Ch. 4 : Algorithmes de tris

Introduction

Problème (le plus ?) fondamental en algorithmique

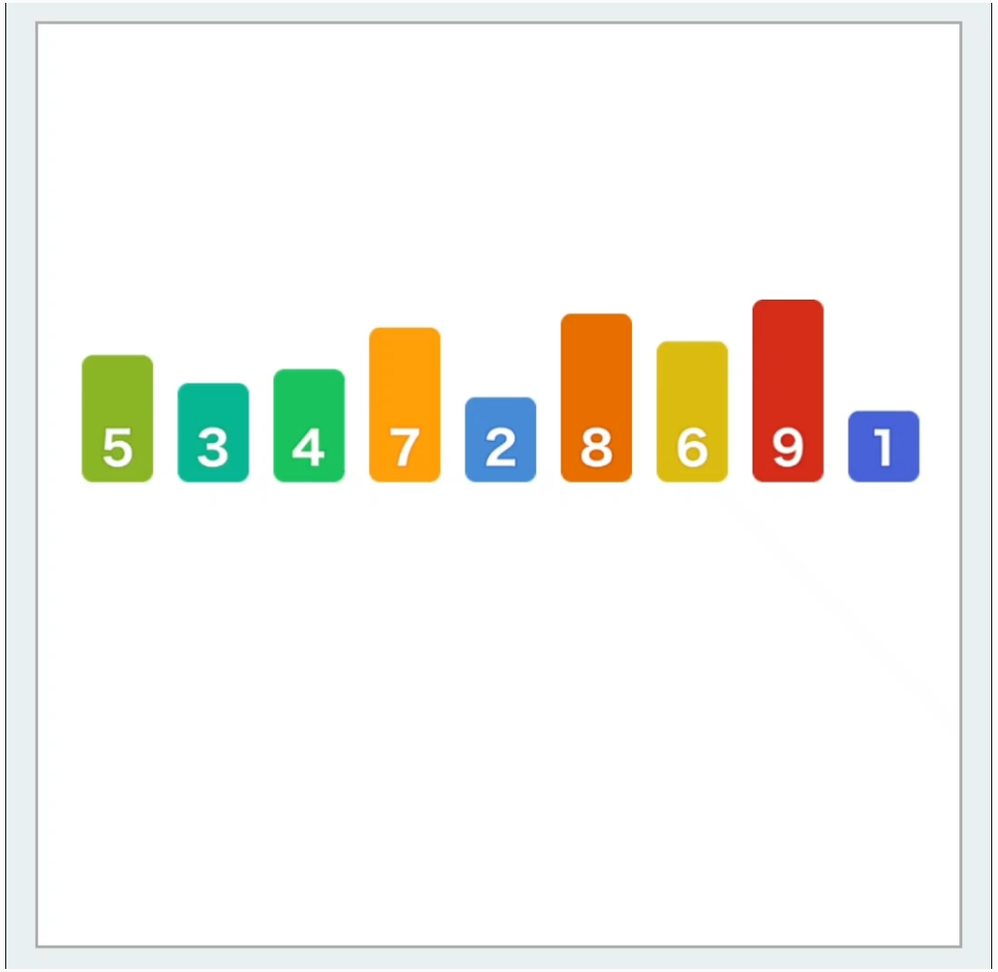
- nombreuses applications pratiques (trier par prix, par intérêt, par ordre alphabétique…)
- de nombreux algorithmes reposent sur le fait que les objets qu'ils traitent sont triés (algorithmes de rendu de scène 3D, parcours de Graham pour calculer l'enveloppe convexe d'un ensemble de points…)
- la recherche d'informations est beaucoup plus rapide dans un ensemble trié
- on peut utiliser les bornes inférieures connues sur les algorithmes de tri pour prouver des bornes inférieures pour d'autres problèmes algorithmiques

3 Tri par insertion

Tri du « jeu de cartes »

- Peu efficace sur les instances de grande taille
- Mais très efficace sur les petites instances, ou si les données sont déjà presque triées

⇒ souvent utilisé conjointement avec un algorithme plus rapide



Particularités :

- Tri « stable » : deux éléments égaux sont classés dans le même ordre que dans le tableau original
- Tri « en place » : n'utilise pas de tableau auxiliaire

Tri par sélection (ou par extraction)

Principe : à chaque itération, on cherche et on range à sa place le i-ème plus petit élément

Particularités :

Tri « en place »

! Tri pas forcément stable

Tri à bulles (ou par propagation)

Nommé ainsi car les éléments « remontent à leur place comme des bulles de champagne »

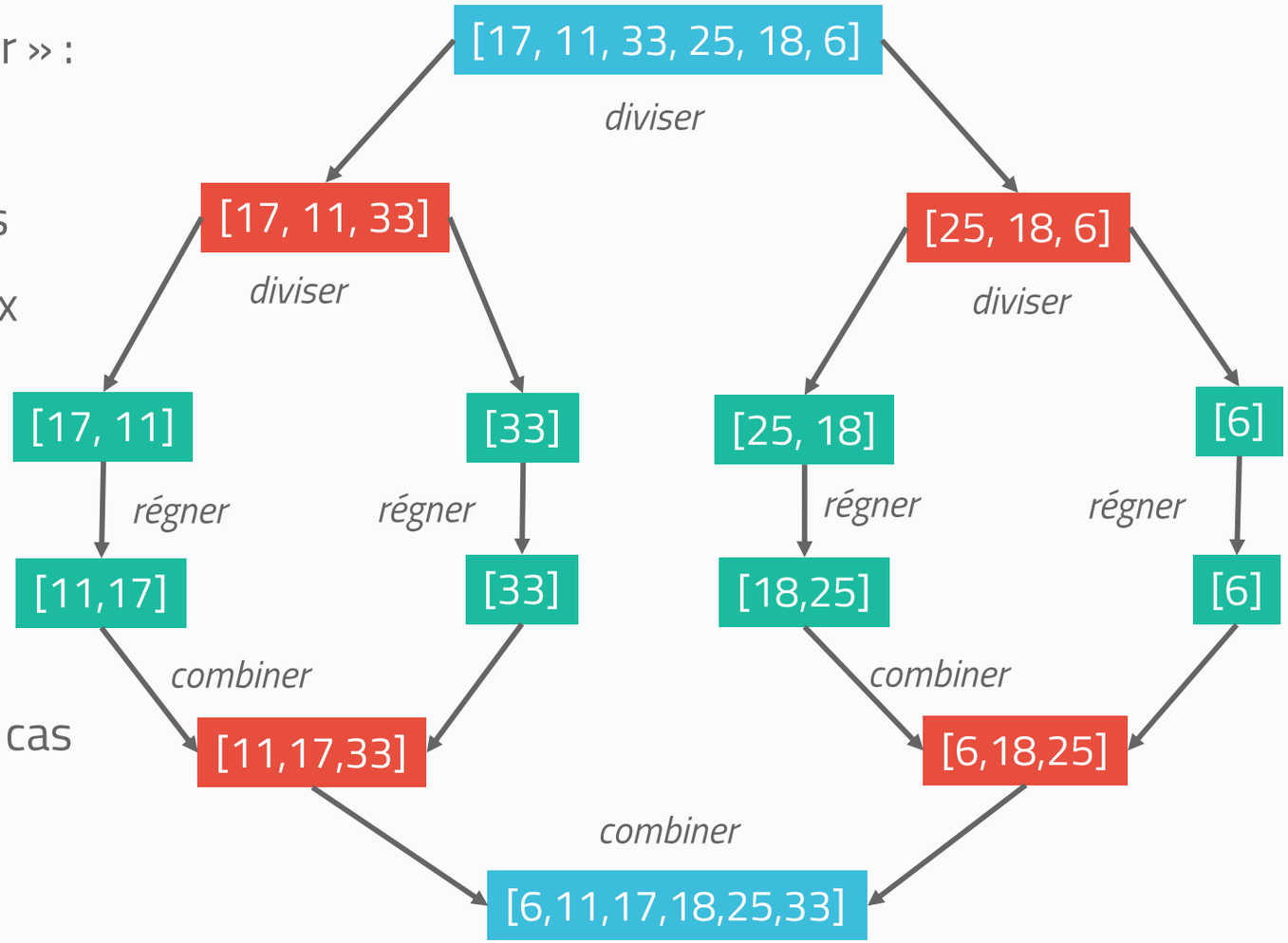
Particularités :

Très simple… mais pas très efficace

Tri fusion

Application du principe « Diviser pour régner » :

- *diviser* récursivement en 2
- *régner* = trier un tableau à 1 ou 2 entrées
- *combiner* = fusionner deux sous-tableaux déjà triés (comment faire ?)



Complexité optimale, même dans le pire cas

Tri rapide

- On choisit un des éléments du tableau comme *pivot*
- A l'aide de deux pointeurs *L* et *R*, on place à gauche tous les éléments plus petits que le pivot, et à droite tous les éléments plus grands que le pivot
- Les deux pointeurs *L* et *R* finissent par se rencontrer sur un élément, qu'on permute avec le pivot

  ⇒ le pivot est à sa place définitive
- On recommence, récursivement, avec la partie à gauche du pivot, puis avec la partie à droite du pivot

A number is chosen as a reference for sorting. This number is called the "pivot".

💡 Algorithme très rapide en pratique, bien que moins bon que le tri fusion en théorie sur les « pires cas »

Synthèse

Algorithme	Complexité en temps en moyenne	Complexité en temps dans le pire cas
Tri par insertion	$O(n^2)$	$O(n^2)$
Tri par sélection	$O(n^2)$	$O(n^2)$
Tri à bulles	$O(n^2)$	$O(n^2)$
Tri fusion	$O(n \log n)$	$O(n \log n)$
Tri rapide	$O(n \log n)$	$O(n^2)$

On peut démontrer que tout tri basé sur des comparaisons est en $\Omega(n \log n)$

Peut-on trier plus rapidement, avec d'autres hypothèses ?

Tri par comptage (ou tri casier)

On dispose d'un tableau de *n* nombres dont on sait qu'il sont compris entre 0 et *k*

Pour trier ce tableau, il suffit de compter le nombre de 0, de 1, de 2..., de k

Exemple : on dispose d'un tableau de nombres compris entre 0 et 15 :

11	3	8	15	3	5	11	2	1	9

On construit l'histogramme des valeurs :

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	2	0	1	0	0	1	1	0	2	0	0	0	1

Et il suffit de parcourir cet histogramme pour obtenir le tableau trié

1	2	3	3	5	8	9	11	11	15

Quelle est la complexité de cette algorithme ?