

Exercices de révision

Exercice 1.

Ordre de grandeur

✎ Pour les fonctions f et g suivantes, dites si $f \in O(g)$ et si $g \in O(f)$.

1. $f(n) = 2^{5n+2}$ et $g(n) = 2^{4n+3}$;
2. $f(n) = n^8$ et $g(n) = n^7 \sqrt{n}$;
3. $f(n) = \ln(5e^{n^2})$ et $g(n) = \ln(3e^n)$.

Exercice 2.

Complexité

✎ Que vaut a à la fin de l'algorithme suivant, en fonction de n ? Quelle est la complexité de l'algorithme?

```

a := 5
Pour i de 1 à n faire
  Pour j de i à n faire
    Pour k de 1 à n faire
      a := a+3
  
```

Exercice 3.

✎ Écrire l'algorithme de la fusion de deux tableaux dans le tri fusion. Plus précisément, écrire une fonction `fusion` prenant en argument deux tableaux A et B d'entiers triés par ordre croissant et renvoyant un tableau C contenant les éléments de A et de B triés par ordre croissant. Prouver la correction de votre algorithme en donnant un invariant. Évaluer sa complexité.

Exercice 4.

Tri par groupage (bucket sort)

Supposons que l'on ait à trier n entiers (pas forcément distincts) appartenant à l'intervalle $[1, k]$ pour un entier k fixé. Par exemple, les âges des personnes d'un groupe de taille n (pour lequel on peut raisonnablement fixer $k = 150$ par exemple). Il faut imaginer n grand et k petit (constant). Soit A le tableau contenant les n entiers.

Dans ce cas, on peut calculer un tableau T de taille k tel que $T[i]$ contienne le nombre de fois que la valeur i apparaît dans A (éventuellement $T[i] = 0$). Expliquer comment se servir de T pour trier A par ordre croissant en temps $O(n + k)$.

Écrire l'algorithme complet qui calcule T puis trie A . Évaluer sa complexité en fonction de k et de n ; qu'obtient-on si k est constant (ou petit devant n)? Prouver la correction de l'algorithme.

Exercice 5.

Tri bulation (stooge sort)

On considère l'algorithme de tri suivant (étrange et inefficace, appelé *stooge sort*) pour un tableau T entre les indices i et j .

```

Tri (T, i, j) :
  Si T[i] > T[j] alors // (alors on échange T[i] et T[j])
    a := T[i]
    T[i] := T[j]
    T[j] := a
  Si j > i+1 alors
    k = (j - i + 1)/3 // Le tiers de l'intervalle
    Tri (T, i, j-k) // On trie les deux premiers tiers
    Tri (T, i+k, j) // On trie les deux derniers tiers
    Tri (T, i, j-k) // On trie encore les deux premiers tiers

```

Expliquer intuitivement pourquoi ce tri est correct. Écrire l'équation de récurrence de sa complexité. Évaluer sa complexité. Comparer avec les autres algorithmes de tri vus en cours.