1

Parcours de tableaux

« *Ordre, Permutations, Jeux.* »

1.1 Jeu d'échecs

Sur un échiquier, on représentera chaque case par ses coordonnées (i, j), la case en bas à gauche étant de coordonnées $(0, 0)$. Sur un tel échiquier, en un coup, un cavalier peut se déplacer de la case (i, j) vers celles d'entre les 8 positions suivantes qui correspondent effectivement à une case de l'échiquier (abscisse et ordonnée comprises entre 0 et 7) : $(i-2, j+1)$, $(i-1, j+2)$, $(i+1, j+2)$, $(i+2, j+1)$, $(i+2, j-1)$, $(i+1, j-2)$, $(i-1, j-2)$ et $(i-2, j-1)$.

Question 1

Écrire une fonction OCaml qui donne toutes les cases accessibles en p coups au plus à partir d'une case (i_0, j_0).

Question 2

Écrire une fonction OCaml qui indique si toutes les cases sont accessibles à partir d'une case (i_0, j_0) donnée, et si oui, quel est le plus petit nombre de coups permettant d'atteindre à partir de cette case n'importe quelle autre case de l'échiquier.

CORRIGÉ

Question 1

Choisissions déjà la structure de données. Définissions un type

```
type case = int * int
```

et donnons-nous une fonction `est_valide (c : case) : bool` qui renvoie `true` si c est bien une case de l'échiquier (i.e $0 \leq$ `fst c`, `snd c` ≤ 7), et `false` sinon. Réalisons tout de suite que quel que soit le nombre de coups, on ne pourra jamais atteindre plus de 64 cases et définissons des tableaux

```
let coups = Array.make_matrix nombremax 64 (0,0)
let ncases = Array.make nombremax 0
```

tel que `coups[i,j]` désigne la $j$ème des cases atteignables en exactement i coups et `ncases[i]` le nombre de cases atteignables en exactement i coups - et non atteignables en moins de coups. La seule difficulté de la fonction et de penser à « faire le ménage »en vérifiant, à chaque fois qu'on ajoute une case, si elle n'a pas déjà été atteinte. Il est à noter qu'il suffit de vérifier ceci pour des valeurs de i (numéros des coups) de même parité que le numéro courant : si on part d'une case blanche, en un nombre par de coups, on sera forcément sur une case blanche, et en un nombre impair sur une case noire.

```
(* Retourne la k-ième case atteignable a partir de
  la case c selon l'ordre arbitraire de l'énoncé *)
let case_suivante (c:case) (k:int) : case =
  assert (1 <= k && k <= 8);
  let i, j = c in
  let positions = [|
    (i-2, j+1); (i-1, j+2); (i+1, j+2); (i+2, j+1);
    (i+2, j-1); (i+1, j-2); (i-1, j-2); (i-2, j-1)
  |] in
  positions.(k-1)

(* Vérifie si la case c a déjà été atteinte, On est à
   l'étape pcour, et on cherche c parmi les cases
   atteintes à un coup i de même parité que pcour,
   pcour compris *)
let existe_deja (c:case) (pcour:int) : bool =
```

Question 2

Il n'y a pas grand chose à modifier : il suffit de remarquer qu'on a forcément atteint toutes les cases atteignables dès que l'ajout d'un nouveau coup n'aporte rien. Il suffit de modifier très légèrement la procédure précédente pour remplacer la boucle « for »sur la variable `pcourant` par une boucle « while »où l'on s'arrête dès que `ncases[pcourant]` vaut zéro. Il est alors facile de voir si toutes les cases sont atteintes (il faut simplement additionner les valeurs des `ncases[i]` pour voir si on trouve 64, surtout ne pas tester pour toute case si elle est dans le tableau `coups`, ceci serait trop long). Le lecteur pourra essayer d'évaluer le coût de ces fonctions. Sans réfléchir, on trouve quelque chose d'exponentiel en le nombre de coups, sauf si on se souvient que l'échiquier n'a que 64 cases !

⋆⋆⋆⋆⋆⋆⋆⋆

1.2 Médiane d'un tableau

On dispose d'un tableau A de n entiers distincts.

Question 1

Écrire une fonction OCaml `echange (A:int array) (i:int) (j:int) : unit` qui échange les éléments d'indices i et j du tableau A.

Question 2

Soient g et d deux entiers, $1 \leq g \leq d \leq n$. Posons $\alpha = A[g]$. On désire effectuer une permutation des éléments de A d'indice compris entre g et d, qui soit telle qu'après la permutation il existe un entier *pivot*, ($g \leq pivot \leq d$) vérifiant :

- $A[pivot] = \alpha$,
- pour tout i conmpris entre g et *pivot*, $A[i] \leq \alpha$,
- pour tout i conmpris entre $pivot + 1$ et d, $A[i] > \alpha$,
- les éléments de A d'indice strictement inférieur à g ou strictement supérieur à d restent inchangés.

Par exemple, si $n = 6$, si les éléments de A sont 5, 8, 7, 3, 9, 15, si $g = 2$ et $d = 5$, les éléments de A après la permutation seront 5, 3, 7, 8, 9, 15 ou 5, 7, 3, 8, 15, 9, ou... (il n'y a pas unicité des permutations possibles), et *pivot* sera égal à 4. Écrire une fonction OCaml qui effectue la permutation et donne la valeur de *pivot* sans utiliser un autre tableau que A. Combien d'affectations (c'est à dire d'instructions « `<-` ») et de tests nécessite-t-elle ?

Question 3

On appelle *médiane* de A un couple (i, α) tel que $1 \leq i \leq n$, $A[i] = \alpha$, et $E(n/2)$ éléments de A sont inférieurs strictement à α ($E(x)$ est la partie entière de x). Proposer une fonction de calcul de la médiane de A utilisant la fonction de la question 2.

——— CORRIGÉ ———

Question 1

La question 1 est élémentaire.

Question 2

Pour la fonction donnant la permutation (c'est une fonction dite de *partition*), on maintient deux indices i et j tels qu'à tout moment les éléments d'indice compris entre g et i sont tous inférieurs ou égaux à α, tandis que les éléments d'indice compris entre j et d sont tous supérieurs ou égaux à α.

2

Jouer avec les mots

« *Reconnaissance, Cosntruction, Codage.* »

3

Stratégies gloutonnes

« *Le meilleur du moment, pour trouver le meilleur.* »

4

Arborescences

« *Un père, des fils. Une racine, des noeuds, des feuilles.* »

5

Graphes

« *Des arêtes, des sommets, des poids.* »

6

Géométrie et Images

« *Des représentations. Des figures. Des intersections. Des pavages.* »

7

Arithmétique et calculs numériques

« *Écrire. Calculer. Résoudre.* »

8

Vers la récursivité

« *Diviser pour régner. Faire moins pour faire plus.* »

A

Un peu d'Algèbre

A.1 Anneaux Booléens

F_2 désignera le corps à deux éléments $\mathbb{Z}/2\mathbb{Z}$. Un *anneau Booléen* est un anneau commutatif unitaire $(A, +, *)$ dans lequel, pour tout $x \in A, x * x = x$ (F_2 est un exemple d'anneau booléen).

Si E est un ensemble, on note $\mathcal{P}(E)$ l'ensemble des sous-ensembles finis de E et $\mathcal{AB}[E]$ l'ensemble des fonctions de $\mathcal{P}(E)$ dans F_2 qui sont nulles sauf pour un nombre fini d'éléments de $\mathcal{P}(E)$. $\mathcal{AB}[E]$ est muni de l'addition et de la multiplication : $(f+g)(x) = f(x) + g(x)$ et $(fg)(x) = f(x)g(x)$. $\mathcal{AB}[E]$ muni de ces opérations est aussi un anneau booléen : c'est l'anneau booléen *engendré* par E.

Dans tout le problème, E sera supposé fini et de cardinal n.