TD 7 Programmation Dynamique

Exercice 1. On souhaite savoir le nombre minimum d'opérations élémentaires permettent de transformer une chaîne de caractères en une autre. Les opérations élémentaires sont :

— supprimer un caractère,
— ajouter un caractère,
— remplacer un caractère par un autre.

Ce nombre est appelé *distance d'édition*.

1. Trouvez la distance d'édition entre `CHAT` et `CHEVAL`.
2. Trouvez la formule de récursion permettant de calculer la distance d'édition.
3. Donnez un algorithme permettant de la calculer ainsi que sa complexité.

Exercice 2. On se pose la questions suivante : comment rendre n euros en utilisant le moins de pièces possibles ? Pour fixer les choses, on considère qu'on dispose de pièces de valeurs 1, 2, 5, 10, 20, et 50.

Donnez un algorithme calculant le nombre minimal $P(n)$ de pièces nécessaires pour rendre n euros. Vous utiliserez la technique de programmation dynamique [1].

Par exemple, $P(29) = 4$, la solution optimale pour rendre 29 euros étant de rendre $1 \times 20, 1 \times 5, 2 \times 2$ euros soit 4 pièces au total.

Exercice 3. On cherche à écrire un algorithme de *pattern matching* (en français, *filtrage par motif*) pour les motifs suivants : étant donné une chaine de caractères s, et un motif p pouvant contenir les caractère spécial joker $*$, on veut savoir si le motif et la chaine correspondent, sachant que $*$ peut signifier n'importe quoi (aucun caractère, un caractère, plusieurs caractères ...)

Par exemple :

— $s = xyzzx, p = x * x$ est valide : $*$ va correspondre à yzz dans s
— $s = xyzzx, p = x * y$ n'est pas valide : la dernière lettre de la chaine et du motif sont différentes
— $s = xyzxyzxyz, p = xy * y * z$ est valide : par exemple, le premier joker $*$ peut correspondre à zx dans s, et le second à zxy

Donner un algorithme `Match(s, p)` où `s` est une chaine de caractères, `p` un motif contentant des caractères et des symboles joker $*$, et renvoyant `Vrai` si le motif `p` correspond à `s`, `Faux` sinon.

Exercice 4. Donner un algorithme comptant, pour l'entrée n un entier positif, toutes les chaines de longueur n sur l'alphabet $\{0,1\}$ ne contenant pas deux 1 consécutifs, en temps $\mathcal{O}(n)$.

Par exemple, pour $n = 5$, les suites binaires de longueur 5 sans 1s consécutifs sont [00000, 00001, 00010, 00100, 00101, 01000, 01001, 01010, 10000, 10001, 10010, 10100, 10101], et il en existe donc 13.

1. Il existe *ici* un algorithme plus simple, dit *glouton*, qui consiste à chaque étape à rendre la pièce de plus grosse valeur possible, et de continuer ainsi jusqu'à obtenir la bonne somme. Cet algorithme, optimal pour ces valeurs, ne l'est cependant pas pour tous les jeux de pièces : par exemple, avec les valeurs $\{1,3,4\}$ et en tentant de rendre 6, "glouton" se trompe, mais l'algorithme de programmation dynamique fonctionne cependant dans tous les cas.

Exercice 5. On dispose initialement de N œufs, et d'un immeuble haut de H étages - l'étage 0 est le rez-de-chaussée, le dernier étage étant donc le $H-1$ème.

On suppose, pour se faciliter la tâche, que les œufs sont identiques et se comportent de la façon suivante :

— Un œuf qui se casse en chutant ne peux pas être gardé, mais on peut réutiliser sans risque un œuf qui y a résisté.
— Un œuf qui ne se brise pas en chutant de l'étage k ne se serait pas non plus brisé en chutant de l'étage $k-1$; réciproquement, s'il se casse en chutant de l'étage k, il n'aurait pas non plus résisté à une chute plus haute.
— Avant l'expérience, on ne sait rien : il est possible que les œufs se brisent même en chutant du rez-de-chaussée, ou à l'inverse qu'ils ne se cassent pas même en tombant du sommet de l'immeuble

Le but est de déterminer *exactement* l'étage à partir duquel la chute devient trop haute pour nos œufs, en minimisant le nombre de tentatives effectuées $T(N,H)$ — en particulier, on ne se préoccupe pas de savoir combien d'œufs vont être cassés, pourvu que l'on détermine l'étage en question en un minimum de lancers. Autrement dit, $T(N,H)$ est le nombre de nombre minimal de jeters d'œufs qui permette dans tous les cas de détérminer l'étage limite.

1. Déterminer un algorithme optimal quand on ne dispose que d'un œuf ($N=1$). Combien de tentatives doit-on faire dans le pire des cas ?
2. Considérez les deux situations suivantes :
   — Il ne vous reste plus que n œufs mais après plusieurs lâchers, vous avez déterminé qu'ils résistent à la chute de l'étage 0 mais pas à celle de l'étage k
   — Il vous reste n œufs, et vous savez qu'ils résistent à la chute de l'étage k' mais pas celle de l'étage $k'+k$

   Que dire du nombre optimal de tentatives restantes dans ces deux scénarii ?
3. En déduire une relation de récurrence sur T, puis un algorithme calculant $T(N,H)$. Quelle est sa complexité ?
4. (Bonus) Donner un algorithme en $\mathcal{O}(NH\log(H))$
5. (Bonus difficile) Donner un algorithme en $\mathcal{O}(N\log H)$. Indication : Considérer le problème sous l'angle suivant : si on note m l'étage limite cherché, poser $F(J,N)$ le nombre maximal de m pouvant être distingués en J jeters et avec N œufs.