

EXERCICE 1.

- Ecrire un algorithme pour vérifier si un entier p ($p \geq 2$) est un nombre premier.
- Donner un algorithme qui calcule le nombre des nombres premiers compris entre 2 et n . (n est un entier supérieur ou égal à 2)

EXERCICE 2.

Soit n un entier strictement positif.

- Ecrire un algorithme qui calcule le plus petit entier p tel que $2^p > n$.
- Quel est le nombre d'itérations (en fonction de n) ?

EXERCICE 3.

- Soit T un tableau de n entiers. Ecrire un algorithme qui renverse le tableau T en faisant des échanges successifs $T[1]$ et $T[n]$, puis $T[2]$ et $T[n-1]$...
- Soient $T1[1..n]$ et $T2[1..n]$ deux tableaux binaires (contenant des 0 et des 1) correspondant à deux entiers $n1$ et $n2$ ayant le même nombre de bits n dans la représentation en base 2. Ecrire un algorithme qui calcule $T3 = T1 + T2$. ($T3[1..n+1]$ correspond à la représentation de $n1+n2$ en binaire)

EXERCICE 4.

- Considérer deux algorithmes A_1 et A_2 avec leurs temps d'exécution respectifs $T_1(n) = 9n^2$ et $T_2(n) = 100n + 96$.
 - En exprimant la complexité des deux algorithmes dans la notation grand-O, quel est le meilleur algorithme ?
 - Pour $n = 10$, votre choix d'algorithme est-il valide ?
 - Déterminer, à partir de quelle valeur de n , l'algorithme choisi est plus efficace que l'autre.
 - Quelle est la complexité de l'algorithme suivant, qui fait appel aux deux algorithmes A_1 et A_2 :
début
 A_1 ;
 A_2 ;
fin
- Comparer les fonctions suivantes selon l'ordre asymptotique Grand-O : $\log(n!)$ et $n \log n$

EXERCICE 5.

- Ecrire un algorithme qui cherche s'il existe deux entiers, dans T , dont la somme est égale à une valeur donnée v . L'algorithme retourne (i,j) si $T[i] + T[j] = v$ ou $(0,0)$ dans le cas où il n'existe pas deux éléments de T dont la somme est v . Quelle est sa complexité ?
- Ecrire une autre version de l'algorithme précédent en ramenant ce problème à la recherche d'un élément dans le tableau T .

EXERCICE 6 .

Une permutation est une bijection de $\{1,2,\dots,n\}$ dans $\{1,2,\dots,n\}$. (S_n est l'ensemble de toutes les permutations de $\{1,2,\dots,n\}$). Exemple d'une permutation σ de S_n .

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 6 & 2 & 4 & 1 & 7 & 3 & 5 & 8 \end{pmatrix}$$

Toute permutation peut être représentée par un tableau.

- Etant donné un tableau T de n entiers. Ecrire un algorithme qui permet de vérifier si T représente bien une permutation. Quelle est sa complexité ?
- Un cycle, d'une permutation σ , contenant i ($1 \leq i \leq n$) est l'ensemble $\{i, \sigma(i), \sigma^2(i), \dots, \sigma^{k-1}(i)\}$, où k est le plus petit entier >0 tel que $\sigma^k(i) = i$; k est appelé longueur de ce cycle .
(les cycles de σ : $\{1, 6, 3, 4\}$; $\{2\}$; $\{5, 7\}$; $\{8\}$)
Ecrire un algorithme qui retourne la longueur du cycle d'un entier i ($1 \leq i \leq n$).
- Donner un algorithme qui calcule le nombre de cycle d'une permutation σ .
Evaluer sa complexité.