Diviser pour résoudre

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)**

- **Division progressive jusqu' a l'obtention d'un sous-problème de taille 1.**

Diviser pour résoudre

Schéma général

```
DPR(x) :
    Si x est suffisamment petit ou simple :
        . Retourner  A(x)
    Sinon
        . Décomposer x en sous exemplaires x1,x2...xk
        . Pour i=1, k : yi := DPR(xi)   Finpour
        . Retourner les yi 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. De la même façon, 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 :**

T(n) = a si n =1

= 2T(n/2) + dn sinon

- **Solution O(nLogn).**

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. ect...**

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 est de scinder X et Y en deux entiers de n/2 chiffres binaires chacun.**
- **Méthode habituelle : $O(n^2)$**

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

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

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 :**

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

  $T(1) = 1$

- **On peut trouver la solution par substitution ou bien en utilisant la technique des équations différentielles en posant par exemple $n = 2^k$, ce qui donne l'équation de récurrence $T(n) - T(n-1) = cn$ dont la solution est connue).**
- **Solution $T(n) = O(n^2)$**

Diviser pour résoudre

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
- **Équation : $T(n) = 3T(n/2) + cn$**
- **Solution $O(n^{1,59})$, avec $1,59 = \log_2 3$, ce qui est meilleur.**
- **<u>Remarques :</u>**

— **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

- **<u>Algorithme récursif:</u>**

```
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 2 : Multiplication des grands entiers

- **<u>Remarque :</u> on peut aussi essayer de construire l'algorithme non récursif en remontant dans les appels.**

Diviser pour résoudre

Applications 3 : Organisation d'un tournoi circulaire :

- **Organisation d'un tournoi comprenant $n=2^k$ joueurs (ou équipe). 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. De la même façon pour élaborer une solution pour n/2 joueurs, on essaie d'élaborer une solution pour n/4. Et ainsi de suite, jusqu'a atteindre 2 joueurs où la solution est triviale.**

Diviser pour résoudre

Applications 3 : Organisation d'un tournoi circulaire :

- **n = 2**

	1
1!	2
2!	1

Diviser pour résoudre

Applications 3 : Organisation d'un tournoi circulaire

- **$n = 4 = 2^2$**

	1	2	3
1!	2!	3	4
2!	1!	4	3
---	---	---	---
3!	4!	1	2
4!	3!	2	1

Diviser pour résoudre

Applications 3 : Organisation d'un tournoi circulaire

- **$n = 8 = 2^3$**

	1	2	3	4	5	6	7
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 3 : Organisation d'un tournoi circulaire :

- **Technique :**

© **Pour k = 1 (ou n=2), la construction de la table est triviale.**

© **A l'étape k :**

— **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 $2^k + 1$, $2^k + 2$, ... Avec permutation circulaire de gauche à droite.**

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

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. Le problème est de transférer les n disques de A vers C, en utilisant B comme intermédiaire en respectant les deux règles suivantes :**

- **à 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

- **Solution récursive simple et élégante.**
- **Supposons que nous avons une solution pour n-1 disques. Nous pourrons alors définir une solution pour n disques au moyen de la solution de n-1 disques.**
- **n=1 constitue le cas trivial : transférer le disque unique de A vers C.**
- **Pour transférer n disques de A vers C, en utilisant B comme auxiliaire, on procède comme suit:**
- **1.Si n=1, alors transférer l'unique disque de A vers C.**
- **2.Transférer les n-1 disques au sommet de A vers B vers C comme auxiliaire.**
- **3.Transférer le disque restant de A vers C.**
- **4.Transférer les n-1 disques de B vers C avec A comme auxiliaire.**

Diviser pour résoudre

Applications 4 : Tour de Hanoi

- **Algorithme :**

```
Procedure TourdeHanoi( n : integer; de, vers, aux : char);
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
```

Diviser pour résoudre

Applications 4 : Tour de Hanoi

- **Remarque :**

- **Si nous voulons construire un algorithme non récursif, en remontant dans les appels, on aboutit à l'algorithme 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, 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-),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(n) = c si n=1**
- **= T(n-1) + n**
- **Ce qui conduit à $O(n^2)$ (inférieur à O(nLog(n))**

Diviser pour résoudre

Recommandations

- **Le tri par fusion subdivise le problème en deux sous problème 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**