Questions

- Algo présentés sur valeur entière. En pratique ?
  Trie tout, tant qu'on a une fonction de comparaison.
  Tri peut dépendre de la structure (par ex pour pivot on veut accès efficace), mais pas du type des données.
- Coût minimal : O(n log n)?
  What about tri par dénombrement, que sur entiers ?
  Tri radix requiert entier aussi. En pratique, possible de faire en O(n) si on fait des hypothèses sur les données.
- Influence de la liste des pas ?
  Il faut pas que y en ait trop (ça sert à rien de faire n pas), mais suffisamment pour en gros trier les éléments.
  Réflexion sur la complexité ?
  Au mieux semble être log carré de n.
- Lister les propriétés qu'on cherche à atteindre
  - le temps
  - la place en mémoire
  - modularité (s'adapte à toutes les structures de donnée)

Graal ?
Tri général, en n log n, qui ne stocke pas plus que les éléments à trier

- Si on note k le pas, après chaque étape, la liste est k-triée ? Oui.
  Discussion sur différence avec le tri à bulle et complexité du tri présenté.
- Remarque : le mot récursif n'apparaît pas, alors que c'est le cas de l'implémentation.
  Attention : ce n'est jamais un algorithme qui est récursif, c'est l'implémentation.
- Différence tri par tas et tri ABR ?

Tri par tas on ajoute puis on supprime. Ti ABR on ajoute puis on parcourt.

Remarques

Parler de parallélisation du tri, pour avoir plus de choses à dire ?
Pourquoi pas, mais il les connaît moins bien
Certains tris sont parallélisables, mais pas tous.

Question importante : choisir le niveau de difficulté où on se pose pour le sujet de leçon choisi.
Dans ce cas, la version de base ce serait les tris les plus faciles. Thomas est déjà allé plus loin, et la parallélisation serait encore un cran au-dessus.
Pour être premier•e à l'agreg, il faut un truc difficile. Mais attention à ne pas prendre trop le

risque de se planter. On peut aussi la jouer safe avec quelque chose de plus simple, mais parfaitement maîtrisé.

Plan était énumératif, mais vu le sujet c'est compliqué.
Faire introduction. Y a plusieurs algos, on s'intéresse à telles propriétés et on vise tel truc

→ plan.

Quand on présente du code, il faut au moins le compiler et montrer un exemple. Ici, on aurait aimé le voir tourner, sur plusieurs étapes.

Pourquoi pas mieux que n logn ? Réponse

Axel : on peut faire le tri par tas en place dans un tableau

⇒ c'est essentiellement le Graal

A quoi sert le shell sort ?
Il est simple, et si y a pas trop d'éléments la constante devant le n log n est assez petite.

Hugo : autre critère : stabilité

Si on parcourt la liste en partant de la fin, dans l'implen du développement ?
Ça va pas marcher. Y a des cas où ça peut très mal se passer.

Est-ce que des algos choisissent le pas en fonction du truc à trier ?
Non, il faudrait pré-trier, essentiellement.