Algorithmes itératifs

Exercice-1 : Recherche du maximum

1. Concevoir un algorithme de recherche du maximum dans un ensemble à n éléments
2. Quelle est la complexité de votre algorithme en nombre de comparaisons ?
3. Montrer qu'il est optimal.

Exercice-2 : Complexité de séquences itératives

Déterminer (en fonction de n à O( ) près) la complexité en nombre « d'opérations » de chaque séquence :

Séq-1 :
```
For i = 1 to N do
    Opération ;
Endfor
```

Séq-2 :
```
For i = 1 to N do
    For j = 1 to i do
        Opération ;
    Endfor
Endfor
```

Séq-3 :
```
i=1
While ( i < N) Do
    i = 2*i
    Opération ;
Endwhile
```

Séq-4:
```
For i = 1 To N Do
    J=1
    While (J < N ) Do
        J = 2 * J
        Opération;
    Endwhile
Endfor
```

Séq-5:
```
i = 1
While ( i < N ) Do
    i = 2*i
    For j = 1 to i Do
     Opération;
Endwhile
```

Séq-6:
```
i = 1
For j=1 To n do
    i = 2*i
Endfor
For j= 1 to i do
    Opération;
Endfor
```

Séq-7 :
```
For k = 1 to n do
    i = 1
    For j = 1 to k do
        i = 2*i
    Endfor
    For j = 1 to i do
        Opération
    Endfor
Endfor
```

Exercice-3 : Recherche du maximum et du minimum

Nous supposons ici que l'ensemble considéré ne contient pas deux fois la même valeur.

1. Proposer un algorithme naif de recherche du maximum et du minimum d'un ensemble de n éléments.
2. Quelle est sa complexité en nombre de comparaisons ?
3. Proposer un algorithme plus efficace.
   Indication : dans une première phase les éléments sont comparés par paire.
4. Quelle est sa complexité en nombre de comparaisons ?

Exercice-4 : Recherche du deuxième plus grand élément

Nous supposons ici que l'ensemble considéré ne contient pas deux fois la même valeur.

1. Proposer un algorithme simple de recherche du deuxième plus grand élément.
2. Quelle est sa complexité en nombre de comparaisons ?
3. Réécrire l'algorithme de recherche du maximum sous la forme d'un tournoi de tennis. Il n'est pas nécessaire de formaliser l'algorithme ici, une figure explicative suffit.
4. Dans combien de comparaisons, le deuxième plus grand élément de l'ensemble s'est rendu compte qu'il est le plus petit des deux éléments comparés ?
5. Proposer un nouvel algorithme de recherche du deuxième plus grand élément.
6. Quelle est sa complexité en nombre de comparaisons ?

Exercice-5 : Anagramme

Deux mots sont dits anagrammes s'ils sont composés des mêmes lettres mais dans un ordre différent.
Exemple : UN et NU, ELLES et SELLE, SURE et RUSE
Etant donné, un tableau de N pointeurs. Chaque pointeur pointe sur une chaîne de caractères représentant un mot.

a- Ecrire une fonction « max_anagram » qui permet de retourner la longueur du plus long anagramme. On vous conseille de commencer par écrire une fonction « anagram » qui vérifie si deux mots sont ou non anagrammes. Elle doit retourner la longueur du mot s'ils sont des anagrammes et 0 sinon.

b- En supposant que les mots ont une longueur moyenne égale à M, estimer en fonction de N et M, la complexité nécessaire pour déterminer l'anagramme le plus long en nombre de comparaisons de caractères.

c- Proposer un autre algorithme pour la fonction « anagram » pour réduire la complexité

Exercice-6 : Mots composés

Il existe des mots qui sont composés d'autres mots. Exemple : bonjour qui est composé du bon et du mot jour. Etant donné une liste de N mots, il s'agit de trouver les mots qui sont composés d'autres mots de la liste. Voici un exemple de listes de mots ;
Bag, sun, day, moon, Sunday, Monday, airbag, MoonBag, air

Exercice-7 : Somme maximale d'un Torus

1- Soit une matrice entière de dimension NxM contenant des valeurs positives négatives ou nulles. Elaborer un algorithme permettant de trouver la sous-matrice donnant la somme maximale. On peut se contenter d'imprimer la valeur de la somme.

2- Un torus est une matrice de dimension NxM circulaire à la fois verticalement que horizontalement. Soit un torus contenant des valeurs entières pouvant être positives négatives ou nulles. Elaborer un algorithme permettant de trouver la sous-matrice donnant la somme maximale.

1	-1	0	0	-4
2	3	-2	-3	2
4	1	-1	5	0
3	-2	1	-3	2
-3	2	4	1	-4

Exemple : pour cette matrice le résultat est 15.
Indication : il faut s'inspirer des algorithmes vus en cours pour déterminer la plus grande somme contigue dans une séquence.

Exercice-8 : Recherche de sous-chaîne : algo. de Knuth-Morris-Pratt

Etant donnée une chaîne de caractères S de longueur N et une sous-chaîne de caractères T de longueur M avec M << N. Le problème consiste à chercher la sous-chaîne T dans la chaîne S.
Exemple d'application : chercher un mot dans un texte.

a- Faire une recherche (voir Wikipédia) sur l'algorithme de Knuth-Morris-Pratt, le comprendre, le programmer et le tester.

b- Estimer sa complexité au meilleur et au pire

Exercice-9 : Recherche de sous-chaîne : algorithme de Rabin-Karp

Etant donnée une chaîne de caractères S de longueur N et une sous-chaîne de caractères T de longueur M avec M << N. Le problème consiste à chercher la sous-chaîne T dans la chaîne S. Exemple d'application : chercher un mot dans un texte.

c- Faire une recherche (voir Wikipédia) sur l'algorithme de Rabin-Karp, le comprendre, le programmer et le tester.

d- Estimer sa complexité au meilleur et au pire

Exercice-10 : Recherche de sous-chaîne : algorithme de Boyer-Moore

Etant donnée une chaîne de caractères S de longueur N et une sous-chaîne de caractères T de longueur M avec M << N. Le problème consiste à chercher la sous-chaîne T dans la chaîne S. Exemple d'application : chercher un mot dans un texte.

e- Faire une recherche (voir Wikipédia) sur l'algorithme de Boyer-Moore, le comprendre, le programmer et le tester.

f- Estimer sa complexité au meilleur et au pire