

Recherche de Point Selle Strict

Le problème traite d'une matrice A , carrée à termes entiers: $A \in M_n(\mathbb{Z})$, de terme générique $a_{i,j}$ (i : ligne, j :colonne). On supposera toujours $n \geq 2$. Une entrée (i,j) de A sera qualifiée de Point Selle Strict (abrégé : PSS) pour A quand on aura

$$\forall k \neq j \quad a_{i,j} > a_{i,k} \quad \quad \quad \forall k \neq i \quad a_{k,j} > a_{i,j}$$

Un PSS est donc le plus grand de sa ligne et le plus petit de sa colonne

Nos objectifs vont être de trouver des algorithmes partant d'une matrice A et disant

- ou bien : A n'a pas de PSS
- ou bien : (i_0, j_0) est un PSS

De plus on souhaite que ces algorithmes fassent aussi peu de comparaisons que possible

Reprécisons le vocabulaire: on parlera du terme $a_{i,j}$ et de l'entrée (i,j)

Tous les raisonnements de comparaisons sont susceptibles d'être peu clair et d'être non compris par votre lecteur. Les explications devront donc toutes être étayées de petits dessins montrant les lignes et les colonnes.
Faites également attention à ne pas mélanger \leq et $<$ ou \geq et $>$

A Généralités

- 1 donner des exemples de matrices P et Q avec $n=3$
 - a P qui n'a pas de PSS
 - b Q qui a un PSS
- 2 prouver qu'un éventuel PSS est unique

B Premiers algorithmes

- 1 une entrée (λ, μ) est fixée
 - a combien (au pire) de comparaisons faudra-t-il pour dire si (λ, μ) est un PSS?
 - b combien (au pire) de comparaisons coûte l'algorithme (A1) qui consiste à utiliser la méthode précédente pour chacun des (λ, μ) ?
 - 2
 - a donner un algorithme (effectuant le moins de comparaisons possible) qui retourne l'entrée du plus grand terme d'une colonne si celle-ci est unique sinon "perdu"
 - b la matrice A a été initialisé par
- ```
let taille=10;;let A = make_matrix taille taille 0;;
```

A est donc du type `int vect vect` et pour `i` et `j` entre 0 et `taille-1` on accède au terme `ai j` par `A.(i).(j)` et on peut le changer en `nouveau` par `A.(i).(j)<- nouveau`

Donner le code caml de l'algorithme de la question précédente; on fournira aussi le code de la création d'une matrice de test et un exemple d'exécution

**3** prouver qu'un algorithme prenant en variable une colonne de taille `n` et n'effectuant que  $(n-2)$  comparaisons ne peut pas toujours retourner le plus grand

**4** donner un algorithme (A2) résolvant le problème de recherche d'un PSS plus efficacement que (A1)

## C Un autre algorithme

### 1 un lemme

On fixe deux entrées  $(i_0, j_0)$  et  $(i_1, j_1)$  sur des lignes et colonnes différentes; on veut savoir si l'une des deux correspond à un PSS

**a** utiliser le rectangle de diagonale  $((i_0, j_0), (i_1, j_1))$  pour éliminer  $(i_0, j_0)$  ou  $(i_1, j_1)$  avec 3 comparaisons au plus

**b** faites de même avec seulement 2 comparaisons

### 2 l'algorithme (A3)

Essayez d'utiliser le lemme précédent pour éliminer des entrées de A décrire un nouvel algorithme (A3) et déterminer sa complexité

Définitions:

Un ensemble D d'entrées de A sera dit "déterminant" si il est "Sans Menace" (abrégé : (SM)) et "Couvrant" (abrégé : (C)) où

(SM) si  $(i_0, j_0)$  et  $(i_1, j_1)$  sont dans D alors  $i_0 \neq i_1$  et  $j_0 \neq j_1$  : D est un ensemble de positions de tours du jeu d'échec qui ne se menacent pas

(C) si  $(l, c)$  est un PSS de A alors il existe  $(l, j_0)$  et  $(i_1, c)$  dans D : un PSS est menacé en ligne et en colonne

Enfin on note  $D_0$  la diagonale de A :  $D_0 = \{(k, k)_{1 \leq k \leq n}\}$

Il est conseillé de mettre en œuvre (A4) sur les exemples de la partie qui suit tout en effectuant les preuves

## D Un bon algorithme

**1** prouver que  $D_0$  est déterminant

Ci-dessous D est un ensemble déterminant

**2** donner un algorithme ne nécessitant que  $\frac{3n}{2}$  comparaisons qui trouve des entrées  $(i, j)$  du plus petit élément (sa valeur est nommée  $\alpha$ ) et du plus grand élément (sa valeur est nommée  $\beta$ ) parmi les `ai j` avec  $(i, j)$  dans D

**3** prouver que si  $(l, c)$  est un PSS et si  $x = a_{l c}$  alors

**a**  $\alpha \leq x \leq \beta$

**b**  $(\alpha = x)$  ou  $(\beta = x) \Rightarrow (l, c) \in D$

on note  $(i_0, j_0)$  et  $(i_1, j_1)$  de entrées telles que  $\alpha = a_{i_0, j_0}$  et  $\beta = a_{i_1, j_1}$  et on suppose, jusqu'à la question 7, que  $i_0 \neq i_1$  et  $j_0 \neq j_1$

4

- a prouver que  $(i_0, j_0)$  est le seul PSS possible en colonne  $j_0$
- b prouver que  $(i_1, j_1)$  est le seul PSS possible en ligne  $i_1$
- On garde les notations précédentes, on ajoute  $\gamma = a_{i_0 j_1}$  et

5 on suppose  $\gamma \leq \alpha \leq \beta$

- a prouver qu'un éventuel PSS en colonne  $j_1$  a une valeur inférieure à  $\alpha$
- b prouver qu'en ce cas il est dans D
- c en déduire qu'il n'existe pas
- d éliminer de même la ligne  $i_1$
- e prouver que  $D \setminus \{(i_1, j_1)\}$  est déterminant

6 on suppose  $\alpha \leq \beta \leq \gamma$  prouver que  $D \setminus \{(i_0, j_0)\}$  est déterminant

7 on suppose  $\alpha < \gamma < \beta$

- a prouver qu'il n'y a de PSS ni en colonne  $j_0$  ni en ligne  $i_1$
- b prouver que  $D \setminus \{(i_0, j_0), (i_1, j_1)\} \cup \{(i_0, j_1)\}$  est déterminant

8 déduire de ce qui précède un algorithme (A4) de détermination de PSS

On admettra (et on verra en cours d'année) qu'il existe des algorithmes permettant de maintenir le plus petit et le plus grand en complexité logarithmique: au prix de  $\log_2(p)$  comparaisons, on peut structurer un ensemble de  $p$  éléments de façon à connaître le plus petit et le plus grand de telle sorte que, si un élément arrive ou si un élément s'en va on se retrouve avec  $(p+1)$  ou  $(p-1)$  éléments dont on connaît toujours le plus petit et le plus grand, cette mise à jour ayant un coût majoré par  $2\log_2(p)$  comparaisons

9 quelle est la complexité de (A4)? La réponse semble paradoxale, essayez d'expliquer qualitativement

10 sans détailler les diverses fonctions caml nécessaires à la programmation de cet algorithme, donner les grandes lignes d'un tel programme. En particulier on vient ci-dessus d'admettre l'existence d'un miraculeux algorithme de maintenance, vous supposerez donc que certaines fonctions "miracle" sont disponibles. Il vous est demandé de donner le type de ces fonctions et de préciser leur fonctionnement

Exemple: l'une des fonction sera `miracle_cree : int*int list -> int chose`; quand on lui fournit la liste des entrées de D, elle retourne une `int chose` selon ce qui a été admis ci-dessus

## E deux exemples

Détailler le fonctionnement de (A4) sur les deux exemples qui suivent

$$A = \begin{pmatrix} 0 & 85 & 24 & 68 & 57 & 73 \\ 56 & 3 & 6 & 90 & 93 & 76 \\ 29 & 95 & 14 & 22 & 41 & 0 \\ 76 & 96 & 21 & 23 & 64 & 28 \\ 86 & 74 & 15 & 53 & 11 & 43 \\ 17 & 16 & 11 & 96 & 86 & 93 \end{pmatrix} \quad B = \begin{pmatrix} 63 & 43 & 51 & 31 & 27 & 11 \\ 22 & 14 & 18 & 27 & 12 & 25 \\ 30 & 19 & 98 & 43 & 19 & 20 \\ 43 & 51 & 70 & 40 & 30 & 15 \\ 82 & 18 & 50 & 32 & 17 & 14 \\ 1 & 29 & 30 & 34 & 28 & 13 \end{pmatrix}$$