Tris
Révisions pcsi - Enoncé

Les tris sont des opérations très fréquentes en informatique : tri lexicographique (annuaires), tri de requêtes (par prix sur des sites commerciaux par exemple), tri par pertinence (moteurs de recherche)…

Stratégies de tri
Une stratégie particulièrement efficace est la stratégie « diviser pour régner ». Cette stratégie a été rencontrée en 1ère année lors de la recherche dichotomique d'un élément dans un tableau ***préalablement trié***, lors de la recherche dichotomique du zéro d'une fonction ou encore avec la méthode récursive d'exponentiation rapide.

Les deux algorithmes de tri proposés sont **dichotomiques** et **récursifs**.

Dans toute la suite, il s'agit de trier par ordre *croissant* un tableau de nombres.
Il existe plusieurs versions de ces algorithmes de tri.

Dans la suite, on présente des versions destinées aux **listes** :

— pop, remove, append, techniques de slicing… disponibles ;
— Rappel : la concaténation des listes L1 et L2 s'effectue par L1 + L2.

Complexité des algorithmes de tri

📚 **Retenir** (résultats admis).

		Tri insertion	Tri rapide	Tri rapide
Algorithme		Itératif	Récursif	Récursif
Stratégie			Diviser pour régner	
Complexité spatiale		En place	Mémoire supplémentaire (*)	
Complexité temporelle	Au pire	$O(n^2)$	$O(n^2)$	$O(n\ ln(n))$
	Moyenne	$O(n^2)$	$O(n\ ln(n))$	$O(n\ ln(n))$

(*) La gestion de la mémoire dans ces versions naïves n'est pas optimale (l'allocation de mémoire à chaque appel ralentit beaucoup ces algorithmes, il est plus efficace d'allouer une fois pour toute un tableau de taille n et de passer ce tableau en argument).

Meilleure complexité temporelle moyenne : $O(n\ ln(n))$.

Tri rapide (Quick sort)

Principe du tri rapide

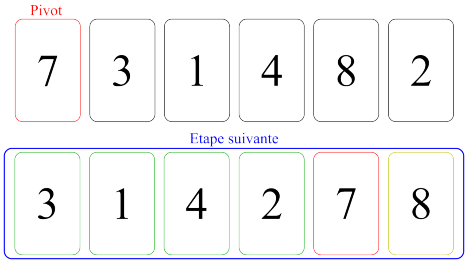
💡 Il s'agit d'un tri **récursif dichotomique** d'un tableau T.

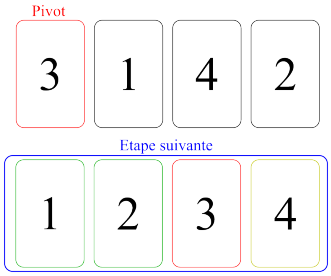
— On partitionne le tableau T autour d'un pivot (élément de T choisi aléatoirement) : on constitue un tableau T_{inf} de valeurs inférieures au égales au pivot (sans chercher à trier ces valeurs), suivi du pivot qui se trouve à sa place définitive, suivi d'un tableau T_{sup} de valeurs strictement supérieures au pivot.
— On itère le procédé sur chacun des deux sous-tableaux T_{inf} et T_{sup} (jusqu'à ce qu'ils ne contiennent plus qu'un seul élément).

👁️ Exemple type jeu de cartes

Dans cet exemple, on choisit comme pivot le premier élément de la liste en cours de traitement (carte rouge).
Etape 1 : on place le pivot au « bon endroit », les autres éléments ne sont pas triés entre eux, ils sont seulement répartis de part et d'autre du pivot (cartes en vert à gauche du pivot et jaune à droite).

Puis la fonction s'appelle elle-même pour trier les deux tableaux T_{inf} = [3,1,4,2] et T_{sup} = [8].
Etape 2 : T_{inf} et T_{sup} (déjà trié ici) sont triés sur le même principe





Et ainsi de suite.
A chaque étape, un tri partiel est effectué, **seul un élément (le pivot, en rouge) est trié** (i.e. les autres éléments sont répartis de part et d'autre du pivot).
A chaque étape, la liste transformée est la forme : [éléments plus petits que le pivot] + [pivot] + [éléments plus grands que le pivot].

A la fin des appels récursifs, le liste initiale est découpée en listes composées d'un **unique élément** mais ces éléments sont classés les uns par rapport aux autres, et la liste complète est reconstituée lors de la phase de « remontée » lorsque les sous-listes de l'étape $i+1$ sont transmises à l'étape i.

Algorithme

💡 Le tableau T est trié récursivement :

- si T est vide ou admet un unique élément, il est trié (critère d'arrêt) : la fonction renvoie T ;
- sinon un tri partiel est effectué par rapport au pivot :
  1. on choisit un élément de T, appelé pivot, qu'on retire de T ;
  2. on construit un sous-tableau Tinf constitué des éléments plus petits que le pivot ;
     on construit un sous-tableau Tsup constitué des éléments plus grands que le pivot ;
  3. le résultat est la *concaténation* de Tinf *récursivement trié*, du pivot et de Tsup *récursivement trié* (i.e. la fonction s'appelle elle-même pour effectuer le tri récursif).

Chacun des termes en italique à l'étape c/ doit être traduit dans le code (concaténer signifie « mettre bout à bout »).

Comprendre : les étapes a/ et b/ correspondent à la **dichotomie**, l'étape c/ correspond au **tri partiel**.

Fonction de tri rapide

✏️ Écrire la fonction `tri_rapide(t:list)->list` en choisissant comme pivot le dernier élément du tableau t (on pourra utiliser la méthode pop( )).
Tester la fonction.

Tri fusion (Merge sort)

Fusion ou interclassement de deux tableaux triés

On considère deux tableaux T_1 et T_2 ***supposés triés***.
L'opération de **fusion** ou **interclassement** consiste à construire un tableau T, trié, à partir des valeurs de T_1 et T_2.

0.0.1 Algorithme récursif de fusion

- Si T_1 est vide alors T_2 est renvoyé puisque déjà trié et réciproquement (critères d'arrêt).
- Sinon, on effectue un ***tri partiel*** :
  1. on sélectionne le plus petit élément e parmi les deux tableaux T_1 et T_2 (triés) ;
  2. on renvoie le tableau constitué de e et de la ***fusion*** d'un tableau diminué de cet élément et du tableau intact (appel récursif).

Sur une feuille, illustrer le principe de cet algorithme à l'aide d'un exemple : choisir deux listes triées et donner le résultat de la fusion de ces deux listes.

Fonction fusion

✏️ Écrire une fonction `fusion(t1:list, t2:list)->list` qui fusionne récursivement les tableaux triés t1 et t2.
Tester la fonction.

Principe du tri fusion

Il reste à écrire la fonction tri_fusion (récursive) qui réalise les dichotomies successives puis le tri grâce à la fonction fusion.

- Si T est vide ou admet un unique élément, il est trié (critère d'arrêt).
- Sinon :
  1. le tableau T est coupé en deux sous-tableaux T[0 : m] et T[m :] (où m = len(T)//2), étape de dichotomie ;
  2. on fusionne, étape de tri partiel, (fonction ci-dessus) les deux sous-tableaux triés récursivement (appel récursif à tri_fusion).

La fonction tri_fusion découpe le tableau de départ en **singletons** qui sont ensuite **fusionnés** (et donc triés) peu à peu lors de la phase de remontée.

Fonction de tri fusion

✏️ Écrire une fonction `tri_fusion(t:list)->list`.
Tester la fonction.

Exercices

✏️ Ecrire une version itérative de l'algorithme de fusion.

✏️ Ecrire une fonction `tri_rapideA(t:list)->list` dans laquelle le pivot est choisi aléatoirement.
Rappel : from random import randint puis randint(a,b) renvoie un nombre $a \leq n \leq b$.