Analyse des exercices sur le concours interne Orange (Sept 2019)

Exercice 1: Empreinte de fichier

L'exercice consiste à se donner une chaine de caractère S, puis on note T_1, T_2, T_3 trois transformations qui renvoient des chaines de caractères et on note $T^*(s) = T^{n_{0,T}(s)}(s)$ où $n_{0,T} = \min\{p \in \mathbb{N} \mid T^p(s) = T^{p+1}(s)\}$, le premier indice qui rend la transformation idempotente, autrement dit T^* est l'itération infinie de la transformation T sur la chaine s.

La solution de l'exercice consiste à calculer $T_3^*(T_2^*(T_1^*(s)))$ pour toute chaine de caractère d'entrée, une fois qu'on repère cette réduction du problème, on peut écrire une fonction calcule T^* pour toute transformation (boucles while et égalité de chaines), puis écrire T_1, T_2, T_3 à base de méthodes / fonctions `replace` et les combiner pour résoudre l'exercice.

C'est un exercice qui requiert:

- une manipulation des chaines de caractères assez primitive: fonction de remplacement dans une chaine ;
- traduire l'énoncé en une opération de « calcul de transformation limite T^* » dans un bon ordre (1 puis 2 puis 3 par exemple) ;

Exercice 2: Casino Royal

L'exercice consiste à effectuer un balayage sur une famille d'entiers relatifs $(a_i)_{i\in[[1,n]]} \in \mathbb{Z}^n$ pour calculer $\min_{(i,j)\in[[1,n]]^2} \sum_{k=i}^{j} a_k$.

Ceci se résout en considérant une quantité cumulative en partant du début, tant qu'on est négatif (on fait perdre de l'argent), on ajoute, dès lors qu'on fait gagner de l'argent, c'est comme si on recommençait à zéro et « on oublie tout ce qu'on a cumulé précédemment ». On note à cumul le meilleur minimum qu'on a vu.

Par parcours exhaustif, à la fin, c'est nécessairement la solution.

C'est un exercice qui requiert de reconnaitre l'astuce: « pour trouver le minimum, il suffit de calculer une somme glissante et dès lorsqu'on arrive à une situation

positive, on se remet à zéro et on continue le parcours. » et de savoir effectuer des balayages simples.

Ceci fournit un algorithme de résolution en $O(n)$.

Exercice 3: Crapette

L'exercice est une variante du Tour de Hanoi: problème du Tour de Hanoi généralisé ; ici, nous avons M piles vides et N cartes sur la pile de départ.

Cet exercice pour être résolu le plus efficacement requiert de réfléchir sur papier ou de remarquer immédiatement la nature récursive du problème (quitte à voir que c'est une variante du Tour de Hanoi), en effet, si on modélise par $T(N, depart, arrivee, piles, vides)$ la chaine de caractère qui donne les mouvements pour déplacer toutes les cartes de la pile `depart` vers la pile `arrivee`, on peut dire:

(1) S'il y a plus d'une carte à déplacer, on a besoin d'une pile intermédiaire, donc on observe si vides contient au moins un élément, si non: il est impossible qu'on puisse s'en sortir, si oui: on prend une pile intermédiaire u.

Dans ce cas, on regarde si la pile d'arrivée est vide, on l'ajoute à l'ensemble des piles vides si oui.

Puis, on calcule $a = T(N-1, depart, u, piles, vides)$, on déplace donc $N-1$ cartes de `depart` vers `u`, ceci est l'étape $(*)$.

Remarquons: le contenu de `vides` ne change pas, car les $N-1$ cartes déplacés sont entièrement sur `u`, on ne doit pas donc perdre des piles vides après tous les mouvements de a exécutés, donc il doit être sauvegardé.

La dernière carte qui reste sur `depart` est la plus grande, celle-ci, on la met directement sur l'arrivée, ceci est l'étape $(***)$.

Donc on en tire un mouvement d'une carte c.

Remarquons : Nous n'avons plus d'utilité pour la pile de départ désormais, si elle est bien vide, utilisons la comme pile vide.

Enfin, on veut remettre les $N-1$ cartes sur la dernière carte qui se trouve sur `arrivee`, donc on calcule $b = T(N-1, u, arrivee, piles, vides)$, ceci est l'étape $(**)$.

Finalement, on retourne la juxtaposition des mouvements a puis c puis b.

(2) Pour que le raisonnement soit complet, il faut désormais savoir ce qu'on fait lorsqu'on calcule $T(1, depart, arrivee, piles, vides)$, il faut que la carte du dessus de `depart` soit plus petite que celle qu'il y a sur `arrivee`, or: ou bien cet appel est effectué à partir du solveur, donc `arrivee` est vide, donc tout va bien. Ou bien cet appel est effectué à la première étape $(*)$ de

$T(2, a, b, piles, vides)$, dans ce cas `arrivee` est une pile vide choisie depuis `vides`, sinon c'est la seconde étape $(**)$ de $T(2, a, b, piles, vides)$, dans ce cas: `arrivee` contient la plus grande carte conformément à $(***)$, donc tout va bien encore.

Dans tous les cas, le cas de départ génère un mouvement valide.

En définitive, par récurrence sur la proposition HR_n : « $T(n, depart, arrivee, piles, vides)$ retourne la bonne série de mouvements » a été démontré.

On voit effectivement que le raisonnement sur papier permet de grandement faciliter les étapes et de ne pas s'emmêler les pinceaux, c'est un écueil classique inévitable pour les problèmes type Hanoi.

Exercice 4: Robots perdus

Cet exercice consistait à trouver une séquence de mouvements qui permettaient de faire rencontrer deux robots sur un graphe.

Dans la même idée, un petit peu de raisonnement sur papier afin de réduire ce problème suffit à le résoudre très vite, le contexte apportait deux indices: un graphe et des mouvements.

Donc, on observe qu'on dispose d'un graphe $G = (S, A)$ et une fonction d'étiquetage un peu étrange $e : A \to \mathcal{P}(\Sigma)$, en effet, pour une même arete, on pouvait avoir plusieurs lettres (explicité par l'avertissement dans l'énoncé). Si on ne reconnait pas la structure d'automate non déterministe à ce stade, on peut essayer de raisonner plus encore.

Reformulons le problème à présent: on veut trouver une suite de couple sommets-mouvement partant respectivement des deux positions initiales des robots et qui tombent sur la même position. Cela parait un peu fastidieux et très peu classique. À ce stade, on se laisse porter par l'insatisfaction manifeste en face de la reformulation du problème et on part à la recherche d'une meilleure structure de données en informatique pour résoudre le problème: l'automate.

Ces mouvements induisaient donc un alphabet Σ et le graphe, une application de transition Δ et des états (qui sont dans l'exercice des positions) qui induisent tous les deux naturellement un automate non déterministe fini $\mathcal{A} = (S, \Delta, \ldots)$, on se fiche de l'état initial et des états finaux ici.

Reformulons à nouveau le problème à présent: on cherche un **mot** m sur Σ tel qu'en notant $(s, s') \in S^2$ les deux positions respectives de nos robots perdus: $\Delta^*(s, m) = \Delta^*(s', m)$ où Δ^* est la fonction de transition étendus aux mots (juste l'itération de Δ sur chaque lettre).

À ce stade, on veut un algorithme efficace pour calculer un tel m, une façon assez directe et de réécrire encore mieux le problème, en considérant l'automate produit $\mathcal{B} = \mathcal{A} \times \mathcal{A}$ de fonction de transition δ.

Cette fois-ci: on cherche un mot m sur Σ tel que $\delta^*((s, s'), m) = \delta^*((c, c), m)$ pour au moins un $c \in S$. Il est bien plus facile de chercher un couple (m, c) tel que cette égalité se produit plutôt qu'un m tel que l'égalité précédente se produit, bien qu'il soit tout à fait possible d'écrire un algorithme à partir de l'égalité précédente, mais qui requiert beaucoup plus de technicité.

À présent, nous sommes invités à écrire un algorithme naturel:

(1) Calculer l'automate produit (construction ensembliste avec des produits cartésiens qui se traduisent par des boucles `for` imbriquées).
(2) Calculer m par un parcours en largeur du graphe induit par l'automate produit, on explore tout le graphe en prenant toutes les transitions jusqu'au jour où on tombe sur un état double de la forme (c, c) qui est une solution du problème.

De plus, par la nature du parcours en largeur, ce mot serait le plus court parmi toutes les solutions, ce n'était pas requis par l'énoncé en revanche.

La vraie difficulté de ce problème réside dans le choix de la structure qui facilitera le mieux la traduction du problème.