NSI 1ère - Algorithmique - Tris 1

QK

Trier

Trier : définition.

Algorithme de **tri**

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

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

Objectifs du programme

Contenus	Capacités attendues	Commentaires
Tris par insertion, par sélection	Écrire un algorithme de tri. Décrire un invariant de boucle qui prouve la correction des tris par insertion, par sélection	La terminaison de ces algorithmes est à justifier.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** -> Tale (?)
- Tri par tas
- Tri par comparaison
- Smoothsort
- **Timsort** -> Python

Que va-t-on faire ?

1ère partie

Introduction

1. Activité “à la main” : dégager les algorithmes les plus naturels
2. Faire tourner les algorithmes en langage naturel, dans un tableau

seconde partie

1. Les algorithmes formalisés
2. Les faire tourner sur des tableaux de nombres

troisième partie

- prouver qu'ils sont corrects et se terminent (invariant, terminaison)
- Étudier la complexité (~vitesse d'exécution en fonction de la taille)

quatrième partie

- les programmer en Python
- Comparer l'efficacité de différents tris
- Compléments éventuels

Première partie

Activité à la main

Trier des boites

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

- on dispose de boites de poids différent,
- **comment les ranger par masse croissante ?**
- Contrainte : **on ne peut les comparer qu'une à une.**

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

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

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
```

Le tri par insertion

Il est important ici de présenter un algorithme d'insertion dans un alignement trié.

```
Entree : Un alignement de boîtes trié, une boîte b
Sortie : rien
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)		