Introduction

- idee de base du cours: comment concevoir d'algorithmes efficaces (et savoir qu'ils sont corrects et efficaces ou savoir que c'est peu probable qu'un algorithme efficace existe);

- livre: Cormen, Leiserson, Rivest et Stein: Intro a l'algorithmique, Dunod, BU 681.31.01 COR;

- `web: www.labri.fr/~robson/AlgAv/`

- langage: EXALGO (voir lien sur la page web)

Diviser pour régner:

- Résoudre un probleme en résolvant 2 ou plus problémes du même type

- Pas la même chose que la programmation structurée: c'est le programme qui divise, pas le programmeur

- Souvent donne algorithmes efficaces

- Souvent donne algorithmes élégants

- Parfois conflit entre ces deux

tri rapide

- Choisir un élément *pivot* du tableau (méthode de base choisit le premier)

- Diviser le tableau en trois parties: ceux plus petits que le pivot, ceux égaux, ceux plus grands

- Appliquer la même méthode aux première et troisième parties

- Rapport avec arbre binaire de récherche
  Etant donné un ensemble de valeurs, en choisir une arbitrairement comme racine/pivot; diviser les autres en celles plus petites et celles plus grandes; appliquer le même processus aux deux parties (sauf si elles sont vides.)

tri par fusion

- Diviser les éléments en deux parties (de tailles équilibrées)

- Trier chaque partie

- Fusionner

Plus complexe que le précédent?
seulement dans le fait qu'on est obligé d'utiliser un deuxième tableau

minmax

Trouver le minimum et le maximum d'un tableau

- Si deux éléments, comparer les, sinon
- Diviser le tableau en deux parties (paires...)
- Trouver min et max de chaque partie
- Comparer les deux min et les deux max
- Approx $3n/2$ comparaisons

multiplication de longs entiers

Multiplication de deux entiers représentés par plusieurs chiffres à une base b

Addition est rapide (linéaire); multiplication lente

Considérer la moitié gauche des chiffres x et la moitié droite y; le nombre est $Bx+y$ où B est une puissance de b.
$(Bx_1+y_1)*(Bx_2+y_2)$
$= B^2x_1x_2 + B(x_1y_2+x_2y_1)+y_1y_2$

Calculer x_1x_2, y_1y_2 et $(x_1+y_1)(x_2+y_2)$ et soustraire.

Economiser une mutiplication à un coût de 3 addition/soustractions de plus

analyse pour nombres de n chiffres

- Considérer l'arbre de tous les appels de la fonction
- hauteur $\log_2 n$
- nombre d'appels multiplié par 3 à chaque niveau
- Aux feuilles: multiplication de chiffres
- nombre de multiplications $3^{\log_2 n} = n^{\log_2 3}$
- Plus additions à chaque autre niveau:
- Nombre d'additions aussi $O(n^{\log_2 3})$

L'algorithme de Strassen pour multiplication de matrices

Même idée (on suppose que les matrices sont carrées et de la même taille paire n): si le produit de A et B est C, on peut diviser chaque matrice en quatre blocs de taille $n/2$ et:

$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \times \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} =$$
$$\begin{pmatrix} A_{1,1}B_{1,1} + A_{1,2}B_{2,1} & A_{1,1}B_{1,2} + A_{1,2}B_{2,2} \\ A_{2,1}B_{1,1} + A_{2,2}B_{2,1} & A_{2,1}B_{1,2} + A_{2,2}B_{2,2} \end{pmatrix}$$

On peut réduire le nombre de multiplications de matrices $n/2 \times n/2$ de 8 à 7 par le même type d'astuce que dans la multiplication de longs entiers.

analyse

- Considérer l'arbre de tous les appels de la fonction

- hauteur $\log_2 n$

- nombre d'appels multiplié par 7 à chaque niveau

- Aux feuilles: multiplication d'éléments de base

- nombre de multiplications $7^{\log_2 n} = n^{\log_2 7}$

- Plus $O(n^{\log_2 7})$ additions