Lyce Faidherbe (Lille)

DS 1 IPT 2019-2020

(Durée : 1 heure)

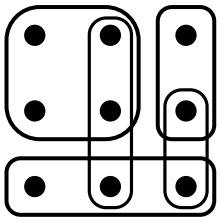
L'utilisation des calculatrices **n'est pas autorisée** pour cette épreuve.
Le langage de programmation choisi par le candidat doit être le python
On attachera une grande importance à la concision, à la clarté, et à la précision de la rédaction.

⋆⋆⋆

Couverture optimale

Ce sujet a pour motivation principale le problème d'allocation de ressources, abordé explicitement dans la Partie III. L'approche choisie consiste ici à considérer ce problème comme une instance de celui de la couverture optimale d'ensemble.

Le problème s'énonce alors comme suit. Soit U un ensemble fini. Étant donnée une famille F constituée d'ensembles inclus dans U telle que F couvre U (c'est-à-dire que chaque élément de U appartient à au moins un ensemble de F), de combien d'ensembles de F a-t-on besoin, au minimum, pour couvrir U? Dans l'exemple ci-dessous, l'ensemble U est constitue de 9 points. La famille F contient 5 ensembles et recouvre U. Aucune sous-famille de F constituée d'au plus 2 ensembles n'est couvrante. En revanche, la sous-famille constituée des 3 ensembles en traits épais sur le dessin est couvrante. Une telle famille est dite *optimale*.



L'utilisation de fonctions puissance et logarithme prédéfinies dans le langage est interdit.

La complexité, ou le temps d'exécution, d'un programme P (fonction ou procédure) est le nombre d'opérations élémentaires (addition, soustraction, multiplication, division, affectation, etc...) nécessaires à l'exécution de P. Lorsque cette complexité dépend d'un paramètre n, on dira que P a une complexité en $O(f(n))$, s'il existe $K > 0$ tel que la complexité de P est au plus $K\,f(n)$, pour tout n. Lorsqu'il est demandé de garantir une certaine complexité, le candidat devra justifier cette dernière si elle ne se déduit pas directement de la lecture du code.

Partie I. Autour des ensembles

Dans cette partie, on représente les ensembles finis d'entiers positifs ou nuls par des tableaux de booléens. L'ensemble S correspondant au tableau t contient exactement les entiers i tels que $t[i]$ est égal à `True` ; S ne contient pas les entiers i tels que est égal à `False`, ou tel que $t[i]$ n'est pas défini. Ainsi, le tableau [`False`, `True`, `True`, `False`, `True`] représente l'ensemble $\{1, 2, 4\}$. On remarque qu'un même ensemble S peut être représenté de multiples façons, en jouant sur la longueur du tableau. Par exemple [`False`, `True`, `True`, `False`, `True`, `False`] et [`False`, `True`, `True`, `False`, `True`, `False`, `False`, `False`, `False`] sont d'autres représentations de l'ensemble $\{1, 2, 4\}$.

Question 0 Écrire une fonction `parite`(k), qui prend en argument un entier k renvoyant 0 si l'entier k est pair et 1 sinon .

Question 1 Écrire une fonction `cardinal`(t), qui prend en argument un tableau t de booléens représentant l'ensemble S, et qui renvoie le cardinal de l'ensemble S .

Question 2 Écrire une fonction `appartient`(i, t), qui prend en argument un entier i et un tableau de booléens t représentant l'ensemble S, et qui renvoie `True` si i appartient à S et `False` sinon.

Question 3 Écrire une fonction `diff`(t_1, t_2), qui prend en argument deux tableaux de booléens t_1 et t_2 représentant les ensembles S_1 et S_2, et qui renvoie un tableau représentant la différence ensembliste $S_1 \setminus S_2$. Ici $S_1 \setminus S_2$ désigne l'ensemble constitué des éléments de S_1 qui ne sont pas dans S_2.

Question 4 Écrire une fonction `union`(t_1, t_2), qui prend en argument deux tableaux de booléens t_1 et t_2 représentant les ensembles S_1 et S_2, et qui renvoie un tableau représentant l'union de S_1 et S_2.

Partie II. Représentation par entiers

Afin de pouvoir manipuler simplement des familles d'ensembles, on souhaite maintenant représenter les ensembles par des entiers. Pour ce faire, on associe à un ensemble S (d'entiers positifs ou nuls) l'entier $s = \sum_{i \in S} 2^i$. Par exemple, l'ensemble $\{1, 2, 4\}$ correspond à l'entier $2^1 + 2^2 + 2^4$, soit 22. Contrairement à la représentation par tableau de booléens, cette représentation est unique. On prendra garde à ne pas confondre les entiers représentant des ensembles avec les entiers contenus dans ces ensembles.

Question 5 Si $U = \{0, 1, 2, ..., n - 1\}$, quel est le plus grand entier représentant un sous-ensemble de U? Quel est le plus petit?

Question 6 Écrire une fonction `set2int`(t) qui prend en argument un tableau t de booléens représentant un ensemble S, qui renvoie l'entier s représentant S, et dont le temps d'exécution est linéaire en la taille de t.

On prendra soin de s'assurer et de justifier brièvement que le temps d'exécution de la fonction est linéaire en la taille de t. On rappelle que ***l'utilisation de fonctions puissance et logarithme prédéfinies dans le langage est interdit****.*

Question 7 Écrire une fonction `int2set`(s) qui prend en argument un entier s représentant un ensemble S, et qui renvoie un tableau représentant S. En estimer le temps d'exécution. On prendra soin d'expliquer de quelle manière on choisit un vecteur particulier parmi les multiples représentations possibles.

Grâce aux deux fonctions ci-dessus, il est possible d'étendre les fonctions de la partie précédente aux entiers représentant des ensembles. Dans le reste du sujet, on supposera que toutes les fonctions de la partie précédente ont été réécrite afin de pouvoir opérer directement sur les entiers (arguments et résultat).

Partie III. Familles, sous-familles, et couvertures

On aborde maintenant le problème de couverture optimale, défini en préambule du sujet.

Question 8 Une école souhaite proposer des activités sportives à ses élèves. Il y a plusieurs activités possibles, et chaque élève est invité à dire lesquelles lui conviennent (en en choisissant au moins une). L'école souhaite minimiser le nombre d'activités à organiser, tout en garantissant que chaque élève puisse suivre une activité qui lui convient. Montrer comment ce problème peut être vu comme un problème de couverture optimale.

On peut représenter une famille finie $F = (F_0, F_1, ...)$ (sous-entendue ordonnée) constituée d'ensembles finis d'entiers (positifs ou nuls) par un tableau f, dont chaque case f est l'entier qui représente l'ensemble F_ℓ au sens de la partie II. Par exemple, le tableau $[17, 3, 8, 22]$ représente la famille $(\{0, 4\}, \{0, 1\}, \{3\}, \{1, 2, 4\})$, puisque $2^0+2^4 = 17$, $2^0+2^1 = 3$, $2^3 = 8$ et $2^1+2^2+2^4 = 22$.

On fixe désormais $U = \{0, 1, 2, ..., n-1\}$ et une famille F d'ensembles inclus dans U telle que F couvre l'ensemble U. Dans tout le reste du sujet, on suppose que F est donnée par un tableau f d'entiers. On suppose aussi que le tableau f et l'entier n sont accessibles dans toutes les fonctions, sans avoir à les définir ni à les passer en argument.

Une sous-famille G de F est constituée de certains des ensembles de la famille F, et est donc de la forme $(F_{\ell_0}, F_{\ell_1}, ...)$, où $\ell_0, \ell_1, ...$ est une sous-suite finie strictement croissante des indices de f. Par conséquent G est définie par l'ensemble des indices $\{\ell_0, \ell_1, ...\}$. Le cardinal de G est donc le cardinal de l'ensemble $\{\ell_0, \ell_1, ...\}$. Au sens de la partie II, cet ensemble d'indices peut une nouvelle fois être représenté par un entier. Par exemple, si F est la famille $(\{0, 4\}, \{0, 1\}, \{3\}, \{1, 2, 4\})$, la sous-famille $G = (\{0, 4\}, \{3\})$, qui est constituée des ensembles F_0 et F_2, est de cardinal 2 et est représentée par l'entier 5, puisque $2^0 + 2^2 = 5$.

Noter que désormais trois objets distincts sont représentés par des entiers.

- Les éléments de $U = \{0, 1, 2, ..., n-1\}$: dans ce cas la lettre i sera de préférence utilisée, et il n'y a pas de codage particulier, i représente l'entier i;
- Les sous-ensembles de U : dans ce cas la lettre s sera de préférence utilisée, et s représente l'ensemble $S = \texttt{int2set}(s)$;
- Les sous-familles de F : dans ce cas la lettre g sera de préférence utilisée, et g représente la sous-famille définie par les ensembles F_ℓ pour $\ell \in \texttt{int2set}(g)$. De plus, chaque ensemble F_ℓ est lui-même représenté par l'entier $f[\ell]$.

Une première façon de trouver une sous-famille couvrante de petit cardinal consiste à utiliser un algorithme dit "glouton". L'idée est de construire une sous-famille étape par étape, en gardant en mémoire l'ensemble des éléments déjà couverts, et en ajoutant à chaque fois l'ensemble qui permet de couvrir le plus de nouveaux éléments.

Question 9 Proposer un exemple de famille où l'algorithme glouton ne renvoie pas une sous-famille couvrante optimale.

Question 10 Écrire une fonction `reste`(s_1, s_2), qui prend en argument deux entiers s_1 et s_2 représentant les sous-ensembles S_1 et S_2 de $U = \{0, 1, ..., n-1\}$, et qui renvoie le nombre d'éléments de S_1 qui ne sont pas dans S_2.

Question 11 Écrire une fonction `glouton`(), qui renvoie un entier g représentant une sous-famille couvrante G de F grâce à l'algorithme glouton décrit ci-dessus. Estimer son temps d'exécution.

Question 12 Écrire une fonction `couverture`(g), qui prend en argument un entier g représentant une sous-famille G de F et qui renvoie `True` si G est couvrante et `False` sinon.

Question 13 Écrire une fonction `optimale`() qui parcourt toutes les sous-familles possibles afin de renvoyer une sous-famille couvrante de F optimale. Estimer son temps l'exécution.

* *
*