

Diviser pour Résoudre

D.E ZEGOUR

École Supérieure d'Informatique

ESI

Diviser pour résoudre

Sommaire

A. Principe général

B. Applications

- . Tri par fusion
- . Multiplication de grands nombres
- . Organisation d'un tournoi
- . Tour de Hanoi

C. Recommandations

Diviser pour résoudre

Principe général

Diviser le problème de taille n en sous-problèmes plus petits
de manière que

la solution de chaque sous-problème facilite la construction du problème entier.
(Analyse descendante)

Diviser pour résoudre

Schéma général

DPR(x) :

Si x est suffisamment petit ou simple :

Retourner $A(x)$

Sinon

Décomposer x en sous exemplaires $x_1, x_2 \dots x_k$

Pour $i=1, k$: $y_i := \text{DPR}(x_i)$ Finpour

Retourner les y_i pour obtenir une solution à x

Retourner y

Fsi

Diviser pour résoudre

Applications 1 : Tri par fusion d'une liste L de n éléments avec $n = 2^k$.

Une technique "diviser pour résoudre" peut être la suivante:

Pour trier une liste de n éléments, il faut trier deux listes de $n/2$ éléments chacune puis de faire l'interclassement entre ces deux listes.

Pour trier une liste de $n/2$ éléments il faut trier deux listes de $n/4$ éléments chacune puis de faire leur interclassement.

Et ainsi de suite jusqu'à l'obtention de listes d'un seul élément.

Diviser pour résoudre

Applications 1 : Tri par fusion d'une liste L de n éléments avec $n = 2^k$.

Énoncé de manière récursive :

Tri(L, n)

Si $n = 1$

 Tri := L

Sinon

 L1 := L [1..n/2]

 L2 := L [n/2+1..n]

 Tri := Fusion(Tri(L1, n/2), tri(L2, n/2))

Fsi

Diviser pour résoudre

Applications 1 : Tri par fusion d'une liste L de n éléments avec $n = 2^k$.

Équation de récurrence associée :

$$\begin{aligned} T(n) &= a \text{ si } n = 1 \\ &= 2T(n/2) + bn \text{ si } n > 1 \end{aligned}$$

--> Solution $O(n \log n)$.

Diviser pour résoudre

Applications 1 : Tri par fusion d'une liste L de n éléments avec $n = 2^k$.

Si nous voulons construire une solution non récursive,

- En remontant dans les appels récursifs, on arrive à l'algorithme de VON-NEUMANN suivant :

Ca revient donc à considérer au départ n sous listes de 1 élément, donc triées.

Ensuite on fusionne deux à deux,

puis quatre en quatre.

etc.

Diviser pour résoudre

Applications 2 : Multiplication des grands entiers

X, Y ayant n chiffres binaires. (n suffisamment grand)

Une solution " Diviser pour résoudre" de ce problème : scinder X et Y en deux entiers de n/2 chiffres binaires chacun.

Méthode habituelle : $O(n^2)$

$$\begin{array}{lcl} X = A B & \text{-->} & X = A 2^{n/2} + B \\ Y = C D & \text{-->} & Y = C 2^{n/2} + D \end{array}$$

$$X = A 2^{n/2} + B$$

$$Y = C 2^{n/2} + D$$

Diviser pour résoudre

Applications 2 : Multiplication des grands entiers

$$X Y = AC 2^n + (AD + BC)2^{n/2} + BD$$

4 multiplications de $n/2$ chiffres (AC, AD, BC, BD)

3 additions

2 décalages (multiplication par 2^n et $2^{n/2}$)

Équation :

$$T(n) = 4T(n/2) + cn$$

$$T(1) = 1$$

Utiliser la méthode *par substitution*

Solution : $T(n) = O(n^2)$

$$X Y = AC 2^n + (AD + BC)2^{n/2} + BD$$

Diviser pour résoudre

$$x^{\lg_b(n)} = n^{\lg_b(x)}$$

Applications 2 : Multiplication des grands entiers

En diminuant le nombre de multiplications, on peut diminuer la complexité.

$$X Y = AC2^n + [(A-B)(D-C) + AC + BD]2^{n/2} + BD$$

3 multiplications (AC, BD, (A-B)(D-C)), 6 additions, 2 décalages

$$\text{Équation : } T(n) = 3T(n/2) + cn$$

Poser $n = 2^k$

$$t_k = T(2^k)$$

$$T(2^k) = 3 T(2^{k-1}) + c2^k$$

$$T(2^k) - 3 T(2^{k-1}) = c2^k$$

$$T_k - 3 T_{k-1} = c2^k$$

Equation caractéristique

$$(x-3)(x-2) = 0$$

$$\text{Solution : } T_k = c_1 3^k + c_2 2^k$$

C'est à dire

$$O(3^k) = O(3^{\lg_2(N)}) =$$

$$O(n^{\lg_2(3)}) = O(n^{1.59})$$

Diviser pour résoudre

Applications 2 : Multiplication des grands entiers

Remarques :

- Méthode asymptotiquement plus efficace mais beaucoup moins limpide de point de vue pédagogique.
- Efficace pour les grands nombres(>500 chiffres)

Diviser pour résoudre

Applications 2 : Multiplication des grands entiers

Algorithme récursif:

Fonction Mult(X, Y, n)

Si n = 1

 Si X=1 et Y=1 Retourner(1) Sinon retourner(0) Fsi

Sinon

 A := n/2 premiers bits de X; B := n/2 derniers bits de X

 C := n/2 premiers bits de Y D := n/2 derniers bits de Y

 m1 := Mult(A,C, n/2); m2 = Mult(A-B,D-C, n/2); m3 := Mult(B,D, n/2) ;

 P1 := Décalage(m1, n) ; P2 := Decalage(m1 + m2 + m3)

 Retourner(P1 + P2 + m3)

Fsi

Diviser pour résoudre

Applications 3 : Organisation d'un tournoi circulaire

Organisation d'un tournoi comprenant $n=2^k$ joueurs (ou équipes).

Chaque joueur doit affronter chaque autre joueur une seule fois à raison d'un match par jour.

Donc la durée du tournoi est de $(n-1)$ jours.

Pour élaborer une solution pour n joueurs, on essaie d'élaborer une solution pour $n/2$ joueurs.

Pour élaborer une solution pour $n/2$ joueurs, on essaie d'élaborer une solution pour $n/4$.

Pour 2 joueurs la solution est triviale.

Diviser pour résoudre

Applications 3 : Organisation d'un tournoi circulaire :

$n = 2$

Joueurs	Journée1
1	2
2	1

Diviser pour résoudre

Applications 3 : Organisation d'un tournoi circulaire

$$n = 4 = 2^2$$

Joueurs	J1	J2	J3
1	2	3	4
2	1	4	3
3	4	1	2
4	3	2	1

Partie supérieure gauche : Table de l'étape k-1

Partie inférieure gauche : Ajouter 2^{k-1} à chaque élément de la partie supérieure gauche

Partie supérieure droite : Au $(k+1)$ i-ème jour mettre les indices $2^k + 1, 2^k + 2, \dots$ Avec permutation circulaire de gauche à droite.

Partie inférieure droite : Au $(k+1)$ i-ème jour mettre les indices $1, 2, \dots$ Avec permutation circulaire de droite à gauche.

Diviser pour résoudre

Applications 3 : Organisation d'un tournoi circulaire

$$n = 8 = 2^3$$

Joueurs	J1	J2	J3	J4	J5	J6	J7
1	2	3	4	5	6	7	8
2	1	4	3	6	7	8	5
3	4	1	2	7	8	5	6
4	3	2	1	8	5	6	7
5	6	7	8	1	4	3	2
6	5	8	7	2	1	4	3
7	8	5	6	3	2	1	4
8	7	6	5	4	3	2	1

Diviser pour résoudre

Applications 4 : Tour de Hanoi

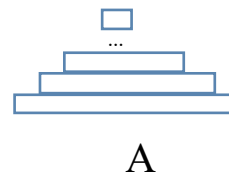
On dispose de 3 tours A, B et C.

Initialement, n disques de diamètres différents sont placés sur A.

Un disque plus grand ne doit jamais se trouver sur un disque plus petit.



Diviser pour résoudre



B

C

Applications 4 : Tour de Hanoi (Problème)

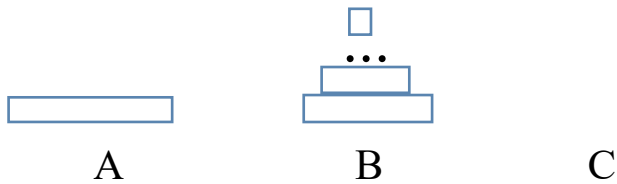
Transférer les n disques de A vers C, en utilisant B comme intermédiaire en respectant les deux règles suivantes :

A un moment donné, seul le disque au sommet d'un piquet peut être transféré vers un autre.

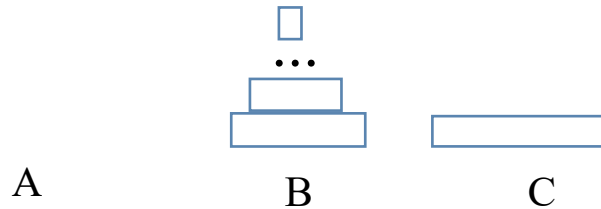
Un disque plus grand ne doit pas se trouver sur un disque plus petit.

Diviser pour résoudre

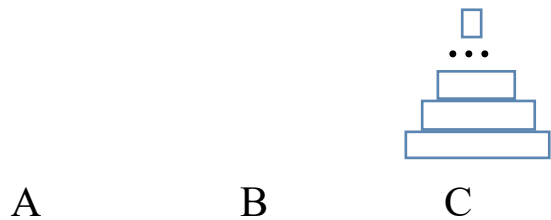
Applications 4 : Tour de Hanoi (Principe)



Transférer les $n-1$ disques au sommet de A vers B vers C comme auxiliaire.



Transférer le disque restant de A vers C.



Transférer les $n-1$ disques de B vers C avec A comme auxiliaire.

Diviser pour résoudre

Applications 4 : Tour de Hanoi (Algorithme)

Procédure TourdeHanoi(n ,de, vers, aux)

Si $n = 1$ { cas trivial } :

"transfert du disque 1 de", de, "vers", vers)

Sinon

{ Transférer les n-1 disques au sommet de A vers B avec C comme auxiliaire }

TourdeHanoi(n-1, de,aux,vers)

{ transférer le disque restant de A vers C }

"transférer le disque", n,"de",de,"vers", vers)

{ Transférer les n-1 disques de B vers C, utilisant A comme auxiliaire }

TourdeHanoi(n-1, aux, vers, de)

Fsi

Equation de récurrence $T(1) = a$; $T(n) = 2 T(n-1) + b$ Solution : $O(2^n)$

Diviser pour résoudre

Applications 4 : Tour de Hanoi (Algorithme itératif)

Si nous voulons construire un algorithme non récursif, en remontant dans les appels, on aboutit à l'algorithme itératif suivant :

- Imaginons que les piquets sont disposés en triangle.
- A tous les coups de rang impair, on déplace le plus petit disque sur le piquet suivant dans le sens des aiguilles d'une montre.
- A tous les coups de rang pair, on effectue le seul déplacement possible permis n'impliquant pas le plus petit disque.

Diviser pour résoudre

Recommandations

Cas général :

Découper le problème en sous problèmes.

Cas de récursion :

Ou bien donner une solution récursive,

Cas de non récursion

ou bien descendre jusqu'au cas trivial et en remontant dans l'arbre, on essaie de construire une solution non récursive.

Diviser pour résoudre

Recommandations : Équilibrer les sous-problèmes

Considérer le tri par insertion : On suppose $T[1..K]$ trié et on insère $T[K+1]$ par décalage

Tri(L, n) :

Si $n = 1$: retourner(L)

Sinon Retourner(Insère[Tri(n-1),T(n)]) Fsi

Division du problème en deux sous problèmes de taille différente, l'un de taille 1 et l'autre de taille $n-1$.

Équation : $T(1) = a$ et $T(n) = T(n-1) + bn$

Ce qui conduit à $O(n^2)$

Diviser pour résoudre

Recommandations

Le tri par fusion subdivise le problème en deux sous problèmes de taille $n/2$. Ce qui donne une complexité égale à $O(n \log(n))$.

La division d'un problème en sous problème de taille sensiblement égales contribue de manière cruciale à l'efficacité de l'algorithme.

Minimiser le nombre de sous problèmes