N° d'anonymat:

Année Universitaire 2016-2017
Session 1 de printemps

**Parcours/Étape:** L3 Informatique **Code UE:** 4TIN602U
**Épreuve:** Techniques Algorithmiques et Programmation
**Date:** 24/04/2017 **Heure:** 9h00 **Durée:** 1h30
Documents: une seule feuille A4 recto-verso autorisée.
Épreuve de M. Cyril Gavoille

université de BORDEAUX

**Collège Sciences et Technologie**

# Questions de cours

Il s'agit de donner un exemple de problème qui peut-être résolu de manière efficace par un algorithme résultant de la technique « diviser pour régner ». Plus précisément :

**Question 1.** *Donner une définition précise du problème choisi.*

**Question 2.** *Donner le principe de l'algorithme.*

**Question 3.** *Donner la complexité de l'algorithme de la question précédente.*

# Minimum local 1D

On considère un tableau `A` de $n \geqslant 1$ réels supposés tous différents. On dira que l'élément `A[i]` est un *minimum local* s'il est plus petit que ses éléments adjacents dans `A`, c'est-à-dire s'il est plus petit que l'élément précédent `A[i-1]` (s'il existe) et suivant `A[i+1]` (s'il existe). Notez bien que les extrémités du tableau, `A[0]` ou `A[n-1]`, peuvent être minimum local.

Le problème consiste à déterminer le plus rapidement possible la position d'un minimum local, c'est-à-dire son indice dans le tableau. Une façon visuelle de se représenter le problème est de voir le tableau comme le profile d'un parcours long de $n$ kilomètres (chemin de randonné, course cycliste, ...), et où `A[i]` donne l'altitude au `i`-ème kilomètre. Un minimum local est tout simplement une des vallées, ou s'il n'y en a pas le point de départ ou d'arrivée.

**Question 4.** *Donner deux exemples de tableaux `A1` et `A2` chacun de $n = 5$ éléments et qui possèdent respectivement un (pour `A1`) et deux (pour `A2`) minimum locaux et qui ne soient pas une des extrémités. Existe-t-il un tableau sans minimum local ? Justifier.*

**Question 5.** *Donner un algorithme naïf permettant de donner l'indice `i` d'un minimum local pour `A` (sous la forme d'une fonction en langage C ou pas).*

**Question 6.** *Est-ce que votre algorithme naïf pourrait être qualifié d'algorithme exhaustif ? Pourquoi ?*

**Question 7.** *Donner la complexité en temps de votre algorithme naïf.*

Il s'agit maintenant de concevoir une version améliorée de l'algorithme naïf.

**Question 8.** *En utilisant la technique « diviser pour régner », donner un algorithme récursif le plus efficace possible donnant l'indice d'un minimum local (en C ou pas).*

Soit $T(n)$ la complexité en temps de l'algorithme de la question précédente lorsqu'il est appliqué à un tableau de taille $n$.

**Question 9.** *Tout en justifiant, donner l'équation de récurrence que doit vérifier $T(n)$, puis la valeur asymptotique de $T(n)$.*

# Minimum local 2D

On généralise le problème précédent aux tableaux à deux dimensions. On considère maintenant un tableau `G` de $n \times n$ réels supposés tous différents. Le tableau `G` est vu comme une grille carrée, `G[i][j]` donnant par exemple l'altitude à la position $(i, j)$. Comme précédemment on dira que l'élément `G[i][j]` est un minimum local s'il est plus petit que ses éléments adjacents dans `G`, c'est-à-dire s'il est plus petit que chacun des (au plus) quatre éléments voisins de la grille qui sont (s'ils existent) : `G[i][j-1]`, `G[i][j+1]`, `G[i-1][j]` et `G[i+1][j]`. Le *bord* de la grille représente les positions $(i, j)$ avec $i \in \{0, n-1\}$ ou $j \in \{0, n-1\}$. C'est aussi tous les éléments qui n'ont pas quatre voisins.

On souhaite calculer la position $(i, j)$ d'un minimum local dans `G`. Pour commencer on considère l'algorithme suivant, dit du gradient, qui en passant permet de démontrer l'existence d'un minimum local.

> *« On part d'une position quelconque de la grille, comme par exemple le centre de la grille. Tant que l'élément de la position courante n'est pas un minimum local on se déplace vers un élément voisin inférieur. »*

La suite des éléments rencontrés décroît strictement ce qui interdit de passer deux fois par le même élément. L'algorithme du gradient s'arrête donc sur un minimum local puisque la grille possède un nombre fini d'éléments.

**Question 10.** *Quelle est, en fonction de $n$, la longueur maximum de la suite d'éléments construite par l'algorithme du gradient ? [Pensez à utiliser un exemple.] En déduire la complexité en temps de l'algorithme du gradient.*

On souhaite calculer encore plus efficacement la position d'un minimum local. Pour ce faire, on va supposer que l'on cherche une position qui n'est pas sur un bord, quitte à ajouter artificiellement un bord tout autour de la grille initiale avec des valeurs plus grandes que toutes les autres. La *croix médiane* de la grille représente les positions $(i, j)$ telles $i = \lfloor n/2 \rfloor$ ou $j = \lfloor n/2 \rfloor$. Lorsqu'on supprime de la grille la croix médiane et le bord on obtient quatre *quadrants*, des sous-grilles de taille $p \times p$ (ou $(p+1) \times (p+1)$ en ajoutant leur bord) avec $p \leqslant \lceil (n-3)/2 \rceil$. Notez qu'un élément de la grille ne peut pas être à la fois dans un quadrant et sur la croix médiane ou le bord.

L'algorithme proposé est basé sur l'observation suivante. Soit $s =$ `G[i][j]` l'élément minimum parmi tous ceux de la croix médiane et du bord. Si $s$ n'est pas un minimum local, alors la suite des éléments construits par l'algorithme du gradient depuis $s$ est entièrement incluse dans un seul quadrant, sauf l'élément $s$ lui-même qui ne fait partie d'aucun quadrant.

**Question 11.** *Expliquer pourquoi cette propriété est vraie.*

**Question 12.** *De cette propriété en déduire un algorithme utilisant la méthode « diviser pour régner ». Donner les détails de cet algorithme. [Tester votre algorithme sur une grille $5 \times 5$ par exemple.]*

Soit $T(n)$ la complexité en temps de l'algorithme de la question précédente lorsqu'il est appliqué à une grille de taille $n \times n$.

**Question 13.** *Tout en justifiant, donner l'équation de récurrence que doit vérifier $T(n)$, puis la valeur asymptotique de $T(n)$.*

On considère le problème de déterminer la position du minimum absolu dans une grille, c'est-à-dire la position `(i,j)` telle que `G[i][j]` est le plus petit élément de la grille. Notez que le minimum absolu

est nécessairement un minimum local.

**Question 14.** *Montrer en justifiant que la complexité en temps du problème du minimum absolu dans une grille $n \times n$ est en $\Omega(n^2)$.*

**FIN.**