IF111 - Algorithmique et structures de données - TD1

Rohan Fossé

2021 - 2022

1 Maximum - Minimum

On souhaite calculer le minimum et le maximum de deux nombres entiers positifs.

1. Dans un premier temps, on n'utilisera pas d'instructions conditionnelles, mais une fonction `abs(int x)`, qui retourne la valeur absolue de l'entier `x`.
   Écrivez un algorithme affichant le maximum et le minimum de deux nombres passés en paramètres.

2. Modifiez l'algorithme précédent, en utilisant cette fois les instructions conditionnelles et l'opérateur $>$(supérieur à).

2 Rendre la monnaie

On se propose d'écrire un algorithme permettant d'obtenir la suite des billets totalisant une somme donnée (dont on suppose qu'elle est un multiple de 10). Les espèces disponibles sont des billets de 50, 20, et 10 euros.

Le principe est de donner le billet de valeur la plus grande possible inférieure ou égale à la somme à rendre et de poursuivre la même stratégie avec la somme restante jusqu'à ce que celle-ci soit nulle.

1. Écrire cet algorithme en utilisant les structures de contrôle suivantes : tant que, si, et si sinon. L'algorithme utilisera une variable nommée qui contiendra la somme restant à rendre au fur et à mesure de la remise de billets au client. On n'utilisera comme opérations arithmétiques que des additions ou des soustractions. Le résultat sera l'affichage de la suite des nombres de billets à rendre.

   Par exemple, si la somme est 180, l'affichage devra être :

   ```
   3 billets de 50 euros
   1 billet de 20 euros
   1 billet de 10 euros
   ```

2. Déterminer le nombre de soustractions effectuées par l'algorithme, on donnera une formule faisant intervenir des divisions entières par les nombres 50, 20, et 10.

3. Généraliser votre algorithme au cas où les valeurs de billets ou pièces disponibles sont en nombre quelconque et figurent dans un tableau à valeurs décroissantes `val[0]` $>$ `val[1]` $>$ ... $>$ `val[k-1]`. Ainsi dans l'exemple considéré plus haut, on aurait `k = 3` et `val[] = { 50, 20, 10 }`.

4. Montrer qu'il existe des valeurs de billets et une somme à rendre pour lesquels cet algorithme ne donne pas le nombre minimum de billets ou de pièces à rendre.

3 Manipulation de tableaux

1. Écrire un algorithme pour trouver l'élément minimum et l'élément maximum d'un tableau de n cases. Déterminer le nombre d'affectations effectuées par votre algorithme.

2. Écrire un algorithme qui permet de renverser un tableau *tab*. Le tableau *tab* sera passé en paramètre avec sa taille n. Déterminer le nombre de comparaisons effectuées par votre algorithme.

3. Un tableau est un palindrome si c'est identique en le lisant de gauche à droite (indices dans l'ordre 0,1,2,... n-1) et de droite à gauche (indices dans l'ordre n-1,...,0). Écrire un algorithme qui teste si le tableau fourni est un palindrome. Donner la complexité asymptotique de votre algorithme.

4. Écrire un algorithme qui vérifié s'il est possible de permuter les éléments d'un tableau donné en entrée de manière à obtenir un tableau palindrome. L'algorithme retournera *vrai* si c'est possible, *faux* autrement. On suppose que chaque élément du tableau est une lettre de l'alphabet français. Vous avez à disposition (pas nécessaire de l'utiliser) une fonction *CalculPositionAlphabet(c)* qui prend en entrée un caractère et retourne l'entier correspondant à sa position dans l'alphabet.

5. Écrire un algorithme pour trouver l'élément minimum et l'élément maximum d'un tableau de n cases. On suppose que tous les éléments du tableau sont différents. L'algorithme doit effectuer au plus $3\lfloor \frac{n}{2} \rfloor$ comparaisons. L'algorithme *peut* utiliser un tableau auxiliaire de taille n.

6. (bonus) Écrire un algorithme récursif qui teste si le tableau fourni est un palindrome.