TP n° 3 : corrigé

MP Lycée Camille Jullian

26 novembre 2021

Comme pour le TP précédent, les programmes de ce corrigé seront mis à disposition sous forme de fichier Python, seuls quelques commentaires sont présents dans ce document.

1 Exercice 1

J'ai écrit deux programmes distincts, un programme **lignetoiles** qui fait simplement une ligne de n étoiles, puis **croissetoiles** effectue la suite de lignes avec nombre d'étoiles croissant. J'ai tenté d'écrire une version récursive de lignetoiles, mais j'avoue y avoir lamentablement échoué, n'arrivant pas à gérer les print et les appels récursifs de façon à avoir tout qui s'affiche sur la même ligne. Tant pis, les programmes suivants fonctionnent quand même (**decetoiles** effectue la suite décroissante, une version récursive et une version classique, et **etoilesrec** répond à la dernière question posée, uniquement en version récursive).

2 Exercice 2

Le programme souslistes répond à la question posée en procédant de la façon suivante :

- on isole le premier terme de la liste, puis on calcule (récursivement) les sous-listes de ce qui reste dans notre liste
- pour chaque sous-liste de l[1 :], on recopie dans une liste m deux versions de la sous-liste, une telle quelle et une à laquelle on a rajouté le premier élément de la liste.

3 Exercice 3

Il faut bien gérer ici le cas où il faut aller chercher notre élément dans la deuxième moitié de la liste (la position à l'intérieur de la deuxième moitié de liste n'est évidemment pas la même que dans la liste initiale, d'où l'ajout de n dans le return de la ligne 74. Bien sûr, il ne faut ajouter ce n que si on trouve l'élément dans notre sous-liste, ce qui complique un peu la rédaction du programme.

4 Exercice 4

Ici, pas grand chose à commenter, les programmes sont assez simples.

5 Exercice 5

Si vous ne comprenez pas le principe de l'algorithme d'Euclide étendu, je vous incite à aller vous renseigner sur le web. Ici, la version récursive est nettement plus simple à programmer car elle ne nécessite pas autant de variables. Pour le reste, c'est assez classique : on teste la primalité en cherchant un diviseur de n entre 2 et $\sqrt{n}$ (pas besoin d'aller plus haut). Bien sûr on arrête le programme dès qu'on a trouvé un diviseur, et on renvoie True si on n'en trouve aucun. La recherche du plus petit facteur premier fonctionne exactement sur le même principe. Enfin, pour la décomposition en facteurs

premiers, on cherche le plus petit, on divise n par ce facteur, et on recommence, c'est un programme « naturellement » récursif.

6 Exercice 6

Encore des programmes assez simples si on a bien lu l'énoncé. On parcourt la liste de pièces disponibles jusqu'à en trouver une de valeur inférieure à n, on la met dans notre liste, et on recommence après avoir diminué la valeur de n. Comme indiqué dans l'énoncé, le programme peut ne pas être optimal en termes de nombres de pièces utilisées. Par exemple, si on a une liste L=[4,3,1] et qu'on cherche à rendre 6 euros, le progamme va proposer comme solution [4,1,1] alors que [3,3] est plus économique. Pour éviter ce problème, on peut tout simplement, au lieu d'imposer de prendre la première valeur inférieure à n recontrée, les tester toutes et garder la liste de longueur minimale parmi celles obtenues. Ce programme assez brutal est sans surprise de complexité exponentielle, donc lent dès qu'on augmente sensiblement la valeur de n. Mais il n'existe pas de solution rapide permettant de résoudre ce problème de façon optimale.

7 Exercice 7

L'énoncé expliquait en fait très mal l'algorithme puisqu'il oubliait la phase « backtracking » qui est parfois nécessaire : si on ne trouve rien qui marche en tentant l'ajout de x_1 puis celui de x_2, il faut modifier le dernier caractère de la chaine s (revenir une étape en arrière, en gros), et recommcner avec cette nouvelle chaîne. En admettant qu'on trouvera toujours une solution au problème, le programme proposé effectue exactement cela. Le programme **dijkstra** teste si la chaine de caractères s contient deux sous-chaînes identiques consécutives se terminant avec le dernier caractère de s (inutile d'en chercher avant, on les aurait repréré lors de la construction de la chaîne).