L3-INFO Algorithmique

Examen – 13 décembre 2021 – durée 2 h 30

Les documents et les téléphones (incluant smartphone, tablettes,... tout ce qui contient une interface réseau) interdits.
Seuls les dictionnaires pour les personnes de langue étrangère sont autorisés.
Le barème (indicatif) est sur 27 pour tenir compte de la longueur du sujet.

Comme d'habitude, vous êtes vivement encouragés à illustrer vos recherches et vos réponses par des schémas.

1 Fausse pièce (diviser pour régner) *(8 points)*

On dispose d'un tas de pièces d'or, dont une (exactement) est fausse. Elle est impossible à reconnaître à l'œil nu, cependant on sait qu'elle est un peu plus lourde que les autres pièces.

Pour détecter la fausse pièce, on dispose d'une balance à plateaux comme celle représentée ci-contre, qui permet de comparer les poids de deux groupes de pièces. **Dans tout cet exercice, les coûts des algorithmes seront exprimés en nombre de pesées effectuées à l'aide de la balance.**

Question 1 : Algorithme naïf *(2 points)*

Écrire un algorithme simple permettant de déterminer quelle est la fausse pièce.
Déterminer le coût de cet algorithme.

Question 2 : Dichotomie *(2 points)*

On propose maintenant de résoudre ce problème par l'approche « diviser pour régner ».
On suppose dans un premier temps que le nombre de pièces dont on dispose est une puissance de 2.
Écrire un algorithme efficace qui détermine quelle est la fausse pièce en séparant le tas de pièces en deux sous-tas de même taille.

Question 3 : Dichotomie dans le cas général *(1 point)*

Comment faut-il procéder si le nombre de pièces est quelconque ?
Indication : il suffit de s'intéresser au cas où le nombre de pièces est impair.

Question 4 : Complexité *(1 point)*

Quel est le coût de cet algorithme efficace ?

Question 5 : Trichotomie *(2 points)*

Proposer une méthode encore plus efficace, qui consiste à séparer le tas de pièces en **trois** sous-tas de même taille.
Pour guider la recherche, on pourra commencer par supposer que le nombre de pièces est une puissance de 3, puis généraliser en s'intéressant à un nombre de pièces non divisible par 3.

2 Gestion de frigo *(11 points)*

Un étudiant a rempli son frigo mais il n'a pas été très attentif aux dates de péremption lorsqu'il a fait ses courses, et il n'aura pas le temps de tout manger. Il faut donc qu'il fasse des choix ; cependant vous pouvez l'aider à optimiser ses menus.

On suppose que le frigo contient des plats numérotés de 1 à n :

— Les jours sont numérotés par des entiers, à partir de 1 pour aujourd'hui.

— Chaque plat i a une date de péremption p_i (un entier $\geqslant 1$) ; au-delà de ce jour, le plat n'est plus mangeable.

— Chaque plat i a également une valeur nutritionnelle v_i ; l'objectif est de maximiser la valeur nutritionnelle totale de tous les plats mangés.

Notre étudiant mange au maximum un plat par jour. Si certains jours il n'y a rien d'intéressant à manger dans le frigo ce n'est pas grave : l'étudiant ne meurt pas de faim, il va au Crous mais les plats qui y sont servis ont une valeur nutritionnelle nulle.

Par exemple, avec le frigo suivant :

Plat i	1	2	3	4
Date de péremption p_i	1	4	4	1
Valeur v_i	10	20	50	30

L'étudiant pourrait :

— manger le plat 4 le jour 1 ;

— manger le plat 2 le jour 2

— aller au Crous le jour 3 ;

— manger le plat 3 le jour 4

le tout pour une valeur totale de $v_4 + v_2 + v_3 = 30 + 20 + 50 = 100$.

Cette solution est clairement optimale :

— Parmi les plats 1 et 4, un seul peut être mangé (l'autre périmera après le jour 1), on choisit donc sans hésiter le plat 4 qui a la meilleure valeur (30).

— Ensuite, il reste les plats 2 et 3. Ceux-ci peuvent être mangés dans n'importe quel ordre, deux jours au choix parmi les jours 2, 3 ou 4 ; la valeur totale sera la même à la fin.

On cherche à construire un algorithme glouton qui, étant données les dates de péremption et les valeurs des plats, renvoie la valeur totale maximale que peut espérer manger l'étudiant.

2.1 Deux stratégies incorrectes

Question 6 : Première stratégie *(2 points)*

> On propose la stratégie gloutonne suivante : *Examiner les plats par **date de péremption croissante**. En cas d'égalité, on départage par **valeur décroissante**. On mange les plats le plus tôt possible, à condition qu'il reste un jour compatible avec leur date de péremption, sinon on les jette.*
> Avec le frigo donné en exemple plus haut, on examine les plats dans l'ordre 4, 1, 3, 2. On jettera donc le plat 1, et on mangera les autres les jours 1, 2 et 3 respectivement.
> Donner un exemple d'instance sur laquelle cette stratégie échoue à trouver les meilleurs menus.

Question 7 : Deuxième stratégie *(2 points)*

On propose une seconde stratégie gloutonne : *Examiner les plats par* ***date de péremption décroissante****. En cas d'égalité, on départage par* ***valeur décroissante****. On mange les plats le plus tard possible, toujours en jetant les plats pour lesquels il ne reste aucun jour convenable.*

Avec le frigo donné en exemple plus haut, on examine les plats dans l'ordre 3, 2, 4, 1. On jettera donc (encore) le plat 1 et on mangera les autres les jours 4, 3 et 1 respectivement.

Donner un exemple d'instance sur laquelle cette stratégie échoue à trouver les meilleurs menus.

2.2 Une troisième stratégie

Cette fois on examine les plats par **valeur décroissante** (les ex-aequo n'ont pas d'importance), et on affecte chaque plat à la date la plus éloignée possible pour lui (sauf s'il n'y a plus de jour disponible pour manger ce plat, dans ce cas on le jette).

Par exemple, avec le contenu du frigo donné plus haut, on classe les plats dans l'ordre :

Plat i	3	4	2	1
Date de péremption p_i	4	1	4	1
Valeur v_i	50	30	20	10

puis on les affecte de la façon suivante :

— on mange le plat 3 le plus tard possible, c'est-à-dire le jour 4

— on mange le plat 4 le plus tard possible, c'est-à-dire le jour 1

— on mange le plat 2 le plus tard possible, c'est-à-dire le jour 3 (le 4è jour est déjà pris)

— on jette le plat 1 car il n'y a plus de date disponible pour lui

Question 8 : Illustration *(2 points)*

Illustrer le fonctionnement de cet algorithme sur un exemple plus élaboré. On prendra soin de choisir l'instance afin de bien observer les différents cas qui peuvent se présenter.

Question 9 : Algorithme *(3 points)*

Rédiger précisément en pseudo-code l'idée d'algorithme décrite ci-dessus.

Les dates de péremption et les valeurs des plats sont données dans deux tableaux P et V indexés de 1 à n. On suppose que ces deux tableaux sont **déjà triés** de sorte que V soit décroissant.

On pourra utiliser d'autres tableaux pour mémoriser toute information utile durant l'exécution de l'algorithme.

Question 10 : Complexité *(2 points)*

Déterminer la complexité au pire de l'algorithme écrit à la question 9.

3 Plus petit élément absent *(8 points)*

On donne un tableau T indexé de 0 à $N-1$, dont les cellules contiennent des entiers positifs (ou nuls) **tous distincts**.

L'objectif de ce problème est de déterminer le **plus petit** entier naturel M qui n'est **pas présent** dans T.

Par exemple, pour le tableau $T = [5, 1, 0, 8, 2]$ l'entier recherché vaut 3.
Pour le tableau $T = [4, 1, 3, 5]$ l'entier recherché est 0.

Question 11 : Une propriété utile *(1 point)*

Démontrer que $M \leqslant N$ (un raisonnement par l'absurde est conseillé).

Pour résoudre ce problème, on utilise l'algorithme suivant :

```
MINIMUM_ABSENT(T)
  Données : Un tableau T indexé de 0 à N − 1, dont les cellules contiennent des entiers
            positifs (ou nuls) tous distincts
  Résultat : Le plus petit entier naturel M qui n'est pas présent dans T
1 Soit P un tableau de booléens indexé de 0 à N − 1.
2 pour i := 0 à N − 1 faire
3     P[i] := Faux
4 pour i := 0 à N − 1 faire
5     si T[i] < N alors
6         P[T[i]] := Vrai
7 i := 0
8 tant que i < N et P[i] == Vrai faire
9     i := i + 1
10 renvoyer i
```

Question 12 : Invariant de la première boucle *(3 points)*

On donne l'invariant suivant pour la boucle lignes 4-6 :
Pour tout indice $j < N$,
$P[j] == Vrai$ *si et seulement si* j *est présent dans* T *entre les indices 0 et* $i-1$ *(inclus).*

Démontrer qu'il est vérifié à l'entrée de la boucle, et qu'il s'agit bien d'un invariant pour cette boucle.

Question 13 : Invariant de la seconde boucle *(4 points)*

Donner un invariant pour la boucle lignes 8-9.
Démontrer qu'il s'agit bien d'un invariant pour cette boucle, et l'utiliser pour prouver que l'algorithme est correct.