Algo 2 – séance 3
Diviser pour Régner

Denis Trystram

27 février 2017

Bilan des séances précédentes

Que fallait-il retenir ?

Analyses de coût (au pire cas et en moyenne).
Rappels sur la récursivité à travers l'exemple du tri fusion (mécanisme et calcul de coût).

Pour les calculs de coût, il est essentiel de savoir écrire la mise en équation, la résolution pouvant être assez technique et difficile...

Bilan des séances précédentes

Que fallait-il retenir ?

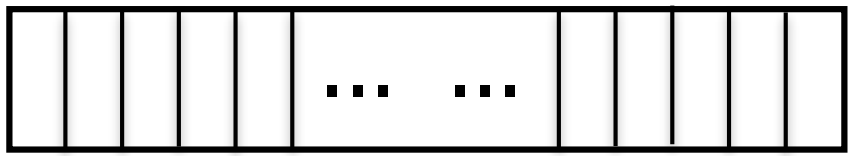
Analyses de coût (au pire cas et en moyenne).
Rappels sur la récursivité à travers l'exemple du tri fusion (mécanisme et calcul de coût).

Pour les calculs de coût, il est essentiel de savoir écrire la mise en équation, la résolution pouvant être assez technique et difficile...

Outline

On reprend Tri Fusion

Considérons un tableau de n éléments :

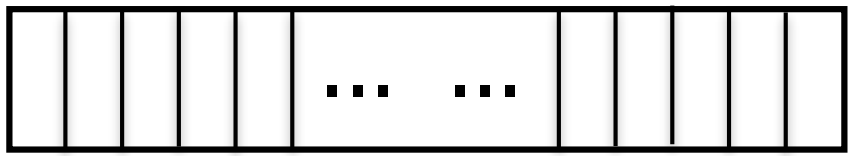


Principe :

- on découpe le tableau en deux parties égales
- on trie les deux sous-tableaux (récursivement)
- on les fusionne (en temps linéaire en la taille du tableau)

On reprend Tri Fusion

Considérons un tableau de n éléments :

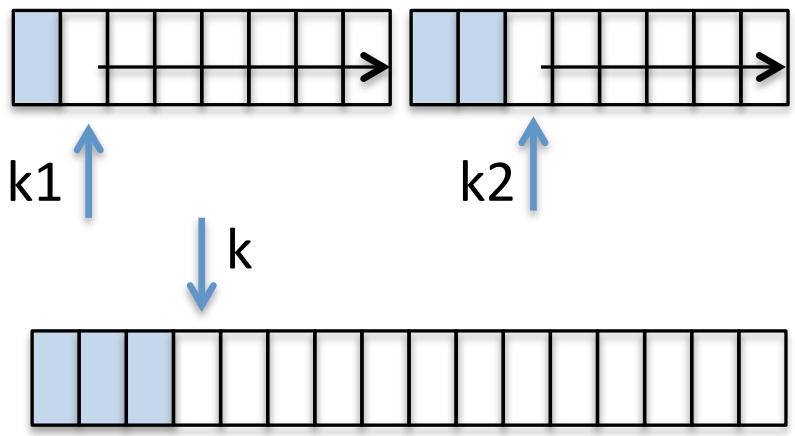


Principe :

- on découpe le tableau en deux parties égales
- on trie les deux sous-tableaux (récursivement)
- on les fusionne (en temps linéaire en la taille du tableau)

Rappel

Principe de la fusion de deux tableaux.
On considère un tableau supplémentaire dans lequel on fait la fusion des deux sous-tableaux.



Diviser pour régner

Chaque sous-tableau est de nouveau re-découper jusqu'à ce que la taille soit élémentaire (trié).

La résolution est une méthode de type *diviser pour régner* que l'on va formaliser et analyser.

Objectifs de la séance

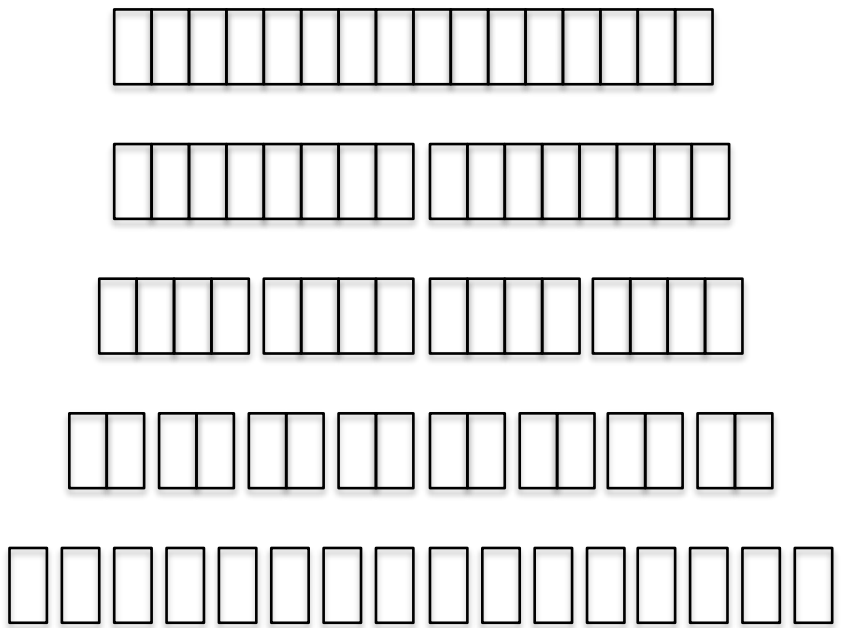
Présenter la méthode de *diviser pour régner*.

Faire l'analyse de coût.

- Applications sur quelques exemples (on reprend rapidement le tri fusion + enveloppe convexe)
- Démonstration générale du **Master Theorem**

Tri Fusion et diviser pour régner

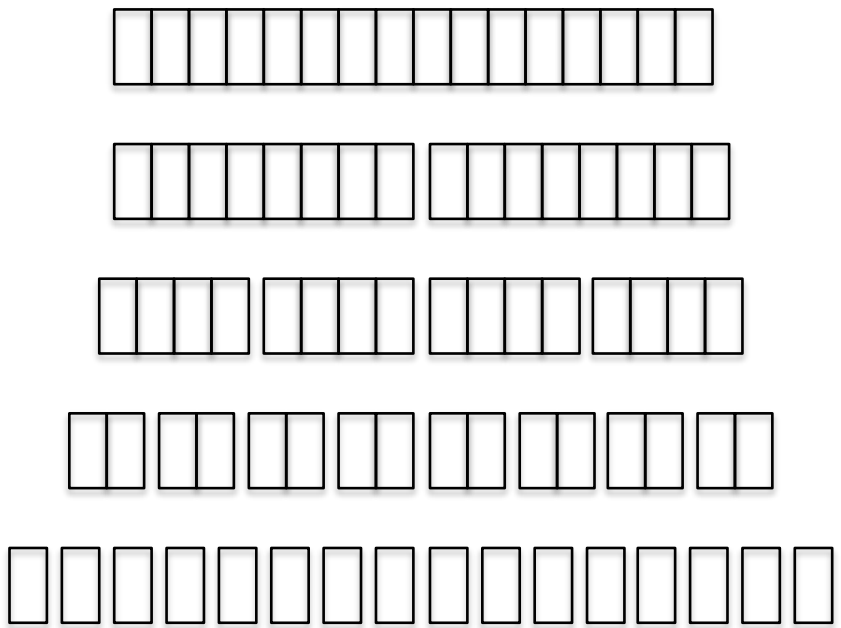
Découpes (coût constant pour chaque opération)



Ici, on découpe exactement en 2 parties chacune contenant la moitié du tableau (on suppose $n = 2^k$). Ici, $n = 16$.

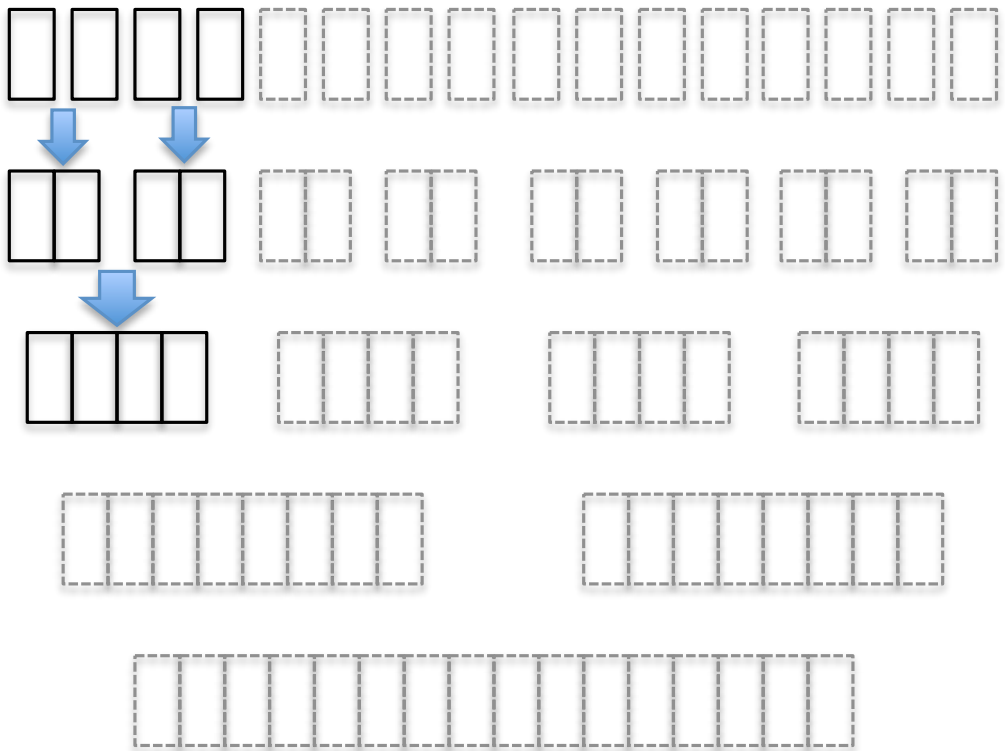
Tri Fusion et diviser pour régner

Découpes (coût constant pour chaque opération)

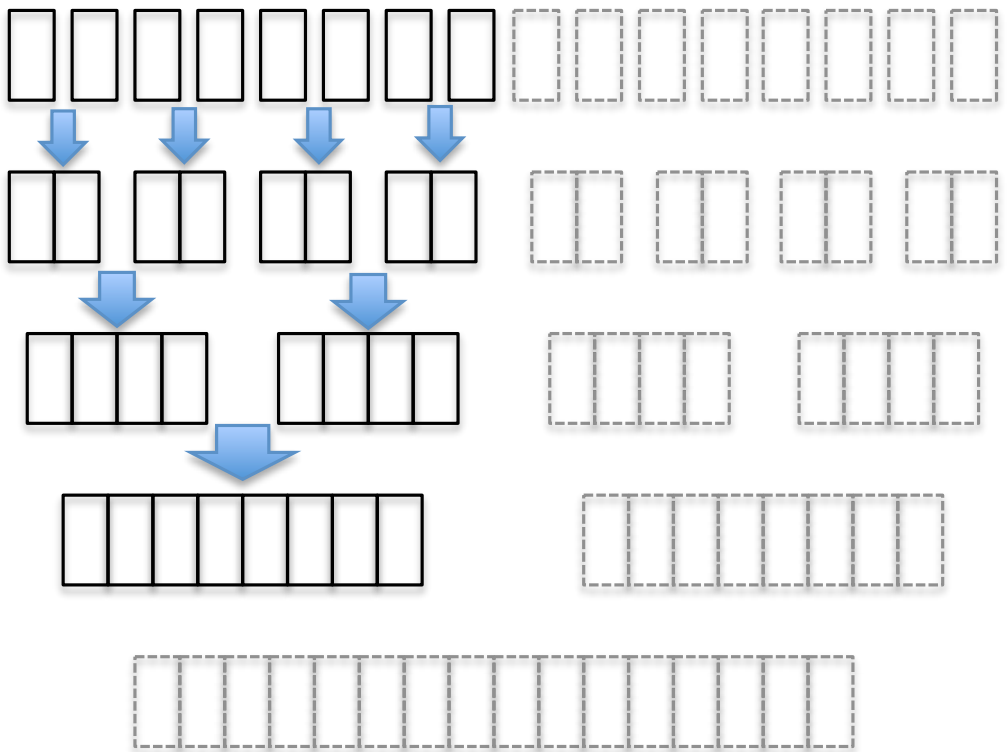


Ici, on découpe exactement en 2 parties chacune contenant la moitié du tableau (on suppose $n = 2^k$). Ici, $n = 16$.

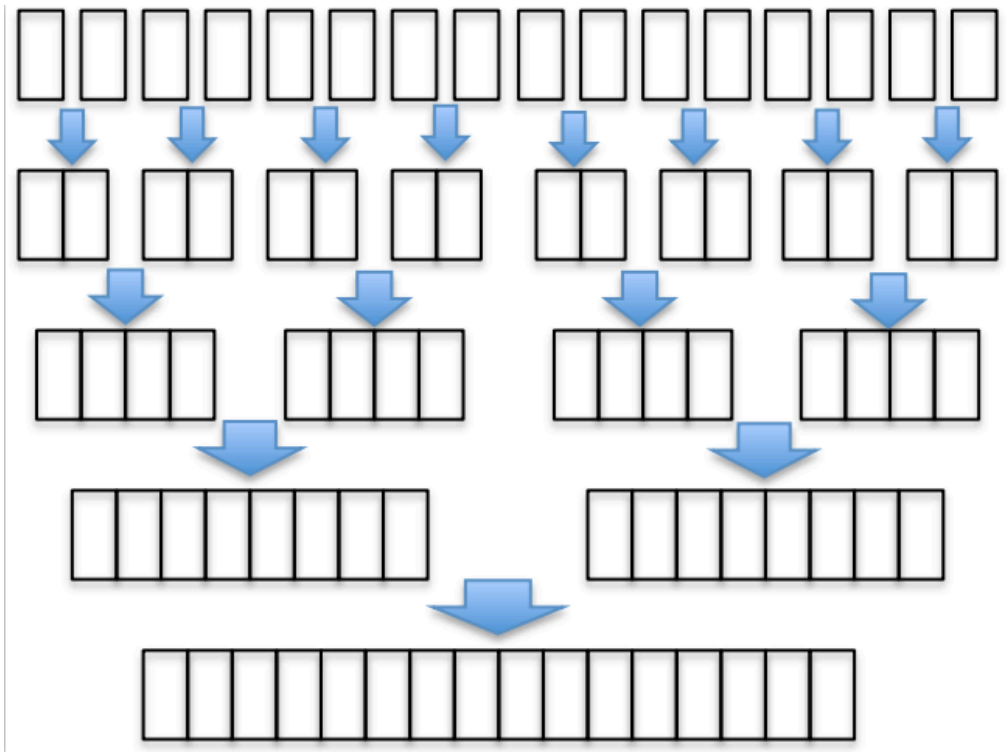
On fusionne (1)



On fusionne (2)

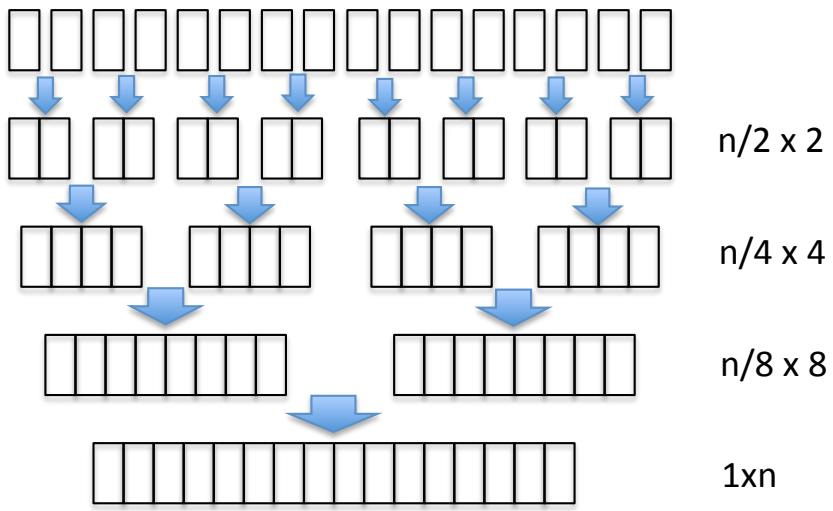


Fusion complète



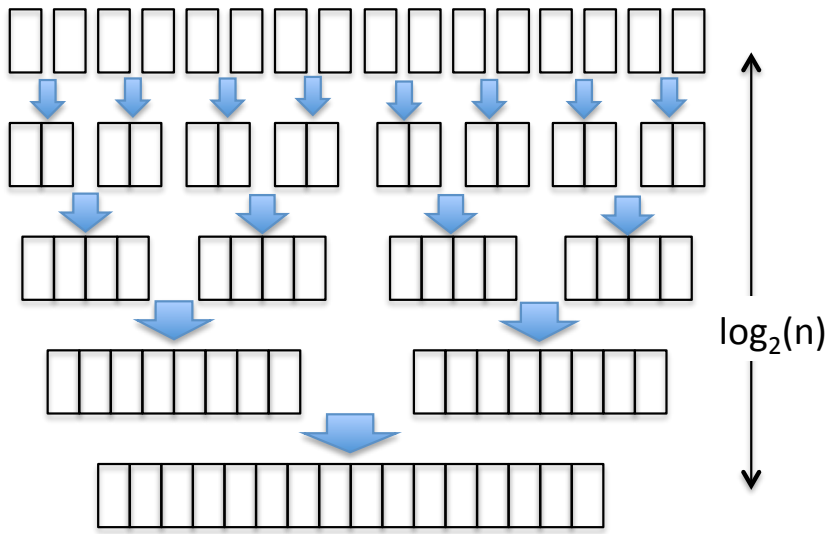
Analyse de coût (1)

Chaque niveau requiert le même nombre d'opération : n.



Analyse de coût (2)

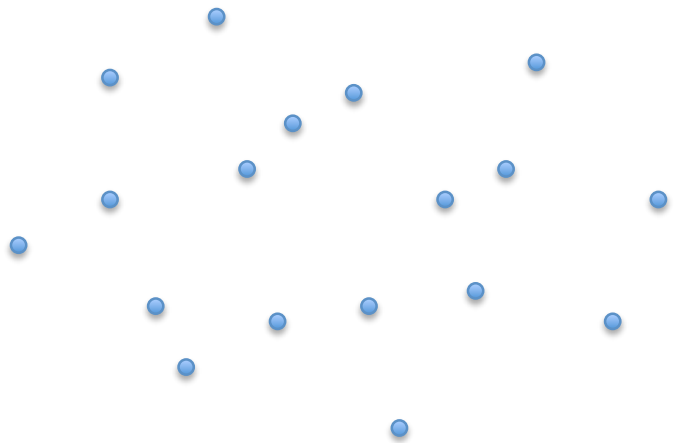
Profondeur de l'arbre des appels : $log_2(n)$.



Au total : $n.log_2(n)$.

Autre exemple : Calcul de l'enveloppe convexe

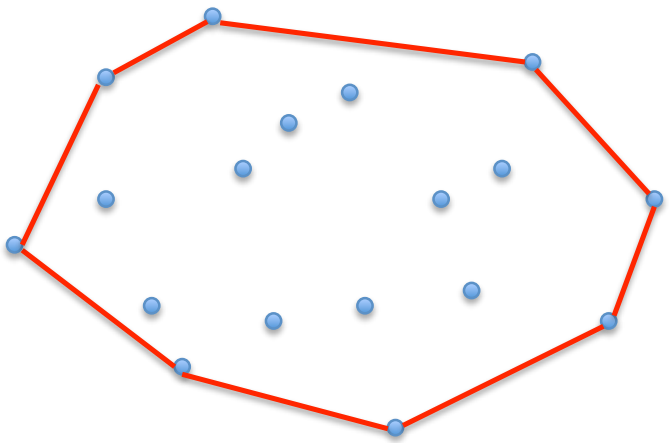
On considère un ensemble de points dans un plan.



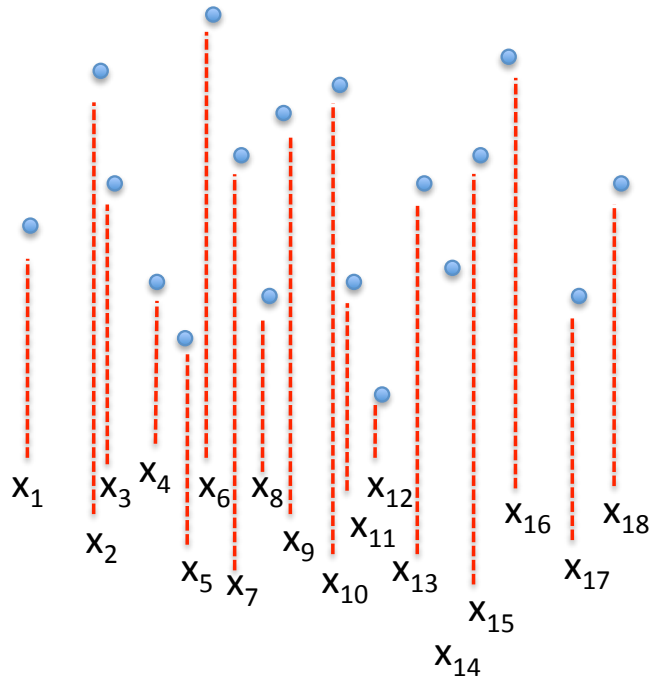
Calcul de l'enveloppe convexe

Définition :

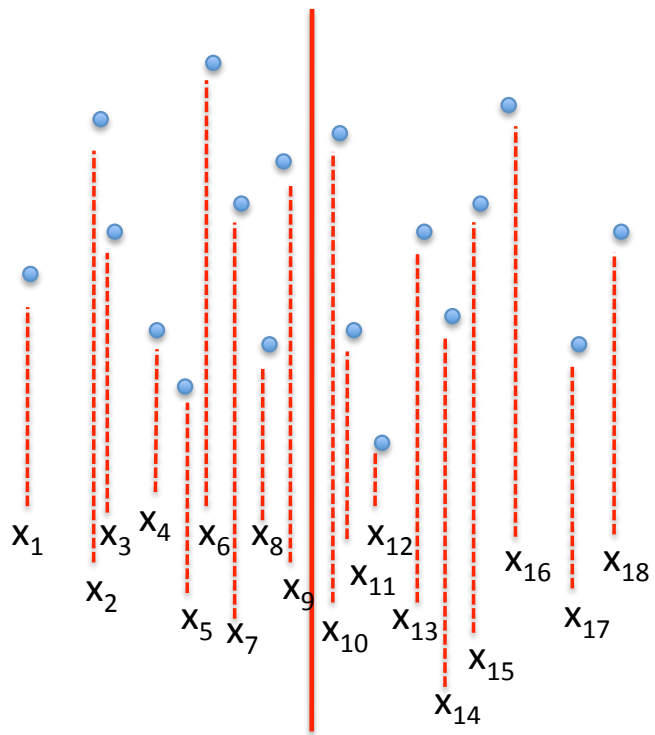
L'enveloppe convexe d'un ensemble de points est le plus petit polygone qui contient tous ces points.



On trie l'ensemble de points par abscisses croissantes :

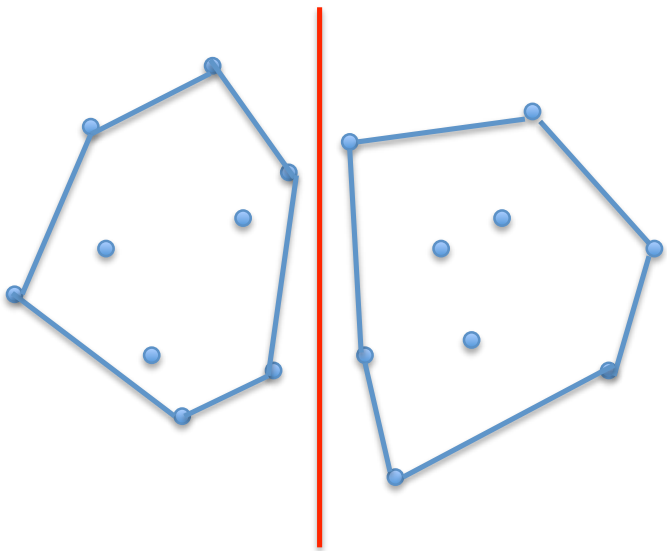


On partitionne l'ensemble de points selon ce tri :



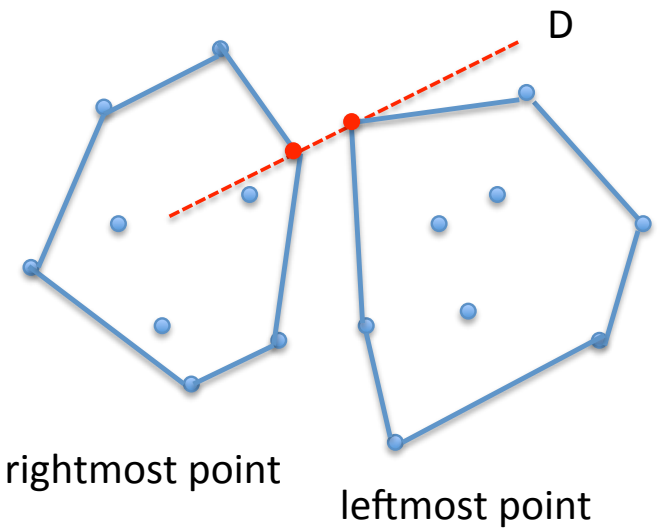
Diviser pour régner

Appel récursif sur chacune des deux parties

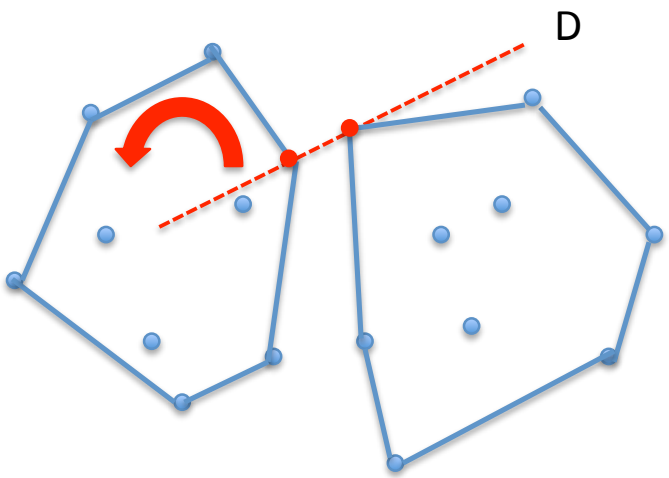


Fusion (1)

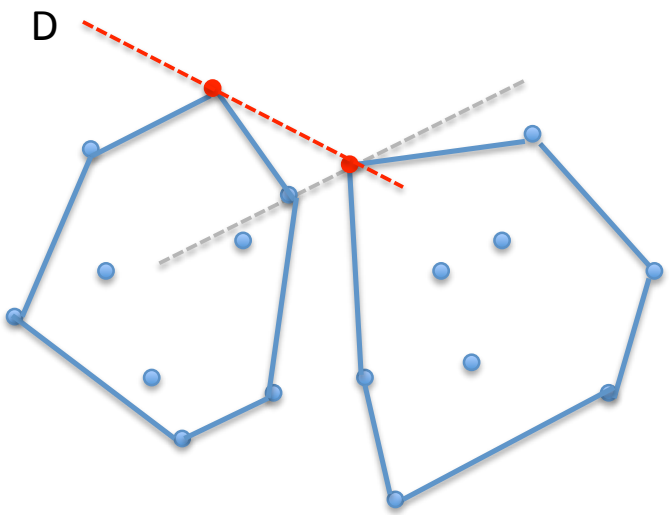
On va déterminer les hyper-tangentes supérieure et inférieure.



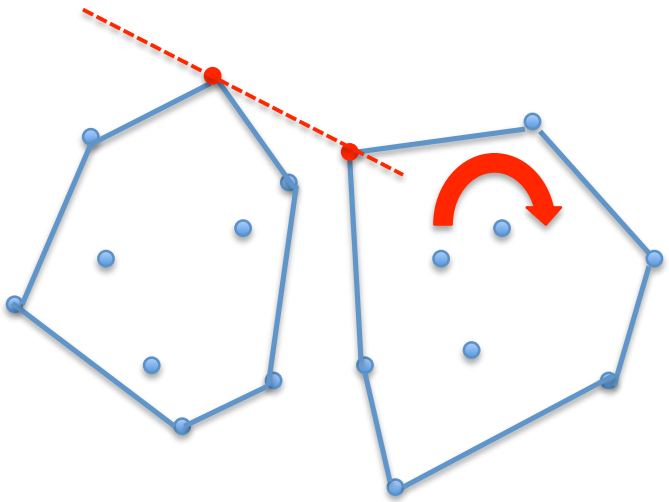
Fusion (2)



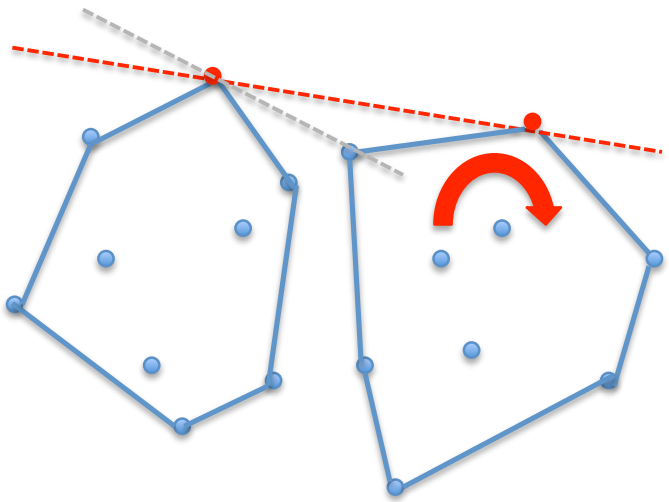
Fusion (3)



Fusion (4)

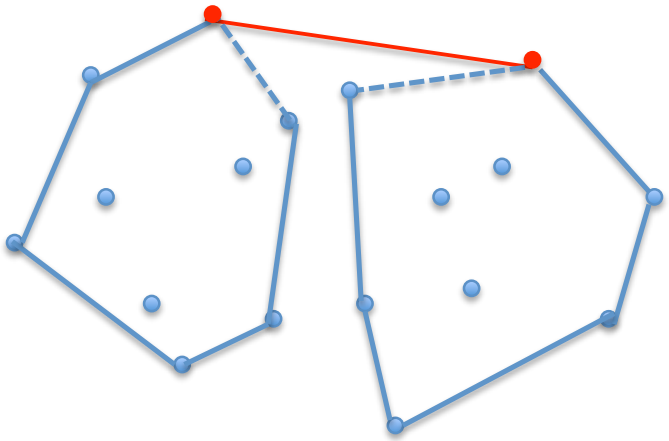


Fusion (5)



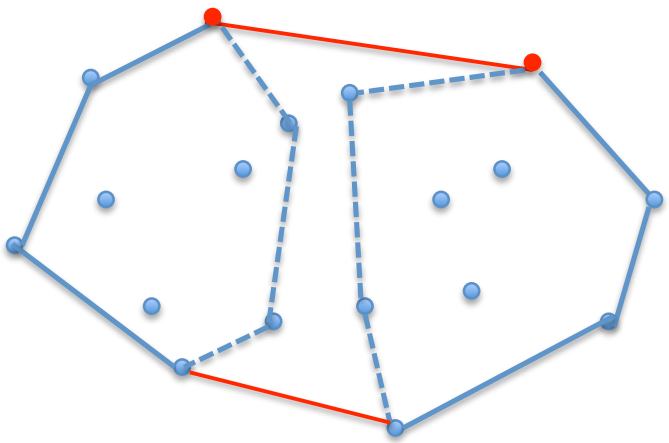
Fusion (6)

On obtient finalement l'hyper-tangente supérieure.



Fusion (7)

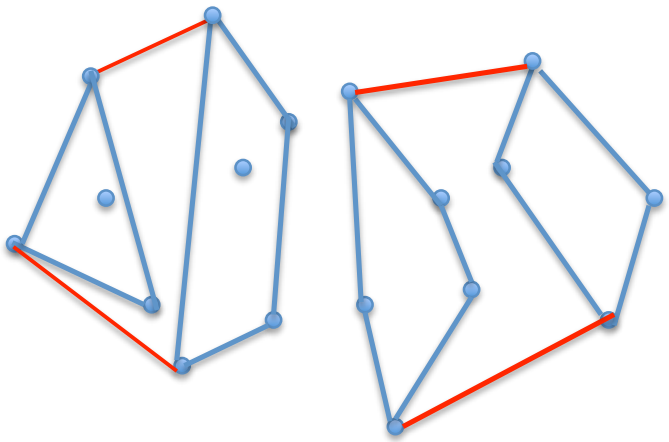
idem pour obtenir l'hyper-tangente inférieure



Coût de la fusion

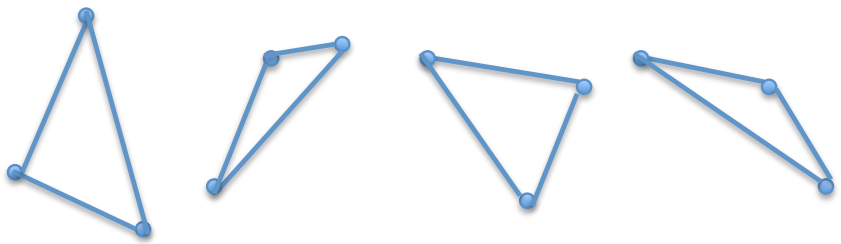
La fusion est majorée par une fonction linéaire en nombre de points.

On continue à descendre dans la récursivité...



Cas terminal

La condition d'arrêt est une analyse de cas (parmi 4 configurations)



4 types of triangles

Le coût reste en $\Theta(1)$.

Coût total

L'analyse est identique à celle du tri fusion :
On découpe exactement en deux l'ensemble des points.

$log_2(n)$ étapes chacune majorée par n

Donc, $n.log_2(n)$

Recursive tree

La méthode détaillée pour l'analyse de coût du tri fusion est généralisable :

Principe :

On construit l'arbre dont les **sommets** sont étiquetés par le coûts des sous-opérations et les **arêtes** correspondent au découpage en sous-problèmes.

Principe du diviser pour régner

Une idée *naturelle*.

Motivation

Divide and conquer est un paradigme pour concevoir certains algorithmes sur des problèmes dont on peut décomposer les entrées.

Principe pour un problème de taille n:
Si n est "petit" on calcule la solution par n'importe quelle méthode.
Sinon

1. Décomposer le problème en k sous-problèmes de taille n_i.
2. Résoudre les k sous-problèmes de tailles n_i $(1 \leq i \leq k)$.
3. Reconstruire la solution originale à partir des k solutions partielles

Principe du diviser pour régner

Une idée *naturelle*.

Motivation

Divide and conquer est un paradigme pour concevoir certains algorithmes sur des problèmes dont on peut décomposer les entrées.

Principe pour un problème de taille n:
Si n est "petit" on calcule la solution par n'importe quelle méthode.
Sinon

1. Décomposer le problème en k sous-problèmes de taille n_i.
2. Résoudre les k sous-problèmes de tailles n_i $(1 \leq i \leq k)$.
3. Reconstruire la solution originale à partir des k solutions partielles

Lien avec la récursivité

Généralement, les k sous-problèmes sont résolus par la même méthode (récursivement).

Analyse de coût (générale)

Le coût correspondant est obtenu par la résolution de l'équation de récurrence suivante :

$T(1) \leq$ Cste.
$T(n) = \sum_{1 \leq i \leq k} T(n_i) + c_1(n) + c_3(n)$
où c_1 et c_3 sont les coûts respectifs des phases de décomposition/reconstruction (1) et (3) précédentes.

Analyse de coût simplifiée

Pratiquement, la plupart des méthodes sont des partitionnements en a sous-problèmes identiques ou de mêmes tailles $n_i = \frac{n}{b}$.
L'équation du coût se simplifie alors en :

$$T(1) = 1$$
$$T(n) = a.T(\tfrac{n}{b}) + c(n)$$

Pour le tri Fusion ou l'enveloppe convexe, $a = b = 2$ et $c(n) = n$

Analyse de coût simplifiée

Pratiquement, la plupart des méthodes sont des partitionnements en a sous-problèmes identiques ou de mêmes tailles $n_i = \frac{n}{b}$.
L'équation du coût se simplifie alors en :

$T(1) = 1$
$T(n) = a.T(\frac{n}{b}) + c(n)$

Pour le tri Fusion ou l'enveloppe convexe, $a = b = 2$ et $c(n) = n$

Hypothèses pour le calcul

Deux considérations techniques pour simplifier les calculs :

- En pratique, la taille de l'instance n ne se divise pas exactement en parties égales.
- On simplifie souvent les conditions aux frontières $\mathscr{O}(1)$.

Cas non réguliers

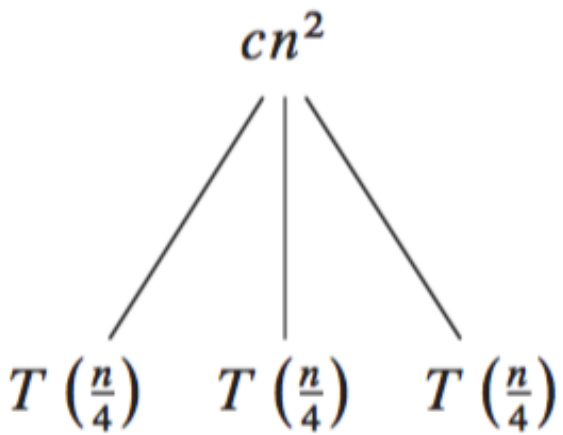
Ici, on suppose que le découpage n'est pas égal dans chaque sous-problèmes :
par exemple, on divise en deux problèmes ($a = 2$) de tailles respectives $\frac{1}{3}$ et $\frac{2}{3}$.

L'équation de coût devient :
$T(1)$
$T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + f(n)$

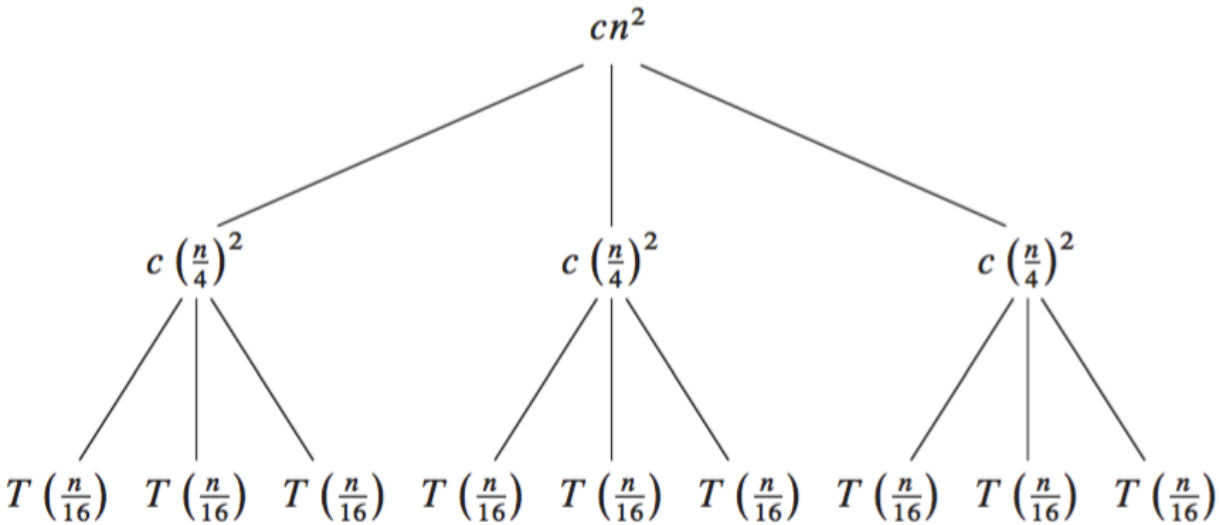
Exemple

On découpe en 3 et le coût des sous-opérations est quadratique, chaque sous-problème étant divisé par 4 ($a = 3$ et $b = 4$).

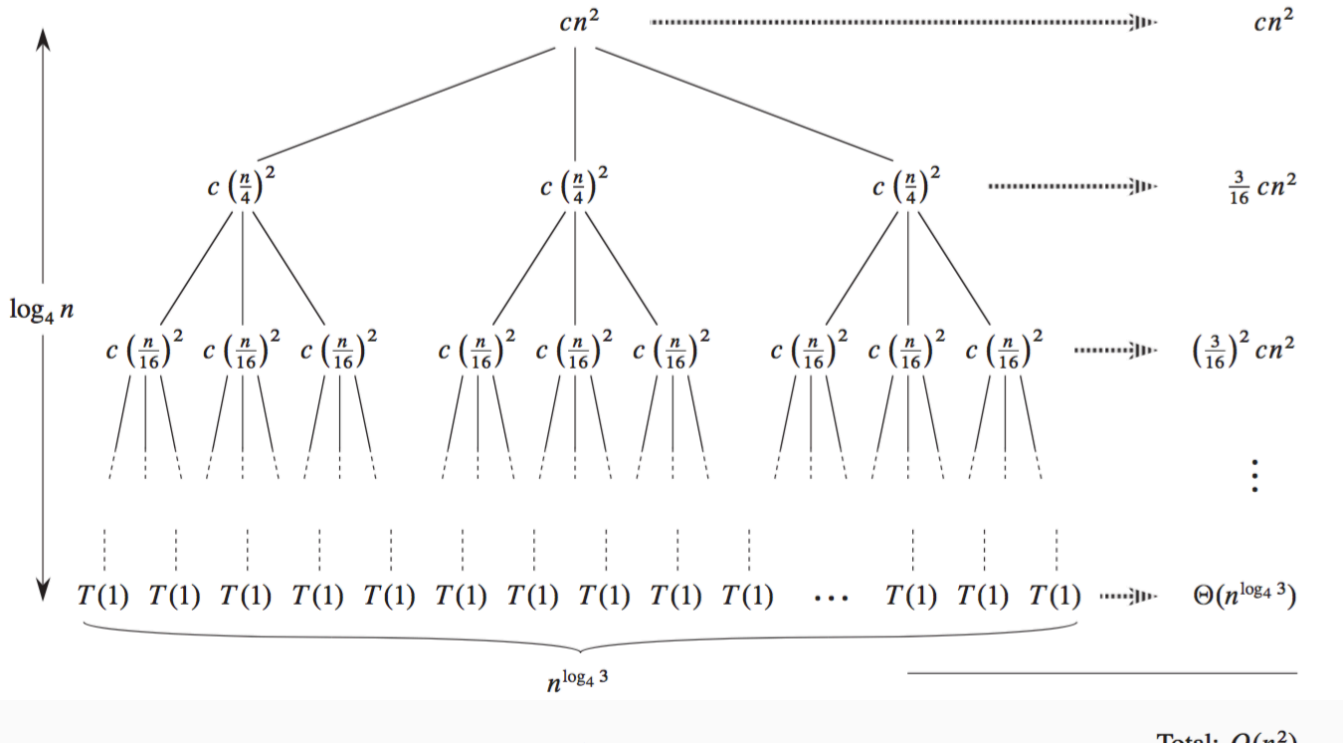
L'équation de récurrence s'écrit :
$T(1) = Cste$
$T(n) = 3.T(n/4) + \Theta(n^2)$.



Recursive tree



Analyse intuitive



Calcul détaillé (analytique)

Premier niveau $(i = 0)$: $c.n^2$

Niveau i, $1 \leq i \leq log_4(n) - 1$:
$3^i.c(n/4^i)^2$

Le dernier niveau correspond à : $3^{log_4(n)}$ sommets $= n^{log_4(3)}$ où $T(1) = 1$.

Donc, au total $T(n) = \sum_{0 \leq i \leq log_4(n)-1} (\frac{3}{16})^i.c.n^2 + \Theta(n^{log_4(3)})$

Résultat final

$T(n) = \sum_{0 \leq i \leq log_4(n)-1} (\frac{3}{16})^i . c . n^2 + \Theta(n^{log_4(3)})$.

On écrit :
$\sum_{0 \leq i \leq log_4(n)-1} (\frac{3}{16})^i . c . n^2 < c . n^2 \sum_{0 \leq i \leq \infty} (\frac{3}{16})^i$
$= \frac{1}{1-3/16} = \frac{16}{13}$

Donc, $T(n) = \Theta(n^2)$,

Ici, **le calcul est dominé par la racine de l'arbre...**

Petite variante

$T(n) = \sum_{0 \leq i \leq log_4(n)-1} (\frac{3}{16})^i . c . n^2 + \Theta(n^{log_4(3)})$

$T(n) = \Theta(n^2)$ et le calcul est dominé par la racine de l'arbre...

Que deviendrait le coût avec une fusion en racine de n ?
$T(n) = \sum_{0 \leq i \leq log_4(n)-1} (\frac{3}{16})^i . c . n^{1/2} + \Theta(n^{log_4(3)})$

Le terme dominant est $\Theta(n^{log_4(3)}) \approx n^{0.79}$, **le calcul est dominé par le nombre de feuilles du recursive tree**...

Petite variante

$T(n) = \sum_{0 \leq i \leq log_4(n)-1} (\frac{3}{16})^i . c . n^2 + \Theta(n^{log_4(3)})$

$T(n) = \Theta(n^2)$ et le calcul est dominé par la racine de l'arbre...

Que deviendrait le coût avec une fusion en racine de n ?
$T(n) = \sum_{0 \leq i \leq log_4(n)-1} (\frac{3}{16})^i . c . n^{1/2} + \Theta(n^{log_4(3)})$

Le terme dominant est $\Theta(n^{log_4(3)}) \approx n^{0.79}$, **le calcul est dominé par le nombre de feuilles du recursive tree**...

Généralisation : Master Theorem

On cherche ici à résoudre l'équation générale :
$T(1) = \Theta(1)$
$T(n) = a.T(n/b) + f(n)$
où $a \geq 1$ et $b > 1$ et $f(n)$ est une fonction asymptotiquement positive et multiplicative.

Le master theorem donne une caractérisation asymptotique du coût.

Généralisation : Master Theorem

$T(n)$ a le comportement asymptotique suivant

1. si $f(n) \in \mathcal{O}(n^{log_b a - \varepsilon})$ alors $T(n) \in \Theta(n^{log_b a})$
2. si $f(n) \in \Theta(n^{log_b a})$ alors $T(n) \in \Theta(n^{log_b a} log(n))$
3. si $f(n) \in \Omega(n^{log_b a + \varepsilon})$ et si $a.f(n/b) \leq cf(n)$ pour une constante $c < 1$ alors $T(n) \in \Theta(f(n))$

où ε est un nombre réel positif.

Interprétation

Intuitivement, le théorème dit que $T(n)$ est déterminé par la dominance de l'une des fonctions découpe/évaluation des sous-problèmes.

Si $n^{log_b a}$ domine, alors la solution est en $\Theta(n^{log_b a})$, si c'est le contraire alors la solution est en $\Theta(f(n))$.
Si elle est équilibrée, la solution est en $\Theta(f(n).log(n))$[1]

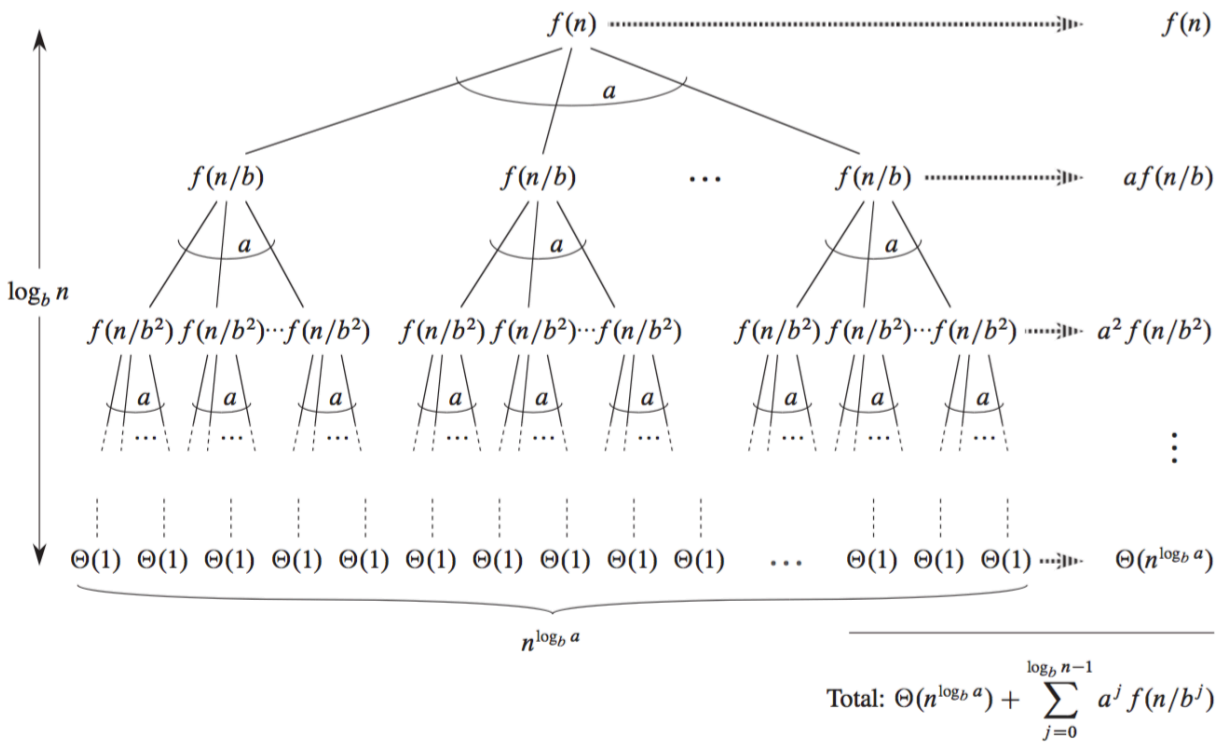
[1] comme dans les cas du tri Fusion ou de l'enveloppe convexe

Analyse

Elle n'est possible que dans des cas restreints.
$a \geq 1$ et $b > 1$ et $f(n)$ est une fonction (asymptotiquement) positive et multiplicative définie sur les puissances de b.
$T(1) = 1$
$T(n) = a.T(n/b) + f(n)$ si n est une puissance de b

Hypothèses simplificatrices

On suppose que n est parfaitement divisible par b^i et que f est définie sur les puissances de b.


$f(n)$
$f(n)$
a
$f(n/b)$
$f(n/b)$
$\cdots$
$f(n/b)$
$af(n/b)$
$\log_b n$
$f(n/b^2)\, f(n/b^2) \cdots f(n/b^2)$
$a^2 f(n/b^2)$
$\Theta(1)$
$\Theta(n^{\log_b a})$
$n^{\log_b a}$
Total: $\Theta(n^{\log_b a}) + \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j)$

Synthèse

Diviser pour régner est une technique efficace, elle ne n'adresse cependant pas à tous les problèmes. Les points importants sont :

- Trouver la bonne décomposition en sous-problèmes.
- Savoir appliquer le *Master Theorem*, au moins en ordre de grandeur.