* de séquences, c'est à dire une plus courte manière de passer de u à v. Coder effectivement une fonction `alignement u v` permettant de passer de `u` à `v`. (On utilisera `print_string` pour des affichages à l'écran).

```
# alignement "mathematique" "arithmetique" ;;
mathematique
mathemetique
mathmetique
athmetique
arthmetique
arithmetique
- : unit = ()
```

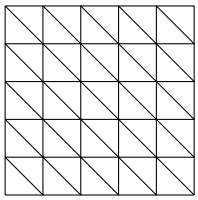
Exercice 4. *Rendu de monnaie.* On se donne $v_0, \dots, v_{n-1}$ des valeurs de pièces de monnaie, avec $v_0 = 1$. On peut supposer les v_i croissantes. On se donne une somme d'argent S à décomposer en les v_i (il y a au moins une solution car $v_0 = 1$), et on cherche à minimiser le nombre de pièces impliquées. En clair, on cherche à résoudre le problème :

$$\text{Trouver } (\alpha_0, \dots, \alpha_{n-1}) \text{ tel que } S = \sum_{i=0}^{n-1} \alpha_i v_i \quad \text{et} \quad \sum_{i=0}^{n-1} \alpha_i \text{ minimal.}$$

Dans un système monétaire usuel, il suffit de procéder en rendant le plus possible de pièces v_{n-1}, puis de pièces v_{n-2}, etc... Ce n'est pas le cas par exemple pour $(v_0, v_1, v_2) = (1, 3, 4)$, où pour $S = 6$ il vaut mieux utiliser deux pièces 3 que trois pièces 1, 1 et 4. Chercher une méthode de résolution du problème avec une complexité $O(Sn)$.

Exercice 5. *Problèmes de combinatoire.* La programmation dynamique peut être utilisée pour résoudre certains problèmes de combinatoire : il n'y a ici pas de quantité à optimiser (et d'extremum à trouver), mais la technique « tabulaire » permet de dénombrer efficacement.

1. Combien existe-il de pavages possibles d'un rectangle 50×2 par des dominos de taille 1×2 ?
2. * Même question avec un rectangle de taille 50×3.
3. Combien y a-t-il de chemins du coin supérieur gauche au coin inférieur droit du rectangle ci-dessous, en suivant uniquement les directions indiquées ?



Et pour un rectangle de taille 30×30 ?

4. *Project Euler 189, difficile...* Combien existe t-il de manières différentes de colorier le triangle de gauche avec trois couleurs, de sorte que deux petits triangles ayant une arête commune n'aient pas la même couleur (comme sur la figure de droite) ?