

Algorithmique avancée

Algorithmes de tri

Youness LAGHOUAOUTA

Institut National des postes et télécommunications
laghouaouta@inpt.ac.ma

Février 2022



- Comprendre l'utilité des outils d'analyse des algorithmes
- Choisir les bonnes structures de données pour concevoir un algorithme
- Concevoir de nouvelles structures de données efficaces
- Comprendre certaines stratégies et les utiliser pour concevoir des algorithmes efficaces

- ① Analyse des algorithmes + TD1
- ② **Algorithmes de tri + TD2**
- ③ Structures de données linéaires 1 + TD3
- ④ TD4
- ⑤ Dictionnaires + TD5
- ⑥ Arbres + TD6
- ⑦ Paradigmes et stratégies algorithmiques

Problème

- Organiser un ensemble d'objets selon un ordre déterminé
- Relation d'ordre se base sur la comparaison des clés

Pourquoi trier?

Brique de base de plusieurs algorithmes:

- Recherche d'un élément dans une liste
- Recherche des i premiers éléments dans une liste
- Eliminer les doublons d'une liste
- ...

Types des algorithmes de tri

- Tris itératifs
 - Tri par sélection
 - Tri à bulles
 - Tri par insertion
 - Tri par dénombrement
 - Tri par base
- Tris récursifs
 - Tri fusion
 - Tri rapide
- Tris optimisés

Principe

- Chercher le minimum dans la liste
- Echanger le minimum avec le premier élément
- Recommencer avec la liste restante jusqu'à épuisement

15, 8, 4, 2, 17, 10

Principe

- Chercher le minimum dans la liste
- Echanger le minimum avec le premier élément
- Recommencer avec la liste restante jusqu'à épuisement

15, 8, 4, 2, 17, 10

15, 8, 4, 2, 17, 10

Principe

- Chercher le minimum dans la liste
- Echanger le minimum avec le premier élément
- Recommencer avec la liste restante jusqu'à épuisement

15, 8, 4, 2, 17, 10

15, 8, 4, 2, 17, 10

2, 8, 4, 15, 17, 10

Principe

- Chercher le minimum dans la liste
- Echanger le minimum avec le premier élément
- Recommencer avec la liste restante jusqu'à épuisement

15, 8, 4, 2, 17, 10

15, 8, 4, 2, 17, 10

2, 8, 4, 15, 17, 10

2, 8, 4, 15, 17, 10

Principe

- Chercher le minimum dans la liste
- Echanger le minimum avec le premier élément
- Recommencer avec la liste restante jusqu'à épuisement

15, 8, 4, 2, 17, 10

15, 8, 4, 2, 17, 10

2, 8, 4, 15, 17, 10

2, 8, 4, 15, 17, 10

2, 4, 8, 15, 17, 10

Tri par sélection - Algorithme

```
fonction triParSelection
entrée t : tableau d'entiers
l:=taille(t)
pour pr allant de 1 à (l-1):
    pmin := pr
    pour i allant de (pr+1) à l
        si t[i]<t[pmin] alors
            pmin := i
        permuter(t , pr , pmin)
```

Principe

- Parcourir les éléments de la liste et comparer les éléments consécutifs
- Echanger les éléments qui ne sont pas en ordre (le plus grand élément prend sa place définitive)
- Recommencer avec la liste restante (sans le dernier élément) jusqu'à épuisement

1, 8, 4, 15, 17, 10

Principe

- Parcourir les éléments de la liste et comparer les éléments consécutifs
- Echanger les éléments qui ne sont pas en ordre (le plus grand élément prend sa place définitive)
- Recommencer avec la liste restante (sans le dernier élément) jusqu'à épuisement

1, 8, 4, 15, 17, 10

1, 8, 4, 15, 17, 10

Principe

- Parcourir les éléments de la liste et comparer les éléments consécutifs
- Echanger les éléments qui ne sont pas en ordre (le plus grand élément prend sa place définitive)
- Recommencer avec la liste restante (sans le dernier élément) jusqu'à épuisement

1, 8, 4, 15, 17, 10

1, **8**, 4, 15, 17, 10

1, **8**, **4**, 15, 17, 10

Principe

- Parcourir les éléments de la liste et comparer les éléments consécutifs
- Echanger les éléments qui ne sont pas en ordre (le plus grand élément prend sa place définitive)
- Recommencer avec la liste restante (sans le dernier élément) jusqu'à épuisement

1, 8, 4, 15, 17, 10

1, 8, 4, 15, 17, 10

1, 8, 4, 15, 17, 10

1, 4, 8, 15, 17, 10

Principe

- Parcourir les éléments de la liste et comparer les éléments consécutifs
- Echanger les éléments qui ne sont pas en ordre (le plus grand élément prend sa place définitive)
- Recommencer avec la liste restante (sans le dernier élément) jusqu'à épuisement

1, 8, 4, 15, 17, 10

1, **8**, 4, 15, 17, 10

1, **8**, **4**, 15, 17, 10

1, **4**, **8**, 15, 17, 10

1, 4, **8**, **15**, 17, 10

Principe

- Parcourir les éléments de la liste et comparer les éléments consécutifs
- Echanger les éléments qui ne sont pas en ordre (le plus grand élément prend sa place définitive)
- Recommencer avec la liste restante (sans le dernier élément) jusqu'à épuisement

1, 8, 4, 15, 17, 10

1, 8, 4, 15, 17, 10

1, 8, 4, 15, 17, 10

1, 4, 8, 15, 17, 10

1, 4, 8, 15, 17, 10

1, 4, 8, 15, 17, 10

Principe

- Parcourir les éléments de la liste et comparer les éléments consécutifs
- Echanger les éléments qui ne sont pas en ordre (le plus grand élément prend sa place définitive)
- Recommencer avec la liste restante (sans le dernier élément) jusqu'à épuisement

1, 8, 4, 15, 17, 10

1, 8, 4, 15, 17, 10

1, 8, 4, 15, 17, 10

1, 4, 8, 15, 17, 10

1, 4, 8, 15, 17, 10

1, 4, 8, 15, 17, 10

1, 4, 8, 15, 17, 10

Principe

- Parcourir les éléments de la liste et comparer les éléments consécutifs
- Echanger les éléments qui ne sont pas en ordre (le plus grand élément prend sa place définitive)
- Recommencer avec la liste restante (sans le dernier élément) jusqu'à épuisement

1, 8, 4, 15, 17, 10

1, 8, 4, 15, 17, 10

1, 8, 4, 15, 17, 10

1, 4, 8, 15, 17, 10

1, 4, 8, 15, 17, 10

1, 4, 8, 15, 17, 10

1, 4, 8, 15, 17, 10

1, 4, 8, 15, 10, 17

Principe

- Parcourir les éléments de la liste et comparer les éléments consécutifs
- Echanger les éléments qui ne sont pas en ordre (le plus grand élément prend sa place définitive)
- Recommencer avec la liste restante (sans le dernier élément) jusqu'à épuisement

1, 8, 4, 15, 17, 10

1, 8, 4, 15, 17, 10

1, 8, 4, 15, 17, 10

1, 4, 8, 15, 17, 10

1, 4, 8, 15, 17, 10

1, 4, 8, 15, 17, 10

1, 4, 8, 15, 17, 10

1, 4, 8, 15, 10, 17

1, 4, 8, 15, 10, 17

Tri à bulles - Algorithme

```
fonction triBulles
entrée t : tableau d'entiers
l:=taille(t)
pour pfin allant de l à 2 pas -1 faire
    pour i allant de 1 à (pfin-1) faire
        si t[i]>t[i+1] alors
            permute(t,i,i+1)
```

Principe

- Supposer que les i premiers éléments de la liste sont triés (une liste de longueur 1 au départ)
- Insérer le nouveau élément de telle sorte que les $i+1$ premiers éléments de la liste soient en ordre

1, 8, 4, 2, 17, 10

Principe

- Supposer que les i premiers éléments de la liste sont triés (une liste de longueur 1 au départ)
- Insérer le nouveau élément de telle sorte que les $i+1$ premiers éléments de la liste soient en ordre

1, 8, 4, 2, 17, 10
1, 8, 4, 2, 17, 10

Principe

- Supposer que les i premiers éléments de la liste sont triés (une liste de longueur 1 au départ)
- Insérer le nouveau élément de telle sorte que les $i+1$ premiers éléments de la liste soient en ordre

1, 8, 4, 2, 17, 10

1, 8, 4, 2, 17, 10

1, 8, 4, 2, 17, 10

Principe

- Supposer que les i premiers éléments de la liste sont triés (une liste de longueur 1 au départ)
- Insérer le nouveau élément de telle sorte que les $i+1$ premiers éléments de la liste soient en ordre

1, 8, 4, 2, 17, 10

1, 8, 4, 2, 17, 10

1, 8, 4, 2, 17, 10

1, 8, 4, 2, 17, 10

Principe

- Supposer que les i premiers éléments de la liste sont triés (une liste de longueur 1 au départ)
- Insérer le nouveau élément de telle sorte que les $i+1$ premiers éléments de la liste soient en ordre

1, 8, 4, 2, 17, 10

1, 8, 4, 2, 17, 10

1, 8, 4, 2, 17, 10

1, 8, 4, 2, 17, 10

1, 8, 4, 2, 17, 10

Principe

- Supposer que les i premiers éléments de la liste sont triés (une liste de longueur 1 au départ)
- Insérer le nouveau élément de telle sorte que les $i+1$ premiers éléments de la liste soient en ordre

1, 8, 4, 2, 17, 10

1, 8, 4, 2, 17, 10

1, 8, 4, 2, 17, 10

1, 8, 4, 2, 17, 10

1, 8, 4, 2, 17, 10

1, 4, 8, 2, 17, 10

Principe

- Supposer que les i premiers éléments de la liste sont triés (une liste de longueur 1 au départ)
- Insérer le nouveau élément de telle sorte que les $i+1$ premiers éléments de la liste soient en ordre

1, 8, 4, 2, 17, 10

1, 8, 4, 2, 17, 10

1, 8, 4, 2, 17, 10

1, 8, 4, 2, 17, 10

1, 8, 4, 2, 17, 10

1, 4, 8, 2, 17, 10

1, 4, 8, 2, 17, 10

Principe

- Supposer que les i premiers éléments de la liste sont triés (une liste de longueur 1 au départ)
- Insérer le nouveau élément de telle sorte que les $i+1$ premiers éléments de la liste soient en ordre

1, 8, 4, 2, 17, 10
1, 8, 4, 2, 17, 10
1, 8, 4, 2, 17, 10
1, 8, 4, 2, 17, 10
1, 8, 4, 2, 17, 10
1, 4, 8, 2, 17, 10
1, 4, 8, 2, 17, 10
1, 4, 8, 2, 17, 10

Principe

- Supposer que les i premiers éléments de la liste sont triés (une liste de longueur 1 au départ)
- Insérer le nouveau élément de telle sorte que les $i+1$ premiers éléments de la liste soient en ordre

1, 8, 4, 2, 17, 10
1, 8, 4, 2, 17, 10
1, 8, 4, 2, 17, 10
1, 8, 4, 2, 17, 10
1, 8, 4, 2, 17, 10
1, 4, 8, 2, 17, 10
1, 4, 8, 2, 17, 10
1, 4, 8, 2, 17, 10
1, 2, 4, 8, 17, 10

Tri par insertion - Algorithme

```
fonction triParInsertion
entrée t : tableau d'entiers
l:=taille(t)
pour i allant de 1 à l faire
    ins := t[i]
    j := i-1;
    while j>=1 and ins<t[j]:
        t[j+1] := t[j]
        j := j-1
    t[j+1] := ins
```

Principe

- Compter le nombre d'occurrences de chaque élément de la liste (domaine des éléments de la liste)
- Déduire l'indice de la première occurrence de chaque élément
- Placer les éléments dans la liste suivant les indices déterminés

1, 2, 1, 1, 3, 5, 7, 3 ,2

Principe

- Compter le nombre d'occurrences de chaque élément de la liste (domaine des éléments de la liste)
- Déduire l'indice de la première occurrence de chaque élément
- Placer les éléments dans la liste suivant les indices déterminés

1, 2, 1, 1, 3, 5, 7, 3 ,2
1, 2, 1, 1, 3, 5, 7, 3 ,2

Principe

- Compter le nombre d'occurrences de chaque élément de la liste (domaine des éléments de la liste)
- Déduire l'indice de la première occurrence de chaque élément
- Placer les éléments dans la liste suivant les indices déterminés

1, 2, 1, 1, 3, 5, 7, 3 ,2

1, 2, 1, 1, 3, 5, 7, 3 ,2

Occurrences 0(0), 1(3), 2(2), 3(2), 4(0), 5(1), 6(0), 7(1)

Principe

- Compter le nombre d'occurrences de chaque élément de la liste (domaine des éléments de la liste)
- Déduire l'indice de la première occurrence de chaque élément
- Placer les éléments dans la liste suivant les indices déterminés

1, 2, 1, 1, 3, 5, 7, 3 ,2

1, 2, 1, 1, 3, 5, 7, 3 ,2

Occurrences 0(0), 1(3), 2(2), 3(2), 4(0), 5(1), 6(0), 7(1)

1, 1, 1, x, x, x, x, x

Principe

- Compter le nombre d'occurrences de chaque élément de la liste (domaine des éléments de la liste)
- Déduire l'indice de la première occurrence de chaque élément
- Placer les éléments dans la liste suivant les indices déterminés

1, 2, 1, 1, 3, 5, 7, 3 ,2

1, 2, 1, 1, 3, 5, 7, 3 ,2

Occurrences 0(0), 1(3), 2(2), 3(2), 4(0), 5(1), 6(0), 7(1)

1, 1, 1, x, x, x, x, x

1, 1, 1, 2, 2, x, x, x, x

Principe

- Compter le nombre d'occurrences de chaque élément de la liste (domaine des éléments de la liste)
- Déduire l'indice de la première occurrence de chaque élément
- Placer les éléments dans la liste suivant les indices déterminés

1, 2, 1, 1, 3, 5, 7, 3 ,2

1, 2, 1, 1, 3, 5, 7, 3 ,2

Occurrences 0(0), 1(3), 2(2), 3(2), 4(0), 5(1), 6(0), 7(1)

1, 1, 1, x, x, x, x, x

1, 1, 1, 2, 2, x, x, x, x

1, 1, 1, 2, 2, 3, 3, x, x

Principe

- Compter le nombre d'occurrences de chaque élément de la liste (domaine des éléments de la liste)
- Déduire l'indice de la première occurrence de chaque élément
- Placer les éléments dans la liste suivant les indices déterminés

1, 2, 1, 1, 3, 5, 7, 3 ,2

1, 2, 1, 1, 3, 5, 7, 3 ,2

Occurrences 0(0), 1(3), 2(2), 3(2), 4(0), 5(1), 6(0), 7(1)

1, 1, 1, x, x, x, x, x

1, 1, 1, 2, 2, x, x, x, x

1, 1, 1, 2, 2, 3, 3, x, x

1, 1, 1, 2, 2, 3, 3, 5, x

Principe

- Compter le nombre d'occurrences de chaque élément de la liste (domaine des éléments de la liste)
- Déduire l'indice de la première occurrence de chaque élément
- Placer les éléments dans la liste suivant les indices déterminés

1, 2, 1, 1, 3, 5, 7, 3 ,2

1, 2, 1, 1, 3, 5, 7, 3 ,2

Occurrences 0(0), 1(3), 2(2), 3(2), 4(0), 5(1), 6(0), 7(1)

1, 1, 1, x, x, x, x, x

1, 1, 1, 2, 2, x, x, x, x

1, 1, 1, 2, 2, 3, 3, x, x

1, 1, 1, 2, 2, 3, 3, 5, x

1, 1, 1, 2, 2, 3, 3, 5, 7

Tri par dénombrement - Algorithme

```
fonction triParDénombrement
entrée t : tableau d'entiers
l:=taille(t)
m:=max(t)
# Calculer le nombre d'occurrences de chaque élément du tableau
# Déclaration d'un tableau effectif de taille m et l'intialiser par 0
pour i allant de 1 à l faire
    effectif[t[i]] = effectif[t[i]]+1
# Positionner les éléments dans le tableau trié t1
# Déclaration du tableau t1 de taille l
posIns := 1
pour i allant de 0 à m faire
    pour j allant de 1 à effectif[t[i]] faire
        t1[posIns] := i
        posIns := posIns+1
```

Problème

Tri par dénombrement mais avec un domaine des clés (valeurs de la liste) très large!!

Tri par base - Principe

Problème

Tri par dénombrement mais avec un domaine des clés (valeurs de la liste) très large!!

Solution

Utiliser un tri par dénombrement en plusieurs passes (pour chaque chiffre des clés)

Tri par base - Principe

Problème

Tri par dénombrement mais avec un domaine des clés (valeurs de la liste) très large!!

Solution

Utiliser un tri par dénombrement en plusieurs passes (pour chaque chiffre des clés)

151, 203, 002, 259, 671, 037

Tri par base - Principe

Problème

Tri par dénombrement mais avec un domaine des clés (valeurs de la liste) très large!!

Solution

Utiliser un tri par dénombrement en plusieurs passes (pour chaque chiffre des clés)

151, 203, 002, 259, 671, 037
151, 203, 002, 259, 671, 037

Tri par base - Principe

Problème

Tri par dénombrement mais avec un domaine des clés (valeurs de la liste) très large!!

Solution

Utiliser un tri par dénombrement en plusieurs passes (pour chaque chiffre des clés)

151, 203, 002, 259, 671, 037

151, 203, 002, 259, 671, 037

151, 671, 002, 203, 037, 259

Tri par base - Principe

Problème

Tri par dénombrement mais avec un domaine des clés (valeurs de la liste) très large!!

Solution

Utiliser un tri par dénombrement en plusieurs passes (pour chaque chiffre des clés)

151, 203, 002, 259, 671, 037

151, 203, 002, 259, 671, 037

151, 671, 002, 203, 037, 259

151, 671, 002, 203, 037, 259

Tri par base - Principe

Problème

Tri par dénombrement mais avec un domaine des clés (valeurs de la liste) très large!!

Solution

Utiliser un tri par dénombrement en plusieurs passes (pour chaque chiffre des clés)

151, 203, 002, 259, 671, 037
151, 203, 002, 259, 671, 037
151, 671, 002, 203, 037, 259
151, 671, 002, 203, 037, 259
002, 203, 037, 151, 259, 671

Tri par base - Principe

Problème

Tri par dénombrement mais avec un domaine des clés (valeurs de la liste) très large!!

Solution

Utiliser un tri par dénombrement en plusieurs passes (pour chaque chiffre des clés)

151, 203, 002, 259, 671, 037
151, 203, 002, 259, 671, 037
151, 671, 002, 203, 037, 259
151, 671, 002, 203, 037, 259
002, 203, 037, 151, 259, 671
002, 203, 037, 151, 259, 671

Tri par base - Principe

Problème

Tri par dénombrement mais avec un domaine des clés (valeurs de la liste) très large!!

Solution

Utiliser un tri par dénombrement en plusieurs passes (pour chaque chiffre des clés)

151, 203, 002, 259, 671, 037
151, 203, 002, 259, 671, 037
151, 671, 002, 203, 037, 259
151, 671, 002, 203, 037, 259
002, 203, 037, 151, 259, 671
002, 203, 037, 151, 259, 671
002, 037, 151, 203, 259, 671

Tri par base - Principe

Problème

Tri par dénombrement mais avec un domaine des clés (valeurs de la liste) très large!!

Solution

Utiliser un tri par dénombrement en plusieurs passes (pour chaque chiffre des clés)

151, 203, 002, 259, 671, 037

151, 203, 002, 259, 671, 037

151, 671, 002, 203, 037, 259

151, 671, 002, 203, 037, 259

002, 203, 037, 151, 259, 671

002, 203, 037, 151, 259, 671

002, 037, 151, 203, 259, 671

002, 037, 151, 203, 259, 671

Tri par base - Algorithme

```
fonction triParBase  
entrée t : tableau d'entiers  
m=nombreChiffreMax(t)  
pour posC allant de 1 à m faire  
    triStable(t, posC)
```

Tri stable

Un algorithme de tri qui conserve l'ordre relatif des éléments égaux (ex. tri par insertion, tri à bulle)

- Tris itératifs
 - Tri par sélection
 - Tri à bulles
 - Tri par insertion
 - Tri par dénombrement
 - Tri par base
- Tris récursifs
 - Tri fusion
 - Tri rapide
- Tris optimisés

Principe

- **Deviser** : deviser les données initiales en plusieurs sous-parties
- **Régner** : résoudre récursivement chaque sous-partie du problème
- **Combiner** : combiner les solutions des sous-problèmes pour obtenir la solution du problème initial

Principe

- Scinder la liste à trier en deux listes de longueurs identiques
- Trier chaque liste à part
- Fusionner les deux listes triées

4, 18, 6, 7, 15, 9, 14, 1, 3, 10, 2

Principe

- Scinder la liste à trier en deux listes de longueurs identiques
- Trier chaque liste à part
- Fusionner les deux listes triées

4, 18, 6, 7, 15, 9, 14, 1, 3, 10, 2
4, 18, 6, 7, 15 9, 14, 1, 3, 10

Principe

- Scinder la liste à trier en deux listes de longueurs identiques
- Trier chaque liste à part
- Fusionner les deux listes triées

4, 18, 6, 7, 15, 9, 14, 1, 3, 10, 2
4, 18, 6, 7, 15 9, 14, 1, 3, 10
4, 6, 7, 15, 18 **1, 3, 9, 10, 14**
1

Principe

- Scinder la liste à trier en deux listes de longueurs identiques
- Trier chaque liste à part
- Fusionner les deux listes triées

4, 18, 6, 7, 15, 9, 14, 1, 3, 10, 2
4, 18, 6, 7, 15 9, 14, 1, 3, 10
4, 6, 7, 15, 18 **1, 3, 9, 10, 14**
1, 3

Principe

- Scinder la liste à trier en deux listes de longueurs identiques
- Trier chaque liste à part
- Fusionner les deux listes triées

4, 18, 6, 7, 15, 9, 14, 1, 3, 10, 2
4, 18, 6, 7, 15 9, 14, 1, 3, 10
4, 6, 7, 15, 18 1, 3, **9**, 10, 14
 1, 3, 4

Principe

- Scinder la liste à trier en deux listes de longueurs identiques
- Trier chaque liste à part
- Fusionner les deux listes triées

4, 18, 6, 7, 15, 9, 14, 1, 3, 10, 2
4, 18, 6, 7, 15 9, 14, 1, 3, 10
4, 6, 7, 15, 18 1, 3, 9, 10, 14
1, 3, 4, 6

Principe

- Scinder la liste à trier en deux listes de longueurs identiques
- Trier chaque liste à part
- Fusionner les deux listes triées

4, 18, 6, 7, 15, 9, 14, 1, 3, 10, 2
4, 18, 6, 7, 15 9, 14, 1, 3, 10
4, 6, 7, 15, 18 1, 3, 9, 10, 14
1, 3, 4, 6, 7, 9, 10, 14, 15, 18

Principe

- Scinder la liste à trier en deux listes de longueurs identiques
- Trier chaque liste à part
- Fusionner les deux listes triées

4, 18, 6, 7, 15

Principe

- Scinder la liste à trier en deux listes de longueurs identiques
- Trier chaque liste à part
- Fusionner les deux listes triées

4, 18, 6, 7, 15
4, 18 6, 7, 15

Principe

- Scinder la liste à trier en deux listes de longueurs identiques
- Trier chaque liste à part
- Fusionner les deux listes triées

4, 18, 6, 7, 15
4, 18 6, 7, 15
4 18 6 7, 15

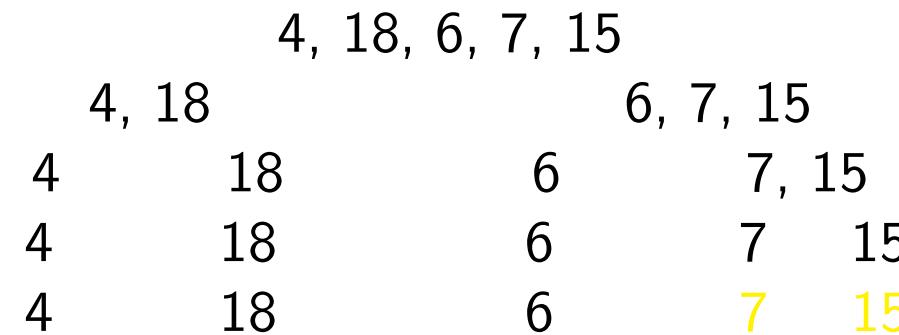
Principe

- Scinder la liste à trier en deux listes de longueurs identiques
- Trier chaque liste à part
- Fusionner les deux listes triées

		4, 18, 6, 7, 15		
	4, 18	6, 7, 15		
4	18	6	7, 15	
4	18	6	7	15

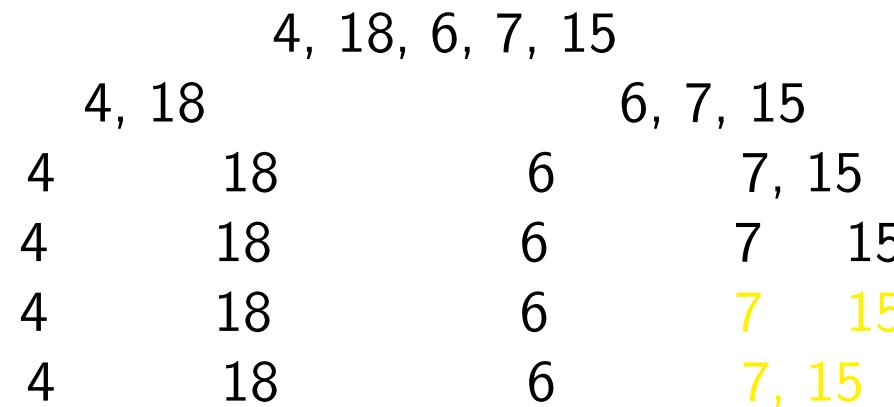
Principe

- Scinder la liste à trier en deux listes de longueurs identiques
- Trier chaque liste à part
- Fusionner les deux listes triées



Principe

- Scinder la liste à trier en deux listes de longueurs identiques
- Trier chaque liste à part
- Fusionner les deux listes triées



Principe

- Scinder la liste à trier en deux listes de longueurs identiques
- Trier chaque liste à part
- Fusionner les deux listes triées

4, 18, 6, 7, 15			
4, 18		6, 7, 15	
4	18	6	7, 15
4	18	6	7 15
4	18	6	7 15
4	18	6	7, 15
4	18	6	7, 15

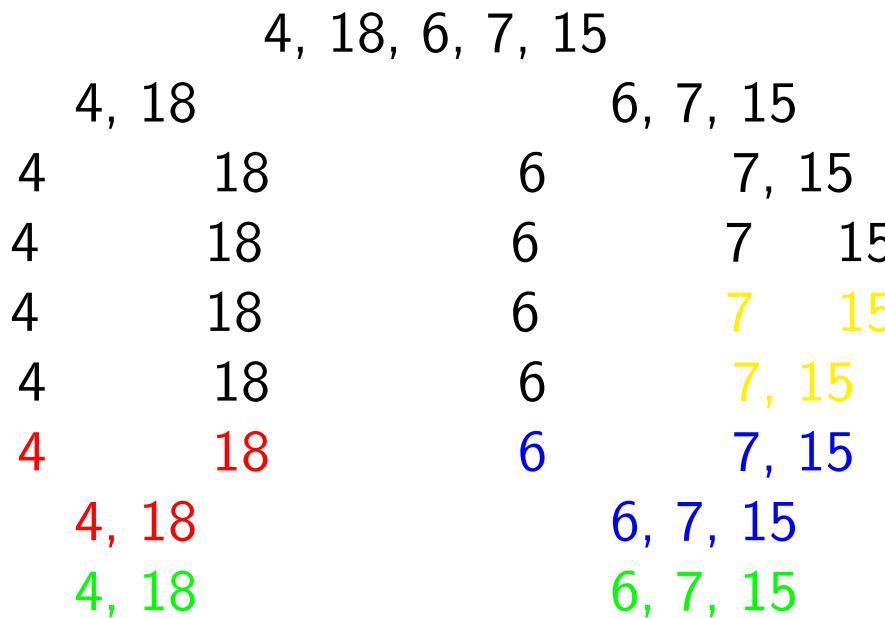
Principe

- Scinder la liste à trier en deux listes de longueurs identiques
- Trier chaque liste à part
- Fusionner les deux listes triées

		4, 18, 6, 7, 15
	4, 18	6, 7, 15
4	18	6
4	18	7, 15
4	18	6
4	18	7
4	18	15
4	18	6
4	18	7, 15
4	18	6
4	18	7, 15
4, 18		6, 7, 15

Principe

- Scinder la liste à trier en deux listes de longueurs identiques
- Trier chaque liste à part
- Fusionner les deux listes triées



Principe

- Scinder la liste à trier en deux listes de longueurs identiques
- Trier chaque liste à part
- Fusionner les deux listes triées

4, 18, 6, 7, 15

4, 18	6, 7, 15
4 18	6 7, 15
4 18	6 7 15
4 18	6 7 15
4 18	6 7, 15
4 18	6 7, 15
4, 18	6, 7, 15
4, 18	6, 7, 15
4, 6, 7, 15, 18	

```
fonction triFusion
entrée t : tableau d'entiers
n:=taille(t)
si n<=1 alors
    retourne t
sinon
    retourne fusion(triFusion(t[1, ..., n/2]),
                    triFusion(t[n/2 + 1, ..., n]))
```

Principe

- Choisir un élément pivot
- Partitionner la liste afin que tous les éléments inférieurs au pivot se trouvent avant lui
- Trier récursivement les deux partitions de la liste (éléments inférieurs au pivot et ceux supérieurs à lui)

4, 18, 6, 7, 15, 9, 14, 1, 3, 10, 2

Principe

- Choisir un élément pivot
- Partitionner la liste afin que tous les éléments inférieurs au pivot se trouvent avant lui
- Trier récursivement les deux partitions de la liste (éléments inférieurs au pivot et ceux supérieurs à lui)

4, 18, 6, 7, 15, 9, 14, 1, 3, 10, 2
4, 18, 6, 7, 15, 9, 14, 1, 3, 10, 2

Principe

- Choisir un élément pivot
- Partitionner la liste afin que tous les éléments inférieurs au pivot se trouvent avant lui
- Trier récursivement les deux partitions de la liste (éléments inférieurs au pivot et ceux supérieurs à lui)

4, 18, 6, 7, 15, 9, 14, 1, 3, 10, 2
4, 18, 6, 7, 15, 9, 14, 1, 3, 10, 2
1, 3, 2 4 18, 6, 7, 9, 14, 10

Principe

- Choisir un élément pivot
- Partitionner la liste afin que tous les éléments inférieurs au pivot se trouvent avant lui
- Trier récursivement les deux partitions de la liste (éléments inférieurs au pivot et ceux supérieurs à lui)

4, 18, 6, 7, 15, 9, 14, 1, 3, 10, 2

4, 18, 6, 7, 15, 9, 14, 1, 3, 10, 2

1, 3, 2 4 18, 6, 7, 9, 14, 10

1, 3, 2 4 18, 6, 7, 9, 14, 10

Principe

- Choisir un élément pivot
- Partitionner la liste afin que tous les éléments inférieurs au pivot se trouvent avant lui
- Trier récursivement les deux partitions de la liste (éléments inférieurs au pivot et ceux supérieurs à lui)

4, 18, 6, 7, 15, 9, 14, 1, 3, 10, 2				
4, 18, 6, 7, 15, 9, 14, 1, 3, 10, 2				
1, 3, 2	4	18, 6, 7, 9, 14, 10		
1, 3, 2	4	18, 6, 7, 9, 14, 10		
1	3, 2	4	6, 7, 9, 14, 10	18

Principe

- Choisir un élément pivot
- Partitionner la liste afin que tous les éléments inférieurs au pivot se trouvent avant lui
- Trier récursivement les deux partitions de la liste (éléments inférieurs au pivot et ceux supérieurs à lui)

4, 18, 6, 7, 15, 9, 14, 1, 3, 10, 2				
4, 18, 6, 7, 15, 9, 14, 1, 3, 10, 2				
1, 3, 2	4	18, 6, 7, 9, 14, 10		
1, 3, 2	4	18, 6, 7, 9, 14, 10		
1	3, 2	4	6, 7, 9, 14, 10	18
1	3, 2	4	6, 7, 9, 14, 10	18

Exemple du mécanisme de partitionnement

2, 7, 6, 5, 3, 9, 8, 4, 1

Exemple du mécanisme de partitionnement

2, 7, 6, 5, 3, 9, 8, 4, 1
2, 7, 6, **5**, 3, 9, 8, 4, **1**

Exemple du mécanisme de partitionnement

2, 7, 6, 5, 3, 9, 8, 4, 1

2, 7, 6, **5**, 3, 9, 8, 4, **1**

2, 7, 6, 1, 3, 9, 8, 4, **5**

Exemple du mécanisme de partitionnement

2, 7, 6, 5, 3, 9, 8, 4, 1

2, 7, 6, **5**, 3, 9, 8, 4, **1**

2, 7, 6, 1, 3, 9, 8, 4, **5**

2, 7, 2, 1, 3, 9, 8, 4, **5**

Exemple du mécanisme de partitionnement

2, 7, 6, 5, 3, 9, 8, 4, 1

2, 7, 6, **5**, 3, 9, 8, 4, **1**

2, 7, 6, 1, 3, 9, 8, 4, **5**

2, 7, 2, 1, 3, 9, 8, 4, **5**

2, **7**, 6, 1, 3, 9, 8, 4, **5**

Exemple du mécanisme de partitionnement

2, 7, 6, 5, 3, 9, 8, 4, 1
2, 7, 6, 5, 3, 9, 8, 4, 1
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 7, 2, 1, 3, 9, 8, 4, 5
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 7, 6, 1, 3, 9, 8, 4, 5

Exemple du mécanisme de partitionnement

2, 7, 6, 5, 3, 9, 8, 4, 1

2, 7, 6, 5, 3, 9, 8, 4, 1

2, 7, 6, 1, 3, 9, 8, 4, 5

2, 7, 2, 1, 3, 9, 8, 4, 5

2, 7, 6, 1, 3, 9, 8, 4, 5

2, 7, 6, 1, 3, 9, 8, 4, 5

2, 7, 6, 1, 3, 9, 8, 4, 5

Exemple du mécanisme de partitionnement

2, 7, 6, 5, 3, 9, 8, 4, 1
2, 7, 6, 5, 3, 9, 8, 4, 1
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 7, 2, 1, 3, 9, 8, 4, 5
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 1, 6, 7, 3, 9, 8, 4, 5

Exemple du mécanisme de partitionnement

2, 7, 6, 5, 3, 9, 8, 4, 1

2, 7, 6, 5, 3, 9, 8, 4, 1

2, 7, 6, 1, 3, 9, 8, 4, 5

2, 7, 2, 1, 3, 9, 8, 4, 5

2, 7, 6, 1, 3, 9, 8, 4, 5

2, 7, 6, 1, 3, 9, 8, 4, 5

2, 7, 6, 1, 3, 9, 8, 4, 5

2, 1, 6, 7, 3, 9, 8, 4, 5

2, 1, 6, 7, 3, 9, 8, 4, 5

Exemple du mécanisme de partitionnement

2, 7, 6, 5, 3, 9, 8, 4, 1
2, 7, 6, 5, 3, 9, 8, 4, 1
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 7, 2, 1, 3, 9, 8, 4, 5
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 1, 6, 7, 3, 9, 8, 4, 5
2, 1, 6, 7, 3, 9, 8, 4, 5
2, 1, 3, 7, 6, 9, 8, 4, 5

Exemple du mécanisme de partitionnement

2, 7, 6, 5, 3, 9, 8, 4, 1
2, 7, 6, 5, 3, 9, 8, 4, 1
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 7, 2, 1, 3, 9, 8, 4, 5
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 1, 6, 7, 3, 9, 8, 4, 5
2, 1, 6, 7, 3, 9, 8, 4, 5
2, 1, 3, 7, 6, 9, 8, 4, 5
2, 1, 3, 7, 6, 9, 8, 4, 5

Exemple du mécanisme de partitionnement

2, 7, 6, 5, 3, 9, 8, 4, 1

2, 7, 6, 5, 3, 9, 8, 4, 1

2, 7, 6, 1, 3, 9, 8, 4, 5

2, 7, 2, 1, 3, 9, 8, 4, 5

2, 7, 6, 1, 3, 9, 8, 4, 5

2, 7, 6, 1, 3, 9, 8, 4, 5

2, 7, 6, 1, 3, 9, 8, 4, 5

2, 1, 6, 7, 3, 9, 8, 4, 5

2, 1, 6, 7, 3, 9, 8, 4, 5

2, 1, 3, 7, 6, 9, 8, 4, 5

2, 1, 3, 7, 6, 9, 8, 4, 5

2, 1, 3, 7, 6, 9, 8, 4, 5

Exemple du mécanisme de partitionnement

2, 7, 6, 5, 3, 9, 8, 4, 1
2, 7, 6, 5, 3, 9, 8, 4, 1
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 7, 2, 1, 3, 9, 8, 4, 5
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 1, 6, 7, 3, 9, 8, 4, 5
2, 1, 6, 7, 3, 9, 8, 4, 5
2, 1, 3, 7, 6, 9, 8, 4, 5
2, 1, 3, 7, 6, 9, 8, 4, 5
2, 1, 3, 7, 6, 9, 8, 4, 5

Exemple du mécanisme de partitionnement

2, 7, 6, 5, 3, 9, 8, 4, 1
2, 7, 6, 5, 3, 9, 8, 4, 1
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 7, 2, 1, 3, 9, 8, 4, 5
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 1, 6, 7, 3, 9, 8, 4, 5
2, 1, 6, 7, 3, 9, 8, 4, 5
2, 1, 3, 7, 6, 9, 8, 4, 5
2, 1, 3, 7, 6, 9, 8, 4, 5
2, 1, 3, 7, 6, 9, 8, 4, 5
2, 1, 3, 4, 6, 9, 8, 7, 5

Exemple du mécanisme de partitionnement

2, 7, 6, 5, 3, 9, 8, 4, 1
2, 7, 6, 5, 3, 9, 8, 4, 1
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 7, 2, 1, 3, 9, 8, 4, 5
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 1, 6, 7, 3, 9, 8, 4, 5
2, 1, 6, 7, 3, 9, 8, 4, 5
2, 1, 3, 7, 6, 9, 8, 4, 5
2, 1, 3, 7, 6, 9, 8, 4, 5
2, 1, 3, 7, 6, 9, 8, 4, 5
2, 1, 3, 4, 6, 9, 8, 7, 5
2, 1, 3, 4, 5, 9, 8, 7, 6

Exemple du mécanisme de partitionnement

2, 7, 6, 5, 3, 9, 8, 4, 1
2, 7, 6, 5, 3, 9, 8, 4, 1
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 7, 2, 1, 3, 9, 8, 4, 5
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 7, 6, 1, 3, 9, 8, 4, 5
2, 1, 6, 7, 3, 9, 8, 4, 5
2, 1, 6, 7, 3, 9, 8, 4, 5
2, 1, 3, 7, 6, 9, 8, 4, 5
2, 1, 3, 7, 6, 9, 8, 4, 5
2, 1, 3, 7, 6, 9, 8, 4, 5
2, 1, 3, 4, 6, 9, 8, 7, 5
2, 1, 3, 4, 5, 9, 8, 7, 6
2, 1, 3, 4, 5, 9, 8, 7, 6

```
fonction tri_rapide
entrée t : tableau d'entiers
tri_rapide_rec(t,1,taille(t))

fonction tri_rapide_rec
entrée t : tableau d'entiers , premier : entier ,
dernier : entier
si premier < dernier alors
    pivot=choix_pivot(t,premier,dernier)
    pos_pivot=partitionner(t,premier,dernier,pivot)
    tri_rapide_rec(t,premier,pos_pivot-1)
    tri_rapide_rec(t,pos_pivot+1,dernier)
```

Performances des algorithmes de tri

- Tri rapide est l'un des algorithmes de tri les plus rapides (choix du pivot!!!)
- Tri fusion est un bon tri stable pour des listes de grande taille
- Tri par insertion est le plus efficace pour des listes de petite taille

Optimisation

- Combiner le tri fusion et le tri par insertion
- Combiner le tri rapide et le tri par insertion