

Algorithmique

TD n° 5 : Programmation dynamique

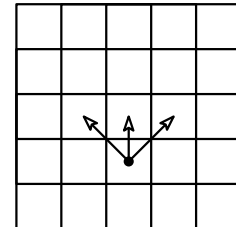
Exercice 1 : Construire un palindrome

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 $\min(w, g, d)$ le nombre minimal d'insertions pour obtenir un palindrome du mot $w[g : d]$ (avec g et d entre 0 et $\text{len}(w) - 1$)
Donnez une relation de récurrence pour $\min(w, g, d)$ de préférence sans utiliser d'auxiliaire pour tester si une sous chaîne est un palindrome.
3. Décrivez un algorithme récursif pour déterminer le nombre minimal d'insertion nécessaire pour transformer une chaîne de caractères en palindrome.
4. Qu'elle est sa complexité?
5. Trouver un algorithme de programmation dynamique pour ce problème.

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... 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.

1. Trouver une relation de récurrence pour calculer le maximum de points que l'on peut gagner en arrivant sur la case (i, j) . Comment gérer les « effets de bord » de manière pratique pour que cette relation reste vraie même pour $j = 1$ ou $j = c$?
2. En déduire un algorithme (de programmation dynamique) trouvant le score maximum. Quelle est sa complexité?
3. Améliorer l'algorithme pour qu'il affiche également le chemin à suivre.

Exercice 3 : La plus longue sous chaîne commune

Une sous chaîne (aussi appelé parfois facteur) commune de deux chaînes est une chaîne constituée des éléments communs des deux chaînes dans le même ordre dans les deux chaînes.

Par exemple, les sous chaînes commune de "abcde" et "cdeij" sont : "c", "d", "e", "cd", "de", "cde".

1. Donnez un algorithme qui permet de vérifier qu'une chaîne est une sous chaîne d'une autre.

2. Donnez un algorithme naïf pour retrouver la plus longue sous chaîne commune de deux chaînes.
3. Trouvez un algorithme de programmation dynamique pour trouver la plus longue sous chaîne commune de deux chaînes.
4. Comment le modifier pour obtenir toutes les sous chaînes commune de longueur maximale ?