Épreuve d'informatique de l'X - 2006 - PSI

L'utilisation des calculatrices **n'est pas autorisée** pour cette épreuve. Le langage de programmation choisi par le candidat doit être spécifié en tête de la copie.

⋆ ⋆ ⋆

Alliance d'entreprises

On attachera une grande importance à la concision, à la clarté et à la précision de la rédaction.

Le temps d'exécution $T(f)$ d'une fonction f est le nombre d'opérations élémentaires (addition, soustraction, multiplication, division, affectation, etc.) nécessaire au calcul de f. Lorsque ce temps d'exécution dépend d'un paramètre n, il sera noté $T_n(f)$. On dit que la fonction f s'exécute :

— en temps linéaire en n, s'il existe $K > 0$ tel que pour tout n, $T_n(f) \leqslant Kn$;

— en temps quadratique en n, s'il existe $K > 0$ tel que pour tout n, $T_n(f) \leqslant Kn^2$;

— plus généralement en temps $g(n)$, s'il existe $K > 0$ tel que pour tout n, $T_n(f) \leqslant Kg(n)$.

Dans tout le problème, un tableau c d'entiers strictement positifs contient les chiffres d'affaires $c[0], c[1], \ldots, c[n-1]$ en euros de n petites et moyennes entreprises ($n > 0$). La variable n sera supposée être une constante globale.

Ces entreprises décident de s'allier pour devenir leader de leur secteur dominé par l'entreprise X qui a un chiffre d'affaires de Obj euros. Il s'agit donc de trouver un sous-ensemble d'entreprises I tel que l'on ait $\sum_{i\in I} c[i] \geqslant Obj$. Un tel ensemble d'entreprises est appelée une *alliance réussie*. Malheureusement, les alliances apportent parfois des désagréments. Pour diminuer ce risque, on cherche à réaliser une *alliance réussie stable* (une alliance *stable* en abrégé) vérifiant $\sum_{i\in I'} c[i] < Obj$ pour tout I' strictement inclus dans I.

On supposera dans tout l'énoncé qu'une telle alliance existe c'est-à-dire que $\sum_{i=0}^{n-1} c[i] \geqslant Obj$ et on dira que le chiffre d'affaires Obj est l'*objectif* des alliances considérées.

Partie I. Alliances

Une alliance est représentée par un tableau a de booléens tel que $a[i]$ vaut `vrai` si l'entreprise i est dans l'alliance et `faux` sinon ($0 \leqslant i < n$). On pourra poser `vrai` $= 1$ et `faux` $= 0$.

Question 1 Écrire une fonction `somme`(c, a) qui retourne, en temps linéaire par rapport à n, la somme des chiffres d'affaires des entreprises de l'alliance représentée par le tableau a.

Question 2 Écrire une fonction `estReussie`(c, a, Obj) qui retourne, en temps linéaire par rapport à n, la valeur `vrai` si l'alliance représentée par le tableau a est une alliance *réussie* pour l'objectif Obj. La valeur retournée est `faux` sinon.

Question 3 Écrire une fonction `estStable`(c, a, Obj) qui retourne, en temps linéaire par rapport à n, la valeur `vrai` si l'alliance représentée par le tableau a est une alliance *stable* pour l'objectif Obj. La valeur retournée est `faux` sinon.

Question 4 On suppose les entreprises triées selon leur chiffre d'affaires $c[0] \leqslant c[1] \leqslant \ldots \leqslant c[n-1]$. Écrire une fonction `allianceMin`(c, Obj) qui imprime, en temps linéaire par rapport à n, les numéros des entreprises d'une plus petite (en nombre d'entreprises) alliance *stable* pour l'objectif Obj.

Partie II. Calcul des alliances stables.

Dans cette partie, on imprime toutes les alliances *stables* en énumérant toutes les alliances et en testant à chaque fois s'il s'agit d'une alliance *stable* pour l'objectif Obj. Au tableau a, on fait correspondre de manière unique le nombre $bin(a)$ défini par :

$$bin(a) = a[0] * 2^0 + a[1] * 2^1 + \ldots + a[n-1] * 2^{n-1}$$

de représentation binaire $(a[0], a[1], \ldots, a[n-1])$. Ainsi, pour $n = 5$ et l'alliance des entreprises $\{1, 3, 4\}$, le nombre associé est 26 de représentation binaire $(0, 1, 0, 1, 1)$. il suffit donc d'énumérer les nombres binaires qu'on peut écrire avec n bits pour énumérer toutes les alliances possibles.

Question 5 Écrire une fonction `suivanteDe(`a`)` qui modifie, en temps linéaire par rapport à n, le tableau a pour donner l'alliance suivante a' telle que $bin(a') = bin(a) + 1$. Cette fonction retourne la valeur `vrai` si cette opération est possible, et `faux` sinon (dans ce cas, il n'y a pas d'alliance suivante).

Question 6 Écrire une fonction `imprimer(`a`)` qui imprime l'alliance correspondant au tableau a. Ainsi pour $n = 5$, et $a[0] = 0$, $a[1] = 1$, $a[2] = 0$, $a[3] = 1$, $a[4] = 1$, l'impression donnera 134.

Question 7 Écrire une fonction `imprimerStables`(c, Obj) qui affiche toutes les alliances stables pour l'objectif Obj. Donner un ordre de grandeur du nombre d'opérations effectuées par rapport à n.

Partie III. Optimisation

On définit l'ordre binaire $\prec$ par $a \prec a'$ si et seulement si $bin(a) < bin(a')$. Dans la partie II, on a imprimé les alliances *stables* dans l'ordre binaire $\prec$ croissant. Dans cette partie, on optimise l'impression de ces alliances en supposant que les entreprises sont classées par chiffres d'affaires croissants. On suppose donc $c[0] \leqslant c[1] \leqslant \ldots c[n-1]$.
Pour réaliser cette impression rapide, on construit le tableau t vérifiant $t[i] = c[0] + c[1] + \ldots c[i]$ pour $0 \leqslant i < n$.

Question 8 Écrire une fonction `cumuls`(c, t) qui calcule, en temps linéaire par rapport à n, le tableau t.

Pour trouver a, première alliance stable dans l'ordre binaire, on cherche l'indice k minimum tel que $t[k] > Obj$. L'entreprise k est dans l'alliance a, et on doit compléter cette alliance avec des entreprises de chiffres d'affaires plus petits pour dépasser la somme restante $Obj' = Obj - c[k]$. Pour compléter cette alliance partielle $\{k\}$, on cherche le plus petit k' tel que $t[k'] \geqslant Obj'$. On sait qu'il existe puisqu'on a $t[k] \geqslant Obj$. Donc k' est aussi dans l'alliance a et on recommence à nouveau pour dépasser la somme restante $Obj'' = Obj' - c[k']$ en complétant l'alliance partielle $\{k, k'\}, \ldots$ jusqu'au moment où la somme restante devient négative ou nulle.

Question 9 Écrire une fonction `completer`(t, c, a, Obj, k) qui, pour la somme restante Obj, retourne k' minimum appartenant à a' première alliance *stable* suivante de a dans l'ordre binaire (k est l'entreprise minimale appartenant à a). Cette fonction modifie ses paramètres a et Obj en ajoutant quelques k' comme indiqué dans le procédé de complétion précédemment décrit. (On se place dans le cas où $k < n$ et $t[k] \geqslant Obj$).

Soit i le plus petit indice d'une entreprise appartenant à l'alliance a. Soit alors ℓ le plus grand entier tel que les entreprises i, $i+1, \ldots, i+\ell$ sont toutes dans a. Pour créer l'alliance *stable* suivante, on définit l'alliance a' qui part de a à laquelle on a enlevé les entreprises i, $i+1, \ldots, i+\ell$. Soient $k = i + \ell + 1$ et Obj' l'objectif Obj diminué des chiffres d'affaires des entreprises de a'. Si $k = n$, alors il n'y a pas de *stable* suivante, sinon on a nécessairement par construction $t[k] > Obj'$. On ajoute alors l'entreprise k à l'alliance et on complète celle-ci comme pour la question précédente. On obtient alors la prochaine alliance *stable*.

Question 10 Écrire une fonction `prelude`(t, c, a, Obj, i) qui retourne l'entier k défini ci-dessus, s'il existe. Sinon, on retourne $k = n$. Cette fonction modifie ses paramètres a et Obj en retirant de a tous les indices entre les entiers i et $k-1$. Ainsi, a sera devenue l'alliance a' et Obj sera égal au nouvel objectif Obj'.

Question 11 En déduire une fonction `imprimerAStables1`(c, Obj) qui imprime rapidement toutes les alliances *stables* pour l'objectif Obj. Donner un ordre de grandeur du temps d'exécution en fonction de n et du nombre K d'alliances imprimées.