

## - TP1. Enoncés étendus. -

### - Exercice 1 - Suite de syracuse.

La *conjecture de syracuse* affirme que, partant de n'importe quel nombre entier  $n$  supérieur à 0, l'itération à partir de  $n$  des opérations suivantes :

- Si  $n$  est pair alors  $n \leftarrow n/2$ .
- Si  $n$  est impair alors  $n \leftarrow 3n + 1$ .

finit toujours par atteindre la valeur 1. Par exemple, en partant de 7, on obtient la suite suivante : 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1.

- a. Ecrire une fonction **int** syracuse(**int** n) qui affiche les valeurs de la suite partant de  $n$  et retourne le nombre d'itérations de la suite. Utiliser un **while**.
- b. Même question, mais avec une fonction récursive.
- c. Ecrire une fonction **void** verificationsyracuse(**int** n) qui teste si la conjecture de syracuse est vraie pour toutes les valeurs entre 1 et  $n$ .
- d. Même question, mais avec une fonction **int** verificationsyracusecrible(**int** n) qui économise du temps de calcul ainsi : lorsqu'on vérifie la valeur 7, la suite des calculs va aussi valider la conjecture pour les valeurs 22, 11, 34,... et il ne sera donc pas nécessaire de vérifier ultérieurement ces valeurs.
- e. Evaluer les performances des deux algorithmes précédents.

### - Exercice 2 - Sous-tableau de somme maximale.

On se donne un tableau  $T[n]$  dont les  $n$  valeurs sont des nombres entiers signés. On se propose de trouver le sous-tableau  $T[i, \dots, j]$ , avec  $i \leq j$ , dont la somme des valeurs est la plus grande possible. Par exemple, pour le tableau  $[1, -2, 3, -2, -2, 2, -1, 5, -2, 1]$ , la solution est  $[2, -1, 5]$  qui a pour somme 6.

- a. Ecrire une fonction **void** affichetableau(**int** tab[ ], **int** n) qui affiche les valeurs  $\text{tab}[0]$ ,  $\text{tab}[1]$ , ...,  $\text{tab}[n - 1]$ .
- b. Ecrire une fonction **void** tableaualeatoire(**int** tab[ ], **int** n) qui remplit aléatoirement les entrées 0 à  $n - 1$  de tab de valeurs entières comprises entre -9 et 9.
- c. Ecrire une fonction **int** sommemaxnaive(**int** tab[ ], **int** n) qui affiche le début  $i$  et la fin  $j$  d'un sous-tableau  $T[i, \dots, j]$  de somme maximale, et retourne la valeur de cette somme. Utiliser une méthode naïve consistant à calculer toutes les sommes  $T[i, \dots, j]$  pour tous les  $i \leq j$ . Votre algorithme comportera trois boucles imbriquées.
- d. Ecrire une fonction **int** sommemaxmoinsnaive(**int** tab[ ], **int** n) qui ne comporte que deux boucles imbriquées.
- e. \* Ecrire une fonction **int** sommemax(**int** tab[ ], **int** n) qui ne comporte qu'une seule boucle.
- f. Evaluer les performances de vos trois algorithmes sur des tableaux aléatoires en faisant varier  $n$ .