C4 Diviser pour régner

Notion de complexité d'un algorithme

Lorsqu'on s'intéresse aux performances d'un algorithme, on fait varier le volume de données traité par l'algorithme et on étudie :

C4 Diviser pour régner

Notion de complexité d'un algorithme

Lorsqu'on s'intéresse aux performances d'un algorithme, on fait varier le volume de données traité par l'algorithme et on étudie :

- l'évolution du nombre d'opération nécessaires au fonctionnement de l'algorithme, c'est ce qu'on appelle la la compléxité en temps de l'algorithme.

C4 Diviser pour régner

Notion de complexité d'un algorithme

Lorsqu'on s'intéresse aux performances d'un algorithme, on fait varier le volume de données traité par l'algorithme et on étudie :

- l'évolution du nombre d'opération nécessaires au fonctionnement de l'algorithme, c'est ce qu'on appelle la la compléxité en temps de l'algorithme.
- l'évolution de l'espace mémoire nécessaire au fonctionnement de l'algorithme, c'est ce qu'on appelle la la compléxité spatiale de l'algorithme.

C4 Diviser pour régner

Complexité linéaire

C4 Diviser pour régner

Complexité linéaire

- On dira qu'un algorithme a une complexité en temps linéaire lorsque qu'un multiplication de la taille des données par un facteur k se traduit par une augmentation du temps de calcul par un facteur proche de k.

C4 Diviser pour régner

Complexité linéaire

- On dira qu'un algorithme a une complexité en temps linéaire lorsque qu'un multiplication de la taille des données par un facteur k se traduit par une augmentation du temps de calcul par un facteur proche de k.
- Par exemple si la complexité est linéaire traiter une liste 10 fois plus grande prendra environ 10 fois plus de temps

C4 Diviser pour régner

Complexité linéaire

- On dira qu'un algorithme a une complexité en temps linéaire lorsque qu'un multiplication de la taille des données par un facteur k se traduit par une augmentation du temps de calcul par un facteur proche de k.
- Par exemple si la complexité est linéaire traiter une liste 10 fois plus grande prendra environ 10 fois plus de temps
- Dans ce cas lorsqu'on trace le graphique du temps de calcul en fonction de la taille des données on obtient une droite.

C4 Diviser pour régner

Complexité linéaire

- On dira qu'un algorithme a une complexité en temps linéaire lorsque qu'un multiplication de la taille des données par un facteur k se traduit par une augmentation du temps de calcul par un facteur proche de k.
- Par exemple si la complexité est linéaire traiter une liste 10 fois plus grande prendra environ 10 fois plus de temps
- Dans ce cas lorsqu'on trace le graphique du temps de calcul en fonction de la taille des données on obtient une droite.

Exemple

Un algorithme de parcourt simple d'une liste (par exemple recherche de minimum ou calcul de moyenne) a une complexité linéaire.

C4 Diviser pour régner

Complexité quadratique

C4 Diviser pour régner

Complexité quadratique

- On dira qu'un algorithme a une complexité en temps quadratique lorsque qu'un multiplication de la taille des données par un facteur k se traduit par une augmentation du temps de calcul par un facteur proche de k^2.

C4 Diviser pour régner

Complexité quadratique

- On dira qu'un algorithme a une complexité en temps quadratique lorsque qu'un multiplication de la taille des données par un facteur k se traduit par une augmentation du temps de calcul par un facteur proche de k^2.
- Par exemple si la complexité est quadratique traiter une liste 10 fois plus grande prendra environ 100 fois plus de temps

C4 Diviser pour régner

Complexité quadratique

- On dira qu'un algorithme a une complexité en temps quadratique lorsque qu'un multiplication de la taille des données par un facteur k se traduit par une augmentation du temps de calcul par un facteur proche de k^2.
- Par exemple si la complexité est quadratique traiter une liste 10 fois plus grande prendra environ 100 fois plus de temps
- Dans ce cas lorsqu'on trace le graphique du temps de calcul en fonction de la taille des données on obtient une parabole.

C4 Diviser pour régner

Complexité quadratique

- On dira qu'un algorithme a une complexité en temps quadratique lorsque qu'un multiplication de la taille des données par un facteur k se traduit par une augmentation du temps de calcul par un facteur proche de k^2.
- Par exemple si la complexité est quadratique traiter une liste 10 fois plus grande prendra environ 100 fois plus de temps
- Dans ce cas lorsqu'on trace le graphique du temps de calcul en fonction de la taille des données on obtient une parabole.

Exemple

Les algorithmes de tri par insertion ou par sélection ont une complexité quadratique.

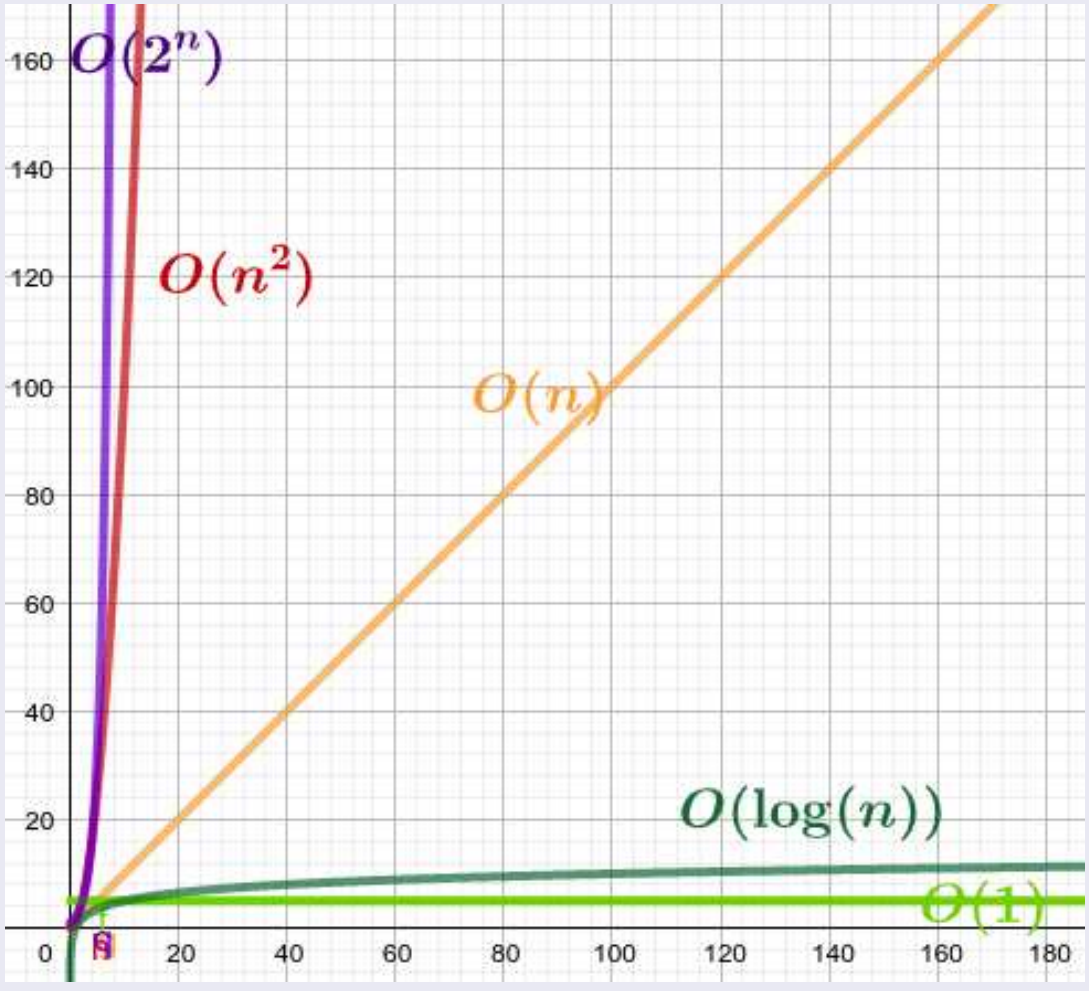
C4 Diviser pour régner

A retenir !

Complexité	Nom	Exemple
$O(1)$	Temps constant	Accéder à un élément d'une liste
$O(\log(n))$	Complexité logarithmique	Recherche dichotomique dans une liste
$O(n)$	Complexité linéaire	Recherche simple dans une liste
$O(n^2)$	Complexité quadratique	Tri par insertion d'une liste
$O(2^n)$	Complexité exponentielle	Algorithme par force brute pour le sac à dos

C4 Diviser pour régner

Représentation graphique



C4 Diviser pour régner

Exemples

- On suppose qu'on dispose d'un algorithme de complexité linéaire travaillant sur une liste, il traite une liste de 1 000 éléments en 0,015 secondes. Donner une estimation du temps de calcul pour une liste de 250 000 éléments.

C4 Diviser pour régner

Exemples

- On suppose qu'on dispose d'un algorithme de complexité linéaire travaillant sur une liste, il traite une liste de 1 000 éléments en 0,015 secondes. Donner une estimation du temps de calcul pour une liste de 250 000 éléments.
  La taille des données a été multiplié par 250, la complexité étant lineaire le temps de calcul sera aussi approximativement multiplié par 250.

C4 Diviser pour régner

Exemples

- On suppose qu'on dispose d'un algorithme de complexité linéaire travaillant sur une liste, il traite une liste de 1 000 éléments en 0,015 secondes. Donner une estimation du temps de calcul pour une liste de 250 000 éléments.
  La taille des données a été multiplié par 250, la complexité étant lineaire le temps de calcul sera aussi approximativement multiplié par 250.
  $0.015 \times 250 = 3.75$, on peut donc prévoir un temps de calcul d'environ 3,75 secondes

C4 Diviser pour régner

Exemples

- On suppose qu'on dispose d'un algorithme de complexité linéaire travaillant sur une liste, il traite une liste de 1 000 éléments en 0,015 secondes. Donner une estimation du temps de calcul pour une liste de 250 000 éléments.
  La taille des données a été multiplié par 250, la complexité étant lineaire le temps de calcul sera aussi approximativement multiplié par 250.
  $0.015 \times 250 = 3.75$, on peut donc prévoir un temps de calcul d'environ 3,75 secondes
- Même question pour un algorithme de complexité quadratique qui traite une liste de 1 000 éléments en 0,07 secondes.

C4 Diviser pour régner

Exemples

- On suppose qu'on dispose d'un algorithme de complexité linéaire travaillant sur une liste, il traite une liste de 1 000 éléments en 0,015 secondes. Donner une estimation du temps de calcul pour une liste de 250 000 éléments.
  La taille des données a été multiplié par 250, la complexité étant lineaire le temps de calcul sera aussi approximativement multiplié par 250.
  $0.015 \times 250 = 3.75$, on peut donc prévoir un temps de calcul d'environ 3,75 secondes
- Même question pour un algorithme de complexité quadratique qui traite une liste de 1 000 éléments en 0,07 secondes.
  La taille des données a été multiplié par 250, la complexité étant quadratique le temps de calcul sera approximativement multiplié par $250^2 = 62500$

C4 Diviser pour régner

Exemples

- On suppose qu'on dispose d'un algorithme de complexité linéaire travaillant sur une liste, il traite une liste de 1 000 éléments en 0,015 secondes. Donner une estimation du temps de calcul pour une liste de 250 000 éléments.
  La taille des données a été multiplié par 250, la complexité étant lineaire le temps de calcul sera aussi approximativement multiplié par 250.
  $0.015 \times 250 = 3.75$, on peut donc prévoir un temps de calcul d'environ 3,75 secondes
- Même question pour un algorithme de complexité quadratique qui traite une liste de 1 000 éléments en 0,07 secondes.
  La taille des données a été multiplié par 250, la complexité étant quadratique le temps de calcul sera approximativement multiplié par $250^2 = 62500$
  $0.07 \times 62\,500 = 4375$, on peut donc prévoir un temps de calcul d'environ $4\,375$ secondes, c'est à dire près d'une heure et 15 minutes !

C4 Diviser pour régner

Principe de la méthode

La méthode diviser pour régner (en anglais *divide and conquer*) est une technique algorithmique qui consiste à :

C4 Diviser pour régner

Principe de la méthode

La méthode diviser pour régner (en anglais *divide and conquer*) est une technique algorithmique qui consiste à :

1. décomposer le problème initial en un ou plusieurs sous problèmes de taille inférieure,

C4 Diviser pour régner

Principe de la méthode

La méthode diviser pour régner (en anglais *divide and conquer*) est une technique algorithmique qui consiste à :

1. décomposer le problème initial en un ou plusieurs sous problèmes de taille inférieure,
2. résoudre chacun des sous problèmes,

C4 Diviser pour régner

Principe de la méthode

La méthode diviser pour régner (en anglais *divide and conquer*) est une technique algorithmique qui consiste à :

1. décomposer le problème initial en un ou plusieurs sous problèmes de taille inférieure,
2. résoudre chacun des sous problèmes,
3. combiner les solutions des sous problèmes pour obtenir la solution au problème initial.

C4 Diviser pour régner

Principe de la méthode

La méthode diviser pour régner (en anglais *divide and conquer*) est une technique algorithmique qui consiste à :

1. décomposer le problème initial en un ou plusieurs sous problèmes de taille inférieure,
2. résoudre chacun des sous problèmes,
3. combiner les solutions des sous problèmes pour obtenir la solution au problème initial.

Exemple

L'algorithme de recherche dichotomique dans un tableau **trié** déjà rencontré en classe de première est un exemple de la méthode diviser pour régner. Le sous problème est alors la recherche dans une liste de taille deux fois plus petite.

C4 Diviser pour régner

Tri fusion

L'algorithme du tri fusion (en anglais *merge sort*) illustre parfaitement la méthode diviser pour régner, en effet, il consiste à

C4 Diviser pour régner

Tri fusion

L'algorithme du tri fusion (en anglais *merge sort*) illustre parfaitement la méthode diviser pour régner, en effet, il consiste à

1. Décomposer la liste en deux sous listes de longueur égale (à une unité près).

C4 Diviser pour régner

Tri fusion

L'algorithme du tri fusion (en anglais *merge sort*) illustre parfaitement la méthode diviser pour régner, en effet, il consiste à

1. Décomposer la liste en deux sous listes de longueur égale (à une unité près).
2. Trier chacune des sous listes (c'est donc un algorithme récursif)

C4 Diviser pour régner

Tri fusion

L'algorithme du tri fusion (en anglais *merge sort*) illustre parfaitement la méthode diviser pour régner, en effet, il consiste à

1. Décomposer la liste en deux sous listes de longueur égale (à une unité près).
2. Trier chacune des sous listes (c'est donc un algorithme récursif)
3. Fusionner les parties triées

C4 Diviser pour régner

Exemples

Pour illuster la méthode diviser pour régner, on peut aussi citer :

C4 Diviser pour régner

Exemples

Pour illuster la méthode diviser pour régner, on peut aussi citer :

- La tri rapide (en anglais *quicksort*),

C4 Diviser pour régner

Exemples

Pour illuster la méthode diviser pour régner, on peut aussi citer :

- La tri rapide (en anglais *quicksort*),
- Le quart de tour d'une image,

C4 Diviser pour régner

Exemples

Pour illuster la méthode diviser pour régner, on peut aussi citer :

- La tri rapide (en anglais *quicksort*),
- Le quart de tour d'une image,
- La recherche des deux points les plus proches,

C4 Diviser pour régner

Exemples

Pour illuster la méthode diviser pour régner, on peut aussi citer :

- La tri rapide (en anglais *quicksort*),
- Le quart de tour d'une image,
- La recherche des deux points les plus proches,
- L'algorithme de multiplication rapide de Karatsuba.