Algorithmique
TD n° 5 : Programmation dynamique

Exercice 1 : Construire un palindrome

On rappelle qu'un palindrome est une chaîne de caractères qui est égale à son miroir (par exemple *abba*). En insérant suffisamment de caractères dans une chaîne on peut la transformer en palindrome. Par exemple, en insérant un b à la fin, on peut transformer *baa* en palindrome *baab*.

1. Combien de caractères faut-il insérer pour obtenir un palindrome à partir de a, ab, $acecb$?

2. Soit $\mathbf{MinI}(w, g, d)$ le nombre minimal d'insertions pour obtenir un palindrome à partir du mot $w[g : d]$ (avec g et d entre 0 et $len(w) - 1$ et $g \leqslant d$). Soit $w = aceccgab$. Remplir le tableau suivant avec les valeurs de $\mathbf{MinI}(w, g, d)$:

g \ d	0	1	2	3	4	5	6	7
0								
1	×							
2	×	×						
3	×	×	×					
4	×	×	×	×				
5	×	×	×	×	×			
6	×	×	×	×	×	×		
7	×	×	×	×	×	×	×	

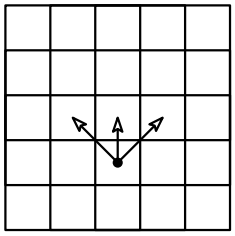
3. Donner une relation de récurrence pour $\mathbf{MinI}(w, g, d)$ (sans tester si une sous chaîne est un palindrome)

4. Trouver un algorithme de programmation dynamique pour ce problème.

5. Quelle est sa complexité ? Comparer celle-ci à la complexité d'un algo récursif basé sur les équations précédentes.

Exercice 2 : un jeu vidéo

Dans un jeu vidéo des années 1980, un vaisseau spatial se déplace à vitesse constante vers le haut (en fait, c'est le décor qui va vers le bas... mais peu importe !). Le joueur peut rester sur la même colonne, ou se déplacer d'une unité vers la gauche ou d'une vers la droite. Selon l'emplacement où il se trouve, il gagne des points, en perd, ou meurt.



On modélise le problème sous forme d'une matrice $\ell \times c$. À chaque case (i, j) est affecté un bonus/malus $b_{i,j}$ (valant $-\infty$ si le joueur meurt). Le but est donc de trouver le chemin qui rapporte le plus, sachant que seules sont accessibles depuis la case (i, j) les cases $(i+1, j-1), (i+1, j)$ et $(i+1, j+1)$, comme sur le dessin. Et que l'on s'arrête dans les cases de la ligne ℓ.

1. Trouver une relation de récurrence pour calculer le maximum de points que l'on peut gagner *depuis* la case (i, j).

2. Comment gérer les « effets de bord » de manière pratique pour que cette relation soit la même pour toutes les valeurs de j (même pour $j = 1$ et $j = c$) ?

3. En déduire un algorithme (de programmation dynamique) trouvant le score maximum à partir de n'importe quelle case. Quelle est sa complexité ?

4. Compléter l'algorithme pour qu'il propose un point de départ optimal et un chemin optimal correspondant.

Exercice 3 : plus longue sous chaîne commune

Une *sous chaîne commune* de deux chaînes de caractères s et t est une chaîne u telle que $s = s_1 \cdot u \cdot s_2$ et $t = t_1 \cdot u \cdot t_2$ avec s_1, s_2, u_1, u_2 des chaines (éventuellement vides). On parle aussi de facteur commun à s et t.
Par exemple, les sous chaînes communes de "abcde" et "acdeij" sont : "a", "c", "d", "e", "cd", "de", "cde".

1. Donner la ou les sous-chaines de taille maximale de *aaab* et *baabc*.

2. Donner deux chaînes de sorte qu'il y ait deux sous chaînes communes de taille maximale.

3. Donner un algorithme qui permet de vérifier qu'une chaîne est une sous chaîne d'une autre. Donner sa complexité.

4. Donner un algorithme naïf pour retrouver une des plus longues sous chaînes communes de deux chaînes.

5. Étant données deux chaînes s et t, soit $\mathsf{Lm}[i, j]$ la **longueur** de la plus longue sous chaîne commune (de s et de t) qui s'arrête à $s[i]$ et à $t[j]$. Soit $\mathsf{SCm}[i, j]$ la plus longue **sous-chaîne** commune (de s et de t) qui s'arrête à $s[i]$ et à $t[j]$. Soit $s =$ "*abcdeg*" et $t =$ "*deabceg*". Par exemple, on aura $\mathsf{Lm}[1, 3] = 2$ et $\mathsf{SCm}[1, 3] = ab$.
Remplir le tableau suivant à gauche avec les valeurs de $\mathsf{Lm}[i, j]$. Remplir le tableau à droite avec les valeurs de $\mathsf{SCm}[i, j]$.

Lm :

i \ j	0	1	2	3	4	5	6
0							
1							
2							
3							
4							
5							

SCm :

i \ j	0	1	2	3	4	5	6
0							
1							
2							
3							
4							
5							

6. Donner une récurrence pour $\mathsf{Lm}[i, j]$. Donner ensuite une récurrence pour $\mathsf{SCm}[i, j]$, la plus longue sous chaîne commune qui s'arrête à $s[i]$ et à $t[j]$.

7. Donner un algorithme de programmation dynamique pour trouver une des plus longues sous chaînes communes de deux chaînes.

8. Comment le modifier pour obtenir toutes les sous chaînes commune de longueur maximale ?