Université Lille1 Sciences et Technologies

UFR IEEA

ACT

2014–2015
Master 1 d'Informatique
TD-Feuille 2

Diviser pour régner

Rappel du Master Theorème. *Si* $T(n) = aT(\lfloor n/b \rfloor) + O(n^d)$, *avec* $a > 0, b > 1, d \geq 0$ *alors*

- *si* $d > log_b a$, $T(n) = O(n^d)$
- *si* $d = log_b a$, $T(n) = O(n^d log n)$
- *si* $d < log_b a$, $T(n) = O(n^{log_b a})$

Exercice 1 : Recherche d'une valeur encadrée

Soit T un tableau trié de n entiers et a et b deux entiers $(a < b)$.
Q 1. Proposer un algorithme en $O(\log n)$ qui retourne $Vrai$ Ssi il existe i tel que $a < T[i] < b$.
Q 2. Pour chacun des algorithmes suivants dire si ils correspondent bien à la spécification de la question précedente. Justifier.

```
//A1;
for (int i=0;i<n;i++)
   if ((T[i]>a) && (T[i]<b)) return true;
return false;
```

```
//A2;
int m;
int g=0;
int d=n-1;
  while (g<d)  {
     m= (g+d) /2;
     if (T[m]>=b) d=m;
     else if (T[m]<= a) g=m;
     else return true;
  }
  return ((T[d]>a) && (T[d]>b));
```

```
//A3;
boolean rech(int g,int d, int a, int b) {
int m= (g+d) /2;
   if (T[m]>=b) return rech(g,m-1,a,b);
   else if (T[m]<=a) return rech(m+1,d,a,b));
   else return true;
   //end if;}
rech(0,n-1, a,b);
```

```
//A4;
int m;
int g=0;
int d=n-1;
while (g<=d){
  m= (g+d) /2;
  if (T[m]>=b) d=m-1;
  else if (T[m]<=a) g=m+1;
  else return true;
}
 return false;
```

```
//A5;
int m;
int g=0;
int d=n-1;
  while (g<d) {
    m= (g+d) /2;
    if (T[m]>=b) d=m-1;
    else if (T[m]<= a) g=m+1;
    else return True;
  }
  return false;
```

Exercice 2 :

Soit un tableau trié par ordre croissant de n entiers distincts(qui peuvent être négatifs).
Proposer un algorithme en $O(\log n)$ qui retourne $Vrai$ Ssi il existe i tel que $T[i] = i$.

Exercice 3 : From "Algorithms", by S. Dasgupta, C. Papadimitriou, U. Vazirani

Suppose you are choosing between the following three algorithms:

- Algorithm A solves problems by dividing them into five subproblems of half the size, recursively solving each subproblem, and them combining the solutions in linear time.
- Algorithm B solves problems of size n by recursively solving two problems of size $n-1$ and them combining the solutions in constant time.
- Algorithm C solves problems of size n by dividing them into nine subproblems of size $n/3$, recursively solving each subproblem, and them combining the solutions in $O(n^2)$ time.

What are the running times of each of these algorithms (in big-O notation), and which would you choose?

A préparer pour la prochaine séance

Exercice 4 : MinMax

Proposer un algorithme de type "diviser pour régner" qui retourne le min et le max d'un tableau de n éléments en faisant au plus environ $3n/2$ comparaisons (pour l'analyse, se restreindre au cas où n est une puissance de 2). Pouvez-vous proposer un autre algorithme simple pour le même problème qui a une complexité équivalente et qui n'utilise pas le paradigme "Diviser pour régner"?

Exercices Supplémentaires

Exercice 5 : Produit de matrices: Algorithme de Strassen

Soient M et N deux matrices carrées $n \times n$.

Q 1. Quelle est la complexité de l'algorithme "classique" de multiplication de deux matrices?

On suppose que n est une puissance de 2. On découpe les matrices M et N en 4 blocs $n/2 \times n/2$ comme suit:

$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$, $N = \begin{pmatrix} E & F \\ G & H \end{pmatrix}$

Et donc le produit est:

$MN = \begin{pmatrix} AE+BG & AF+BH \\ CE+DG & CF+DH \end{pmatrix}$

Q 2. Déduire de ce qui précède un premier algorithme et calculer sa complexité.

Q 3. Soient maintenant $P_1 = A(F-H), P_2 = (A+B)H, P_3 = (C+D)E, P_4 = D(G-E), P_5 = (A+D)(E+H), P_6 = (B-D)(G+H), P_7 = (A-C)(E+F)$.

Q 3.1. Exprimer $AE+BG, AF+BH, CE+DG$ et $CF+DH$ avec les P_i en utilisant uniquement l'addition et la soustratcion.

Q 3.2. En déduire un algorithme pour calculer le produit de matrices et évaluer sa complexité.

Exercice 6 : Enveloppe convexe

Proposer un algorithme de type "diviser pour régner" pour calculer l'enveloppe convexe d'un ensemble de points 2D. Quelle est sa complexité?

Exercice 7 : La ligne des toits

On cherche à dessiner en 2D la ligne des toits d'un ensemble d'immeubles rectangulaires vus par une personne qui les regarde de face. Plus précisément, on a n rectangles alignés sur l'axe des abscisses dans le premier quadrant, le rectangle $(G_i, 0), (G_i, H_i), (D_i, H_i), (D_i, 0)$ étant donné par le triplet (G_i, H_i, D_i) $(0 \leq G_i < D_i, 0 < H_i)$. L'entrée du problème est donc:

n le nombre d'immeubles

Im la liste des triplets (G_i, H_i, D_i).

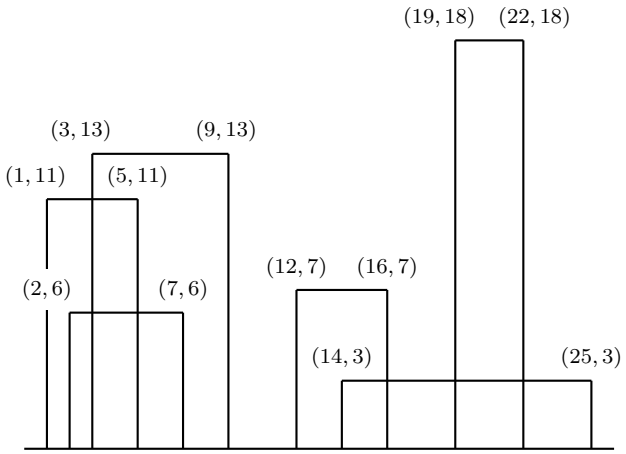


Figure A

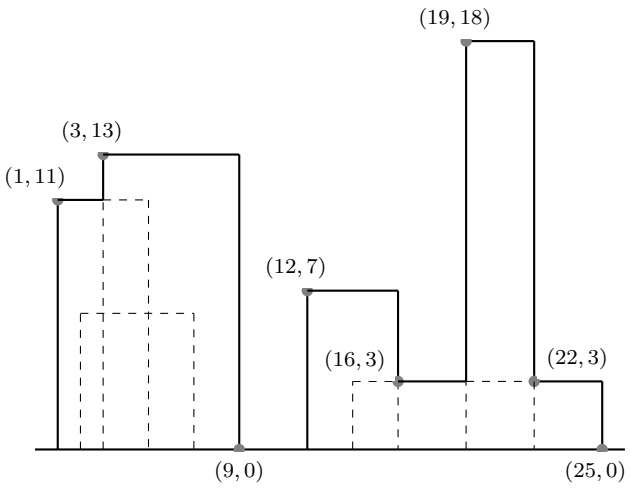


Figure B

La sortie sera une suite ordonnée par abscisses croissantes des points qui permettent de tracer la ligne des toits. Par exemple, pour 6 immeubles donnés par la liste (1,11,5), (2,6,7), (3,13,9), (12,7,16), (14,3,25), (19,18,22) (voir figure A), la sortie sera la liste (1,11), (3,13), (9,0), (12,7), (16,3), (19,18), (22,3), (25,0) (voir figure B).

Remarque: une liste de points $(x_1, y_1), (x_2, y_2), \ldots, (x_p, y_p)$ triée par abscisses croissantes correspond à la ligne tracée en reliant dans l'ordre les points suivants:
$(x_1, 0), (x_1, y_1), (x_2, y_1), (x_2, y_2), (x_3, y_2) \ldots (x_i, y_i), (x_{i+1}, y_i), (x_{i+1}, y_{i+1}) \ldots (x_p, y_{p-1}), (x_p, y_p)$

On pourra supposer que pour une telle liste, $y_1 > 0$, $y_p = 0$ et que deux points consécutifs ont des ordonnées différentes.

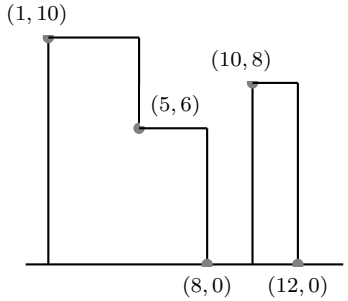
On va appliquer une méthode de type diviser pour régner pour construire la ligne des toits des immeubles. Pour cela, on va d'abord réaliser la fusion de deux lignes de toits.

Q 1. Fusion Le problème est donc défini par:

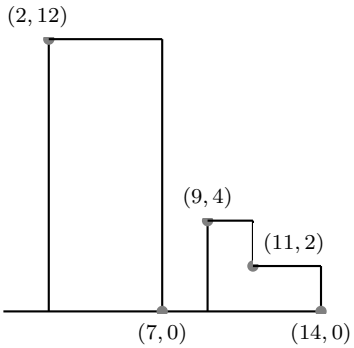
Entrée: deux lignes de toits données par la suite ordonnée par abscisses croissantes des points qui permettent de les tracer

Sortie: la ligne de toits obtenue par fusion

Par exemple, soit la première ligne donnée par (1,10), (5,6), (8,0), (10,8), (12,0) et la seconde par (2,12), (7,0), (9,4), (11,2), (14,0), comme dans les figures ci-dessous:

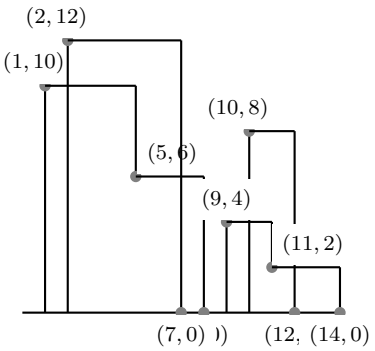


Ligne 1

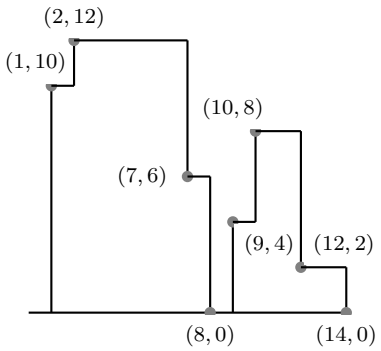


Ligne 2

La fusion des lignes des toits est (1,10), (2,12), (7,6), (8,0), (9,4), (10,8), (12,2), (14,0) :



Lignes 1 et 2



Fusion

Q 1.1. Si la première ligne de toits est donnée par (1,4), (6,0), (9,7), (12,0) et la seconde par (3,1), (5,5), (10,2), (15,0), quelle est la ligne de toits obtenue par fusion?

Q 1.2. Proposez un algorithme en $O(n)$ pour fusionner deux listes de toits, n étant le max des deux longueurs des lignes de toits. Vous pouvez supposer que les listes et les points sont représentés comme vous le souhaitez.

Q 2. Proposez un algorithme en $O(n \log n)$ de type "Diviser pour régner" pour résoudre le problème initial, i.e. sortir à partir de la liste des immeubles représentés par leurs triplets, la ligne de toits correspondante. Vous pouvez supposer que les listes, points et triplets sont implémentés comme vous le souhaitez. Vous pouvez supposer avoir répondu à la question précédente et donc disposer d'un algorithme en $O(n)$ pour la fusion de deux lignes de toits.