Les tris

Tri simple

tri1(in collection **V**={$\mathbf{v}_1$,..,$\mathbf{v}_n$},out collection **R**={})

Tant que **V** non vide faire

trouver **k** tel que $\mathbf{v}_k$ est le minimum

supprimer $\mathbf{v}_k$ de **V**

ajouter $\mathbf{v}_k$ en fin de **R**

Fin tant que

Tri simple : coût

- Avec V, un tableau
  - Le coût est décroissant à chaque itération
  - Trouver, supprimer, ajouter : O(n)
  - Coût total : O(n²)
- Avec V, une liste
  - Le coût est décroissant à chaque itération
  - Trouver : O(n)
  - Supprimer, ajouter : O(1)
  - Coût total : O(n²)

Tri par AVL

- Principe de l'algorithme

```
Pour chaque élément e faire
   Insérer e dans  AVL
Fin pour
Parcourir AVL en profondeur // infixé
```

Tri par AVL

- Principe de l'algorithme

  Pour chaque élément **e** faire

  Insérer **e** dans **AVL**

  Fin pour

  Parcourir **AVL** en profondeur // parcours infixé

- Problème :
  - nécessite une structure supplémentaire

Tris par tas (heapsort)

- Principe
  - Ajouter les éléments 1 à 1 dans le tas
  - Supprimer les éléments 1 à 1 du tas

$$O(n \ln(n))$$

- Possibilité de faire sur place

Diviser pour régner : principes

- Diviser le problème en sous-problèmes
  - Identiques
  - De tailles inférieures

- Opérations de combinaison des résultats

Diviser pour régner : principes

- Diviser le problème en sous-problèmes
  - Identiques
  - De tailles inférieures

- Opérations de combinaison des résultats

Diviser en *b* pour régner : le coût

$$C(n) = l\, C\left(\frac{n}{b}\right) + O\left(n^k\right)$$

$$C(n) = \begin{cases} O\left(n^k\right) & \text{si} \quad l < b^k \\ O\left(n^k \ln(n)\right) & \text{si} \quad l = b^k \\ O\left(n^{\ln_b(l)}\right) & \text{si} \quad l > b^k \end{cases}$$

Diviser pour Régner : tri par segmentation

- Diviser l'ensemble en 2
- Appeler le tri sur chaque des 2 sous-ensembles
- Mélanger

Diviser pour Régner : tri par segmentation

```
tri2(in collection V={v1,..,vn}, out collection
R={})
   Si n <= 0 faire
      R ← V
   Sinon faire
      U ← {v1,..,vn/2}
      W ← {vn/2+1,..,vn}
      tri2(U,R1)
      tri2(W,R2)
      mélanger R1 et R2 dans R
```

Diviser pour Régner : tri par segmentation

```
tri2(in collection V={v1,..,vn}, out collection
R={})
   Si n < 0 faire
      R ← V
   Sinon faire
      U ← {v1,..,vn/2}
      W ← {vn/2+1,..,vn}
      tri2(U,R1)
      tri2(W,R2)
      mélanger R1 et R2 dans R
   Fin si
```

O(n ln(n))

Diviser pour régner : tri par pivot (quicksort)

- Prendre un pivot

- Échanger les éléments jusqu'à que
  - A gauche de pivot : éléments plus petits
  - A droite de pivot : éléments plus grands

- Trier les éléments à gauche et à droite du pivot

Fonction pivoter

```
pivoter(inout collection V={v_i,..,v_j}, inout indice k)
   v_k ↔ v_j // le pivot à la fin
   // les éléments plus petits que vk au début
   s ← i;
   Pour p=i..j faire
      Si v_p < v_j faire // vj est le pivot
         v_p ↔ v_s
         s ← s+1
      Fin si
   Fin pour
   v_s ↔ v_j // le pivot à la bonne place
   k ← s
```

quicksort

```
quicksort(inout collection V={v_i,..,v_j})
  Si #V > 1 faire
     k ←  i // le pivot à la fin
     pivoter(V,k)
     quicksort({v_i,..,v_k-1})
     quicksort({v_k+1,..,v_j})
  Fin si
```

Au pire : $O(n^2)$

Au mieux, en moyenne : $O(n \ln(n))$

Meilleur cas

- Le pivot est le médian
  - Autant d'élément à gauche qu'à droite.

- Amélioration, trouver le médian
  - Mais en O(n)

- Autre solution : médian de 3 éléments
  - Premier, milieu et dernier