IUT de Montpellier **TDS4 : Complexité**
R3.02 Développement efficace

Exercice 1. 3-SUM

Question 1.1.
Ecrire une méthode `boolean somme(int[]t)` qui retourne vrai ssi il existe trois entiers distincts i, j et k tels que $t[i] + t[j] + t[k] = 0$ et qui s'execute en $\mathcal{O}(n^3)$, où n est la taille de t. Faites le calcul de complexité.

Question 1.2.
Montrez tout d'abord que l'algorithme précédent n'est pas en $\mathcal{O}(n^2)$ (ni même en $\mathcal{O}(n^{3-\epsilon})$ pour tout $\epsilon > 0$). On rappelle que pour ce faire, il suffit de trouver pour tout n un tableau t (un "pire cas") tel que `somme(t)` fasse $m(n) \geq p(n)$ opérations, avec p un polynôme de degré 3. D'après le cours, cela impliquera bien que l'algorithme précédent n'est pas en $\mathcal{O}(n^{3-\epsilon})$.

Question 1.3.
Ecrire une méthode `boolean aux(int[]t, int x)` qui étant donné un tableau **trié par ordre croissant** retourne vrai ssi il existe deux i, j distincts tels que $t[i] + t[j] = x$ et qui s'exécute en $\mathcal{O}(n)$. Indication : commencez par tester $t[0] + t[t.length - 1]$:

- si cela fait x, on a trouvé
- si c'est strictement supérieur ou inférieur à x, que faire ?

Question 1.4.
Ecrire la méthode `boolean sommeV2(int []t)` qui retourne vrai ssi il existe trois entiers distincts i, j et k tels que $t[i]+t[j]+t[k] = 0$ et qui s'exécute en $\mathcal{O}(n^2)$. Indication : commencez par utiliser une méthode (que l'on suppose existante) tri(t), qui trie t par ordre croissant, et s'exécute en $\mathcal{O}(n^2)$. Faites le calcul de complexité.

Exercice 2. Etoile

On considère un groupe de n personnes numérotées de 0 à $n - 1$, et on considère que certaines personnes du groupe en admirent d'autres. Plus précisément, on connaît tous les i, j dans $[0, n-1]$ (avec possiblement $i = j$) tels que la personne i admire la personne j (noté $i \rightarrow j$). On a pas d'hypothèse particulière sur ce qui se passe entre deux personnes i et j (ils peuvent s'admirer mutuellement, que un seulement admire l'autre, ou que aucun n'admire l'autre). On modélise cette information par un tableau t de $n \times n$ casess, tel que $t[i][j] = true$ ssi $i \rightarrow j$. On dit qu'une personne i est une star ssi tout le monde admire i (y compris elle même), et i n'admire personne d'autre.

Par exemple, 1 est une star dans le groupe représenté par le tableau ci-dessous (où la première ligne dénote t[0][0], t[0][1], t[0][2]):

$$\begin{pmatrix} f & t & t \\ f & t & f \\ f & t & f \end{pmatrix}$$

Question 2.1.
Est ce que tout groupe possède une star ? Est ce qu'un groupe peut posséder deux stars ?

Question 2.2.
Ecrire la méthode `boolean verifStar(boolean [][]t, int i)` qui retourne vrai ssi i est une star, et qui s'exécute en $\mathcal{O}(n)$.

Question 2.3.
Ecrire la méthode `boolean existeStar(boolean [][]t)` qui retourne vrai ssi il existe une star dans le groupe représenté par t, et qui s'exécute en $\mathcal{O}(n^2)$. Faites le calcul de complexité.

Question 2.4.
Montrez que `existeStar` n'est pas en $\mathcal{O}(n^{2-\epsilon})$ pour tout $\epsilon > 0$. Rappel : pour ce faire, il suffit de trouver pour tout n un tableau t (de taille $n \times n$) (un "pire cas") tel que `exsiteStar(t)` fasse $m(n) \geq p(n)$ opérations, avec p un polynôme de degré 2.

Question 2.5.
Ecrire la méthode `boolean existeStarV2(boolean [][]t)` qui retourne vrai ssi il existe une star dans le groupe représenté par t, et qui s'exécute en $\mathcal{O}(n)$.