U.S.T.H.B - 2025-2026
Faculté d'Informatique
N. Bensaou

Master Informatique Visuelle
Algorithmique et Complexité (M1 - MIV)

Série 2 : Complexité des algorithmes

Exercice 1.- Complexité en fonction de deux paramètres.

Déterminer la complexité des algorithmes suivants (par rapport au nombre d'itérations effectuées), où m et n sont deux entiers positifs.

Algorithme A

```
i<--1 ; j<--1;
Tantque (i<= m) et (j <= n)  faire
      i<--i+1;
      j<--j+1;
fait;
```

Algorithme B

```
i<--1 ; j<--1;
Tantque (i<= m) ou (j <= n)  faire
     i<--i+1;
     j<--j+1;
fait;
```

Algorithme C

```
i<--1 ; j<--1;
Tantque(j<=n)  faire
      Si(i<=m)
       alors
        i<--i+1;
       sinon
         j<--j+1;
      fsi;
fait;
```

Algorithme D

```
i<--1 ; j<--1;
Tantque(j<=n)  faire
      Si(i<=m)
       alors
        i<--i+1;
       sinon
         j<--j+1; i<--1;
      fsi;
fait;
```

Exercice 2.- Tableau de permutation.

Déterminer un algorithme qui teste si un tableau de taille n est un "tableau de permutation" (i.e. tous les éléments sont distincts et compris entre 1 et n).

1. Donner un premier algorithme naïf qui soit quadratique.
2. Donner un second algorithme linéaire utilisant un tableau auxiliaire.
3. Donner un troisième algorithme linéaire sans utiliser un tableau auxiliaire.

Exercice 3.- Interclassement de deux tableaux triés.

On dispose de deux tableaux $T_1[1..n]$ et $T_2[1..n]$ dont les éléments sont triés de façon croissante. On veut créer un tableau trié $T_3[1..2n]$ contenant tous les éléments de T_1 et T_2. Pour cela on propose deux algorithmes `Fusion_A` et `Fusion_B`.

— `Fusion_A` : initialise T_3 avec T_1 (déjà trié) et y insère un à un les éléments de T_2 de façon à ce que l'ordre soit respecté.

— `Fusion_B` : remplit T_3 en parcourant simultanément T_1 et T_2 du début jusqu'à leur fin. Soit i_1 et i_2 les indices courant dans T_1 et T_2, on a 3 cas possible :

— Si $T_1[i_1] < T_2[i_2]$ alors mettre $T_1[i_1]$ à la fin de T_3 et avancer dans T_1

— Si $T_1[i_1] > T_2[i_2]$ alors mettre $T_2[i_2]$ à la fin de T_3 et avancer dans T_2

— Sinon mettre $T_1[i_1]$ puis $T_2[i_2]$ à la fin de T_3 et avancer dans T_1 et T_2

1. Ecrire les deux algorithmes et déroulez sur l'exemple :
   T_1= | 1 | 3 | 5 | et T_2 = | 2 | 3 | 4 |
2. Donnez la complexité, au pire des cas, des algorithmes en fonction de la taille des données.
3. Quel algorithme choisissez-vous d'implémenter ?

Exercice 4.- Recherche du maximum et du minimum d'un tableau

Trouver respectivement le maximum et le minimum dans un tableau $A[1..N]$. Un algorithme naïf :

```
Fonction MinMaxNaïf(A : Tableau d'Entier, N : Entier) : Couple d'Entier
Début
i, min, max : Entier;
min <-- A[1]; max <-- A[1];
pour i <-- 2 à N Faire
Si (A[i] < min) alors min <-- A[i]
sinon Si (A[i] > max) Alors max <-- A[i]; Fsi;
Fsi
fait;
retourner(min, max);
Fin
```

Taille de l'entrée : N (le nombre des éléments dans le tableau).
Opérations fondamentales : comparaisons entre les éléments du tableau.

a- Quel est le meilleur cas et donnez la complexité.

b- Quel est le pire des cas et donnez la complexité.

c- Question. Peut-on faire mieux ?

Exercice 5.- Tri d'un tableau ne contenant que des 0 et des 1.

On souhaite trier un tableau T de n entiers appartenant à l'ensemble $0, 1$ de façon à ce que les valeurs nulles soient rangées au début du tableau. Par exemple :

$T =$ | 0 | 1 | 1 | 0 | 0 | ... | 1 | 1 | 1 |

Après l'application de l'algorithme de tri on aura :

$T =$ | 0 | 0 | 0 | 0 | 1 | ... | 1 | 1 | 1 |

Au cours du traitement, une partie des données est déjà triée et le tableau est organisé de la façon suivante :

	1			i			j				n
$T =$	0	0	0	?	?	?	?	1	1	1	1

***Les ? représentent les données non encore traitées.**

a- Que représentent i et j ?

b- Selon la valeur de $T[i]$ quel traitement doit-on effectuer ?

c- Quand doit-on arrêter l'algorithme ?

d- Écrire l'algorithme de tri et donner sa complexité en temps.

Exercice 6.- La recherche séquentielle.

On considère un tableau A de n éléments. On cherche à construire un algorithme permettant de savoir à quel endroit du tableau se trouve une valeur clé. (On suppose que clé est bien dans le tableau et on recherchera la première occurrence de cette valeur.)

a- Donner un algorithme itératif qui résout ce problème. Indiquer et démontrer un invariant de boucle pour cet algorithme.

b- A quoi correspond le pire des cas ? A quoi correspond le meilleur des cas ? En déduire la complexité en O de l'algorithme.

c- Ecrire cet algorithme sous forme récursive.

d- Prouver la validité de cet algorithme.

Exercice 7.- La recherche dichotomique.

On considère un tableau A de n éléments, que l'on suppose trié en ordre croissant. On cherche à construire un algorithme permettant de savoir à quel endroit du tableau se trouve une valeur clé. (On suppose que clé est bien dans le tableau et on recherchera la première occurrence de cette valeur.)

a- Donner un algorithme itératif qui recherche une clé par la méthode dichotomique.

b- Indiquer et démontrer un invariant de boucle pour cet algorithme.

c- On suppose que le tableau A[1..n] contient $n = 2k$ éléments (où k est un entier positif). Combien d'itérations l'algorithme effectuera-t-il au maximum ?

d- En déduire la complexité (en O) de l'algorithme.

e- Ecrire l'algorithme sous forme récursive

f- Déterminer la complexité (en O) de l'algorithme récursif.