Informatique Théorique
TP6 : Complexité et algorithmes de classe P

▶ Exercice 1 : Complexité

1. Prouver que $n^2 + n = \mathcal{O}(n^2)$.
2. Donner l'ordre de grandeur des expressions suivantes :
   (a) $\frac{n^2+3n+1}{n+1}$
   (b) $\frac{n\log(n)+n^2+\log(n)^2}{n+1}$
3. Si $f(n)$ est une fonction en $\mathcal{O}(n)$, les affirmations suivantes sont-elles vraies ?
   (a) $(f(n))^2 = \mathcal{O}(n^2)$
   (b) $2^{f(n)} = \mathcal{O}(2^n)$

▶ Exercice 2 : Complexité

Ecrire un algorithme qui calcule, pour un tableau T de taille n donné en paramètre, la somme :

$$\sum_{i,j=1}^{n} (T(i) - T(j))^2$$

1. Ecrire l'algorithme "naïf" quadratique qui vient en premier à l'esprit.
2. Est-il possible de linéariser cet algorithme ?

▶ Exercice 3 : Complexité

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.

▶ Exercice 4 : Ordres de complexité

Soient 2 programmes A et B qui utilisent 2 algorithmes différents pour résoudre le même problème, à savoir trier une liste de N objets.
Le programme A a besoin d'un temps $t = 10000 * N$ pour trier la liste.
Le programme B a besoin d'un temps $t = 2 * N^2$ pour trier la même liste.

1. Quel programme est plus rapide pour trier une liste de 5 objets ?
2. Quel est la longueur critique N_c telle que pour tout $N > N_c$, le programme A est plus rapide que le B ?
3. Comparer les performances des programmes A et B avec un troisième programme C qui trie la liste en un temps $t = 10 + 0.5 * N^3$.

▶ Exercice 5 : Tri à bulle

Soient n nombres rangés dans un tableau $A[1..n]$. Calculer la complexité de l'algorithme suivant, dit tri par échange ou tri à bulle, qui remet les éléments de A dans l'ordre croissant de leur valeur :

```
procedure BULLE(n: entier; A: tableau[1..n]de entiers);
        pour i de n-1 à 1 faire pas -1
            pour j de 1 à i faire
                si A[j+1] < A[j] alors
                   ech := A[j] ;
                   A[j] := A[j+1] ;
                   A[j+1] := ech ;
                finsi
            finpour
        finpour
finBULLE;
```

▶ Exercice 6 : Complexité

Ecrire un algorithme pour calculer p^n, qui soit de complexité $\mathcal{O}(\log_2(n))$.
Indication : Considérer n écrit en binaire.

▶ Exercice 7 : Tri fusion

Trouver la complexité de l'algorithme du tri fusion qui utilise les fonctions fusion et split ci-dessous.
On supposera que la taille de la liste est une puissance entière de 2, soit $n = 2^k$.

▶ Exercice 8 : Parcours en profondeur

Soit un graphe $G = (S, A)$. Montrer que la procédure du parcours en profondeur sur G appartient à la classe $\mathcal{P}$.

PP(G)
pour chaque sommet u de $V(G)$
  faire $couleur[u] \leftarrow$ blanc
    $\pi(u) =$vide
pour chaque sommet u de $V(G)$
  faire si $couleur[u] =$ blanc
    alors VISITER_PP(u)

VISITER_PP(u)
$couleur[u] \leftarrow$ gris
pour chaque v adjacent à u
  faire si $couleur[v] =$ blanc
    alors $\pi[v] \leftarrow u$
      VISITER_PP(v)
$couleur[v] \leftarrow$ noir

```
fonction fusion(L,K:liste):liste
 si L vide alors
   résultat ← K
 sinon si K vide alors
   résultat ← L
 sinon si elmCourant(L)< elmCourant(K) alors
   résultat ←concatène(elmCourant(L),fusion(suivant(L),K)
 sinon
   résultat ← concatène(elmCourant(K),fusion(L,suivant(K))
```

```
fonction split(L:liste): un couple de listes
 J,K ← listeVide
 si vide(L) alors
   résultat ← (L,L)
 sinon si vide(suivant(L)) alors
   résultat ← (L, listeVide)
 sinon
   i ← 1
   tant que non listeVide(L) faire
     e = retire(L)
     si impair(i) alors
       ajoute (e,J)
     sinon
       ajoute (e,K)
     finsi
     i ← i + 1
   fin tant que
   résultat ← (J,K)
 finsi
```

```
procédure triFusion(L:liste): liste
 J,K ← listeVide
 si vide(L) alors
   résultat ← L
 sinon si vide(suivant(L)) alors
   résultat ← L
 sinon
   (J,K) = split(L)
   J ← triFusion(J)
   K ← triFusion(K)
   résultat ← fusion(J,K)
 finsi
```