NSI 1ère - Algorithmique - Tris 1

QK

Plan

1. cours/td : activité boites,
2. résumé des 2 algos,
3. faire tourner sur des ex sans indice,
4. animation pgzero

Trier

Trier : définition.

Algorithme de **tri**

Algorithme qui, partant d'une liste, renvoie une version ordonnée de la liste.

$$[5, 1, 4, 3, 2] \rightarrow [1, 2, 3, 4, 5]$$

Certains algorithme "ne renvoient rien"... c'est le *tableau de départ* qui est modifié.

Objectifs du programme

Contenus Tris par insertion, par sélection

Capacités attendues Écrire un algorithme de tri. Décrire un invariant de boucle qui prouve la correction des tris par insertion, par sélection

Commentaires

On montre que leur coût est quadratique

Trier : de nombreux algorithmes

Il existe de nombreux algorithmes de tri.

- **Tri par insertion** -> 1ère
- **Tri par sélection** -> 1ère
- Tri à bulle
- Tri rapide
- **Tri fusion** -> Terminale
- Tri par tas
- Tri par comparaison
- Smoothsort
- **Timsort** -> Python

Que va-t-on faire ?

1. Dégager les algorithmes "naturels" du tri par insertion et par sélection
2. Étudier ces algorithmes (preuve et complexité)
3. Les programmer en Python
4. Comparer leur efficacité avec d'autres algorithmes de tris (timsort, tri rapide, tri à bulle)

Pourquoi étudier les algorithmes de tri ?

Trier est une opération fréquente, certains algorithmes (dichotomie par exemple) partent d'un tableau déjà trié.

Tous les langages "haut niveau" proposent une fonction native pour trier les tableaux. En terminale on utilisera apprendra à faire des sélections sur des bases de données du type "quels sont les 5 films les plus vus au cinéma ?" ou bien "déterminer les 100 derniers inscrits à un jeu en ligne".

Ces sélections nécessitent un tri (des films par nombre de spectateurs, des joueurs par date d'inscription).

Notre objectif n'est pas *d'utiliser* en pratique nos algorithmes mais de comprendre leur fonctionnement.

Non seulement ils sont d'excellents candidats pour des questions à l'examen mais les principes mis en oeuvres sont utilisés dans nombreux algorithmes.

Tous les algorithmes de tri ne se valent pas. Au prix de transformations un peu plus élaborées on parvient à les rendre beaucoup plus rapides.

Nous allons les étudier de plus en plus précisemment.

Première partie

Activité à la main

Trier des boites

On dispose de boites de poids différent,

L'objectif est de décrire un algorithme "naturel" qui réponde au problème :

- **comment les ranger par masse croissante ?**
- Contrainte : **on ne peut qu'effectuer des comparaisons par paire.**

Activité à la main

Trier des boites

L'important, c'est **comment ?**

Description de la séquence

Vous avez 25 minutes pour :

1. écrire un algorithme "papier" qui réalise le tri des boites
2. permette à n'importe qui de le reproduire et d'aboutir au résultat
3. aucune explication supplémentaire ne doit être apportée

Résumé de la séquence

1. Deux algorithmes "naturels" se dégagent
2. Pas évident quand on ne dispose que d'une comparaison
3. généralisable à tout ce qui peut se "comparer" (ex : mots par longueur, fichiers par taille etc.)

Tri par sélection

Le tri par sélection est l'un des deux algorithmes de tris étudiés cette année.

Tri par sélection

En français

```
Je débute avec un alignement vide de boîtes triées
Tant qu'il y a des boîtes non triées :
   Je cherche la plus légère parmi les boîtes non triées
```

```
   Je la place à la suite des boîtes déjà triées.
fin Tant que
```

Le tri par sélection

La plus légère des boites parmi les boîtes non triées ?

```
Entrée : Des boîtes
Sortie : La boite la plus légère
Effet de Bord : Enlève une boite

Je prends une boîte (main gauche)
Pour chacune des autres (main droite) :
    Si elle est plus légère que la boite dans ma main,
    Alors j'échange.
    Fin Si
    Je mets l'autre de côté.
Fin Pour
```

Tri par insertion

Le tri par insertion

Le tri par insertion est le tri du joueur de cartes, il est très proche du tri par sélection, et dispose d'ailleurs de propriétés similaires.

En français

Version 1

```
Je débute avec un alignement vide de boîtes triées
Tant qu'il y a des boîtes non triées :
   Je choisis une boite
   Je l'insère dans l'alignement de telle sorte qu'il reste trié
fin Tant
```

Fonction : insérer un élément dans un alignement trié.

```
Entrees : Un alignement de boîtes trié, une boîte b
Sortie : aucune
Effet de bord: l'alignement reste trié

Prends la boîte la plus à droite (la plus lourde)
Tant que cette boite est plus lourde que b
     passe à la boite suivante
insère b à la droite de la boite courante
```

Le tri par sélection : tableau de nombres

Tri par sélection

On commence avec une liste déjà triée vide. On itère sur la liste et, à chaque tour on range le plus petit élément de la liste non triée à droite de la liste triée.

Tri *stable* : il ne change pas l'ordre de deux éléments "égaux" Tri en *place* : il n'utilise pas plus de mémoire

Exemple

Triés	Non Triés	Plus petit restant
()	(1, 3, 4, 2)	(1)
(1)	(3, 4, 2)	(2)
(1, 2)	(3, 4)	(3)
(1, 2, 3)	(4)	(4)
(1, 2, 3, 4)		

Le tri par insertion

Exercice :

Reprendre le tableau de nombres `[1, 3, 4, 2]` et donner les étapes :

- au départ,
- après chaque échange,
- à la fin.

Présentation d'une animation réalisée avec Pygame zero.