Spé Bio 1 & 2 - Lyc. Champollion — Octobre/Novembre 2023

Informatique - Python - TP N°4
Les tris "rapides" récursifs : Quicksort et tri fusion

Objectifs :
- Connaître l'algorithme de "**tri rapide**" ("**Quicksort**") qui utilise la récursivité.
- Grâce à des chronométrages, observer l'efficacité du Quicksort et ses limites.

Remarque :on utilisera à quelques endroits de ce TP le *slicing* : L[i :j] est la sous-liste [L[i], L[i+1], ..., L[j-1]] des éléments de la liste L d'indice compris entre i et j (j exclu).

1. Le Tri Rapide, ou Quicksort

(a) Codage du tri rapide

Ce tri est basé sur deux fonctions, dont une **récursive**, que nous allons écrire en deux étapes.
On travaille sur une liste L, de longueur N.

- **cas de base** de la fonction récursive : on renvoie **la liste elle-même** si elle contient **au plus un élément** (dans ce cas, elle est déjà triée).
- sinon :
  - on choisit un **pivot** P, c'est-à-dire un élément de la liste ; ici nous choisirons (arbitrairement) le premier élément de la liste, soit L[0].
  - on **partitionne** la liste L privée du pivot L[0], donc L[1 : N] en deux sous-listes :
    L1, formée des éléments de L[1 : N] strictement inférieurs à P = L[0] ;
    L2, formée des éléments de L[1 : N] supérieurs ou égaux à P.
    → Cette étape est confiée à une fonction partition(L), qui renvoie L1 et L2 ;
  - on recolle les morceaux, c'est-à-dire qu'on renvoie la liste L1 + [P] + L2.
  - la dernière étape consiste à **rendre cette fonction récursive**, c'est à dire à appliquer le **même** principe de tri (déjà écrit) à L1 et L2 !

a. Écrire une fonction partition de paramètre L, qui définit P et renvoie les 2 listes L1 et L2 définies ci-dessus.

b. Écrire une fonction tri_rapide(L), non récursive dans un premier temps.

Tester dans le shell sur la liste [2] puis [5,1,8,7,3,9,2,5,0,8].

c. Dernière étape : rendre cette fonction **récursive** ; il suffit pour cela d'appliquer tri_rapide à L1 et à L2.

Tester sur des listes simples.

(b) Tri "rapide" ? - Quand tout se passe bien...

Dans un programme principal, chronométrer et faire afficher les temps mis pour trier (avec tri_rapide) une liste aléatoire de N nombres, pour N=2000,4000,8000,...256000 (on peut bien sûr réutiliser la fonction temps_tri du TP précédent).
Noter le temps obtenu pour N=16000.
Représenter cela par un graphique "temps=f(N)".

Observations ?

(c) Les limites du tri rapide

• Placer au début de votre fichier from sys import setrecursionlimit
Imposer au début du programme principal : setrecursionlimit(17000)
(nombre maximum d'appels récursifs autorisés).
• Tester (dans le principal ou dans le shell) votre tri_rapide avec une liste de 16000 valeurs **déjà triée**, puis avec une liste de 16000 valeurs **triée à l'envers**.
Remarque : pour obtenir une liste triée (resp. triée à l'envers) il suffit de fabriquer une liste L de valeurs aléatoires comme précédemment, puis de la trier avec L.sort() (resp. L.sort(reverse=True)avant le chronométrage.

Que se passe-t-il ? **Pourquoi ?**

(d) Un remède ?

Proposer une solution pour éviter au tri rapide les deux "pièges" (pires cas) de la liste déjà triée et de la liste triée à l'envers.
Tester cette solution en écrivant une fonction tri_rapide2(L) : vérifier qu'elle se comporte bien dans tous les cas.

2. (*) **Le tri fusion (merge sort)**

Principe : on divise la liste en deux sous-listes de même taille, on les trie récursivement, puis on les "fusionne", l'opération de fusion étant très rapide car les deux listes à fusionner sont triées.

Ce tri est, comme le précédent, basé sur le principe "**diviser pour régner**" , mais ici employé toujours de manière optimale car la taille de la liste à trier est **divisée par deux** à chaque appel récursif (*comme dans la méthode de dichotomie*), alors qu'avec *Quicksort* les tailles des sous-listes peuvent être déséquilibrées.

(a) **La fonction de fusion**

Écrire une fonction `fusion(L1, L2)` qui prend en paramètres deux listes triées dans l'ordre croissant.

Cette fonction doit renvoyer une liste `L` **triée** qui contient les éléments de la réunion des deux listes (sa longueur est la somme des longueurs des deux listes).

Par exemple, `fusion([1,2,5,8],[3,4,5,9,10])` doit renvoyer `[1,2,3,4,5,5,8,9,10]`.

Aide : Soit `n1` la longueur de `L1` et `n2` la longueur de `L2`. Le plus simple est d'initialiser la liste "résultat" `L` à `[ ]` puis de faire une boucle sur le numéro `k` de l'élément à insérer dans la liste `L` finale (`for k in range(n1+n2) : ...`) ; on utilise deux compteurs `i` et `j` qui repèrent la position du prochain élément à regarder dans chaque liste.

Ainsi à chaque itération de la boucle, soit on ajoute `L1[i]` dans `L` (et on incrémente `i`), soit on ajoute `L2[j]` dans `L` (et on incrémente `j`).

Il faut aussi tenir compte de l'épuisement potentiel d'une des listes : on ajoute `L1[i]` dans `L` si `j==n2` ou si `j<n2` et `L1[i] <= L2[j]`.

(b) **Le tri fusion récursif**

Ecrire la fonction `tri_fusion(L)` :

Soit `n` la longueur de la liste `L`.

- Cas de base : si `L` contient au plus un élément, on renvoie `L` car elle est déjà triée ;

- Cas général :

- on crée les listes `L1` (formée des `n//2` premiers éléments de `L`) et `L2` (formée des éléments suivants).
- Puis on renvoie la liste obtenue par fusion de (L1 triée) et (L2 triée), L1 et L2 étant triées récursivement par la fonction `tri_fusion(L)`.

(c) **Chronométrages**

Chronométrer et représenter graphiquement les temps obtenus pour différentes tailles de listes (reprendre les valeurs de `N` utilisées pour le test du tri Quicksort).

Vérifier que ce tri reste efficace sur une liste triée ou triée à l'envers.