Charge de cavalerie

Épreuve pratique d'algorithmique et de programmation
Concours commun des Écoles normales supérieures

Durée de l'épreuve : 3 heures 30 minutes

Juin/Juillet 2023

ATTENTION !

N'oubliez en aucun cas de recopier votre u_0
à l'emplacement prévu sur votre fiche réponse

Important.

Il vous a été donné un numéro u_0 qui servira d'entrée à vos programmes. Les réponses attendues sont généralement courtes et doivent être données sur la fiche réponse fournie à la fin du sujet. À la fin du sujet, vous trouverez en fait deux fiches réponses. La première est un exemple des réponses attendues pour un $\widetilde{u_0}$ particulier (précisé sur cette même fiche et que nous notons avec un tilde pour éviter toute confusion !). Cette fiche est destinée à vous aider à vérifier le résultat de vos programmes en les testant avec $\widetilde{u_0}$ au lieu de u_0. Vous indiquerez vos réponses (correspondant à votre u_0) sur la seconde et vous la remettrez à l'examinateur à la fin de l'épreuve.

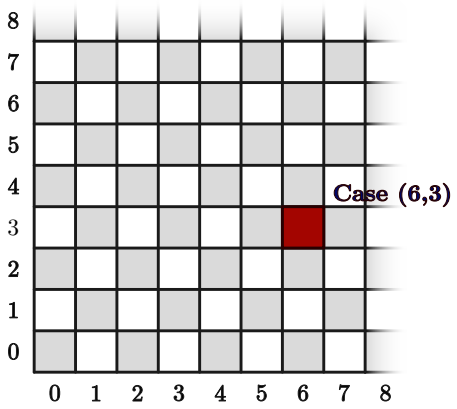
En ce qui concerne la partie orale de l'examen, lorsque la description d'un algorithme est demandée, vous devez présenter son fonctionnement de façon schématique, courte et précise. Vous ne devez en aucun cas recopier le code de vos procédures !
Quand on demande la complexité en temps ou en mémoire d'un algorithme en fonction d'un paramètre n, on demande l'ordre de grandeur en fonction du paramètre, par exemple : $O(n^2)$, $O(n \log n)$,...
Il est recommandé de commencer par lancer vos programmes sur de petites valeurs des paramètres et de ***tester vos programmes sur des petits exemples que vous aurez résolus préalablement à la main ou bien à l'aide de la fiche réponse type fournie en annexe***. Enfin, il est recommandé de lire l'intégralité du sujet avant de commencer afin d'effectuer les bons choix de structures de données dès le début.

1 Préliminaires

1.1 Échecs et cavaliers

Les échecs sont un jeu de stratégie opposant deux joueurs qui contrôlent respectivement des pièces noires et blanches. Aucune connaissance préalable sur le jeu d'échecs n'est requise. Nous précisons ici quelques-unes des règles du jeu, qui seront les seules importantes dans ce sujet.

Le jeu d'échecs se joue sur un plateau appelé l'échiquier, qui est une grille carrée de 8 cases de côté. Ces cases sont alternativement noires et blanches, mais leur couleur n'aura pas d'importance dans ce sujet. On considèrera ici plus généralement des échiquiers de taille $n \times n$, c'est-à-dire des grilles carrées de n cases de côté, où $n \in \mathbb{N}^\star$. On munit l'échiquier d'un système de coordonnées : à chaque case est associé un couple d'entiers $(i, j) \in [\![0, n-1]\!]^2$, appelé sa position, désignant respectivement sa colonne (abscisse) et sa rangée (ordonnée) [1], comme illustré dans l'exemple suivant où l'on a marqué la case $(6, 3)$.

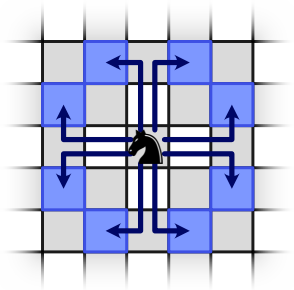


Sur un échiquier peuvent être placées et déplacées des pièces, qui peuvent être noires ou blanches. Le cavalier est l'une de ces pièces, symbolisée par ♘ (pour un cavalier blanc) ou ♞ (pour un cavalier noir). Son mouvement suit une forme de "L" : lorsque l'on joue un coup avec un cavalier, on peut le déplacer

— de 2 cases horizontalement et 1 case verticalement,

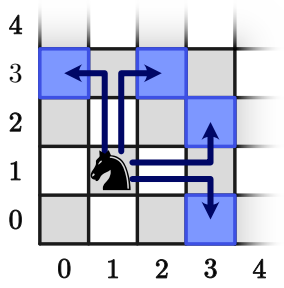
— ou bien de 2 cases verticalement et 1 case horizontalement.

Il en résulte en général 8 mouvements possibles pour un cavalier :

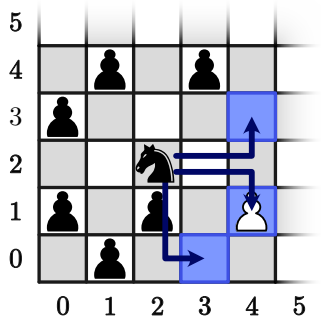


Le cavalier ne peut cependant pas sortir de l'échiquier, ce qui limite ses mouvements lorsqu'il est trop près du bord. Par exemple dans le cas suivant, le cavalier, trop proche du coin inférieur gauche de l'échiquier, n'aura que 4 mouvements autorisés.

1. La colonne est habituellement plutôt désignée par une lettre, mais il sera plus pratique pour nos programmes d'utiliser un entier à la place.



La présence ou l'absence d'autres pièces sur les cases au dessus desquelles passe le cavalier ne gêne pas son mouvement. En revanche, la case d'arrivée doit, soit être vide, soit contenir une pièce de la couleur opposée à celle du cavalier. Dans ce deuxième cas, on dit que la pièce en question est capturée par le cavalier à l'issue du mouvement.
Par exemple dans le cas suivant, le cavalier noir ♞ n'a que 3 mouvements valides, car 5 des 8 cases d'arrivée *a priori* possibles sont occupées par des pièces noires ♟. La présence d'une pièce sur la case $(2, 1)$ ne bloque en revanche pas son mouvement. L'un des 3 mouvements conduit à la capture d'une pièce blanche ♙ : celui qui atteint la case $(4, 1)$.



1.2 Graphes, chemins, flots

On entendra dans ce sujet par "graphe" un graphe orienté pondéré. Un tel graphe $\mathcal{G}$ est un triplet $\mathcal{G} = (\mathcal{S}, \mathcal{A}, \mathsf{poids})$. $\mathcal{S}$ est l'ensemble des sommets de $\mathcal{G}$. $\mathcal{A} \subseteq \mathcal{S}^2$ est celui de ses arcs : un arc $(u, v) \in \mathcal{A}$ est dit sortant de u et entrant dans v. On ajoute la restriction que $\mathcal{A}$ ne peut jamais contenir à la fois (u, v) et (v, u) : $\forall u, v \in \mathcal{S}.\ (u, v) \in \mathcal{A} \Rightarrow (v, u) \notin \mathcal{A}$.
Enfin $\mathsf{poids} : \mathcal{A} \mapsto \mathbb{Z}$ est une fonction attribuant à chaque arc un entier appelé son poids.
Pour un sommet $u \in \mathcal{S}$ on notera $\mathsf{succ}_{\mathcal{G}}(u)$ l'ensemble des successeurs de u dans $\mathcal{G}$:

$$\mathsf{succ}_{\mathcal{G}}(u) = \{v \in \mathcal{S} \mid (u, v) \in \mathcal{A}\}.$$

Un chemin C sur le graphe $\mathcal{G}$ est une séquence de $l + 1 \in \mathbb{N}$ sommets reliés par des arcs : $\mathsf{C} = [u_0, \ldots, u_l]$, pour des $u_i \in \mathcal{S}$ tels que $\forall i \in [\![0, l-1]\!].\ (u_i, u_{i+1}) \in \mathcal{A}$. u_0 est le départ de C, u_l son arrivée. Le poids du chemin C est la somme des poids de ses arcs : $\mathsf{poids}(\mathsf{C}) = \sum_{0 \leq i \leq l-1} \mathsf{poids}(u_i, u_i + 1)$.

Considérons un graphe $\mathcal{G} = (\mathcal{S}, \mathcal{A}, \mathsf{poids})$, contenant deux sommets particuliers $s, t \in \mathcal{S}$, tels que s ne possède que des arcs sortants, et t que des arcs entrants, c'est-à-dire $\forall u \in \mathcal{S}.\ (u, s) \notin \mathcal{A} \wedge (t, u) \notin \mathcal{A}$. On appellera s une source et t un puits. Soit $f \in \mathbb{N}$ un entier. On appelle flot de valeur f de s vers t tout sous ensemble $\mathsf{F} \subseteq \mathcal{A}$ tel que

— F contient f arcs sortants de s : $\big|\{(x, y) \in \mathsf{F} \mid x = s\}\big| = f$
— F contient f arcs entrants dans t : $\big|\{(x, y) \in \mathsf{F} \mid y = t\}\big| = f$

— pour tout autre sommet $u \in \mathcal{S}$, F contient autant d'arcs entrants dans u que sortants de u :

$$\forall u \in \mathcal{S} \setminus \{s,t\}.\ \big|\{(x,y) \in \mathsf{F} \mid x = u\}\big| = \big|\{(x,y) \in \mathsf{F} \mid y = u\}\big|$$

On note alors $\mathsf{valeur}(\mathsf{F}) = f$. On appelle le poids du flot F la somme des poids de ses arcs :

$$\mathsf{poids}(\mathsf{F}) = \sum_{(u,v)\in\mathsf{F}} \mathsf{poids}(u,v)$$

2 Génération de données

Pour commencer, nous allons définir quelques suites générant des données pseudo-aléatoires qui serviront au test numérique des fonctions des parties suivantes.
On rappelle que, pour tous $a, b \in \mathbb{Z}$ tels que $b \neq 0$, $a \bmod b$ désigne le reste de la division euclidienne de a par b, c'est-à-dire le plus petit entier naturel r tel qu'il existe un entier q vérifiant $a = b \times q + r$.
De plus, pour une liste l et un élément x, on note $x::l$ la liste obtenue en ajoutant x au début de l. $\mathsf{longueur}(l)$ désigne le nombre d'éléments de l, et `[]` la liste vide. Par exemple, si l est la liste $[1,2,2]$ alors $3::l$ désigne une nouvelle liste $[3,1,2,2]$, et $\mathsf{longueur}(l) = 3$.

2.1 Génération de nombres pseudo-aléatoires

Étant donné u_0, on définit par récurrence :

$$\forall i \in \mathbb{N}.\ u_{i+1} = (989\,909 \times u_i) \bmod 1\,212\,149$$

Question 1 *Écrire un programme qui calcule la suite u, et en donner les valeurs suivantes :*

a) $u_1 \bmod 1\,000$ ***b)*** $u_{255} \bmod 1\,000$ ***c)*** $u_{2\,023} \bmod 1\,000$

d) $u_{54\,321} \bmod 1\,000$

Indication. Il pourra être judicieux, afin d'obtenir de meilleures performances dans les questions suivantes, de précalculer les valeurs de u_n jusqu'à un n assez grand : on pourra les calculer au début du programme, et les mémoriser dans un tableau, pour ne pas devoir refaire le calcul à chaque fois.

2.2 Génération de positions sur un échiquier

On définit, pour tous $n, p, i \in \mathbb{N}$ tels que $n \neq 0$ le couple d'entiers dans $[\![0, n-1]\!]$ suivant :

$$v_{n,p}(i) = ((9\,029 \times u_{p+2\times i}) \bmod n,\ (9\,029 \times u_{p+2\times i+1}) \bmod n)$$

Pour tous $n, p \in \mathbb{N}$ tels que $n \neq 0$ et $p \leq n^2$, on construit une liste $\mathsf{Pos}_{n,p}$ de longueur p contenant les $v_{n,p}(i)$ pour $i = 1, 2, 3, \ldots$, en éliminant les doublons. Plus précisément, $\mathsf{Pos}_{n,p}$ est la valeur calculée par l'algorithme suivant :

```
P := []
i := 1
tant que longueur(P) < p
    si v_{n,p}(i) ∉ P alors
        P := v_{n,p}(i)::P
    i := i + 1
renvoyer P
```

Algorithme 1 : Calcul de $\mathsf{Pos}_{n,p}$

$\mathsf{Pos}_{n,p}$ contient ainsi p couples deux à deux distincts d'entiers de $[\![0, n-1]\!]$, c'est-à-dire p positions distinctes sur un échiquier de taille $n \times n$.

Question 2 *Écrire un programme qui calcule* $\mathsf{Pos}_{n,p}$*, et donner, pour les valeurs de* n, p *suivantes, la somme* $\left(\sum_{(i,j)\in\mathsf{Pos}_{n,p}} i+j\right) \bmod 1\,000$ *:*

a) $n = 8$, $p = 30$ ***b)*** $n = 120$, $p = 1000$ ***c)*** $n = 1\,300$, $p = 200\,000$

Question à développer pendant l'oral 1 *Décrire le fonctionnement de votre programme : comment le test d'appartenance et l'ajout à la liste sont-ils implémentés ? Évaluer leur complexité en temps.*

2.3 Génération d'échiquiers avec scores

Pour tous entiers naturels non nuls n, p, i, j, on définit

$$w_{n,p}(i,j) = ((7\,681 \times u_p + i) \bmod n) \times n + ((8\,059 \times u_p + j) \bmod n) + 1$$

Question 3 *Écrire une fonction qui calcule* w*, et en donner les valeurs suivantes* mod 10 000 *:*

a) $w_{12,20}(6, 11)$ ***b)*** $w_{340,1\,000}(268, 64)$ ***c)*** $w_{1\,300,200\,000}(653, 208)$

On manipulera dans la suite du sujet des échiquiers $\mathcal{E}$ dont certaines cases contiendront des pièces associées à un entier non nul appelé score. On note $\mathsf{Pieces}(\mathcal{E})$ l'ensemble des coordonnées où sont placées des pièces sur $\mathcal{E}$. Pour une case $(i,j) \in \mathsf{Pieces}(\mathcal{E})$ contenant une pièce, $\mathsf{Score}_{\mathcal{E}}(i,j)$ désigne le score qui lui est associé. Pour une case $(i,j) \notin \mathsf{Pieces}(\mathcal{E})$ ne contenant pas de pièce, on considèrera que $\mathsf{Score}_{\mathcal{E}}(i,j) = 0$.

Pour tous $n, p, k \in \mathbb{N}$ tels que $n \neq 0$ et $p + k \leq n^2$, on notera $\mathcal{E}_{n,p,k}$ l'échiquier de taille $n \times n$ contenant $p + k$ pièces, aux coordonnées $\mathsf{Pos}_{n,p+k}$. Le score associé $\mathsf{Score}_{\mathcal{E}_{n,p,k}}(i,j)$ est l'entier positif $w_{n,p+k}(i,j)$ si (i,j) est l'un des p premiers éléments de $\mathsf{Pos}_{n,p+k}$, et l'entier négatif $-w_{n,p+k}(i,j)$ si c'est l'un des k derniers.

Question 4 *Écrire un programme qui calcule* $\mathcal{E}_{n,p,k}$*, et donner pour les valeurs suivantes de* n, p, k *la valeur absolue* $\left|\sum_{(i,j)\in\mathsf{Pieces}(\mathcal{E}_{n,p,k})} \mathsf{Score}_{\mathcal{E}_{n,p,k}}(i,j)\right| \bmod 10\,000$*.*

a) *n=8, p=23, k=10* ***b)*** *n=174, p=1 700, k=1 000* ***c)*** *n=3 400, p=4 300, k=5 000*

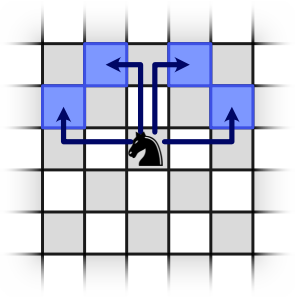
Question à développer pendant l'oral 2 *Décrire la structure de données que vous avez employée pour représenter* $\mathcal{E}$*. Évaluer pour cette structure de données la complexité en temps des opérations consistant à obtenir le score d'une position* (i,j)*, et à parcourir les positions de* $\mathsf{Pieces}(\mathcal{E})$*.*

3 Marche d'un cavalier

Dans cette partie, on s'intéresse à un échiquier $\mathcal{E}$ de taille $n \times n$, avec $n \in \mathbb{N}^\star$, sur lequel sont placées p pièces blanches, sur p cases distinctes (avec $p \in [\![0, n^2]\!]$).

Du point de vue d'un joueur, il peut être plus utile de capturer certaines pièces, stratégiquement placées, que d'autres. Pour modéliser ceci, on supposera donc qu'à chacune des p pièces blanches est associé un score positif distinct, qui représentera l'intérêt de capturer cette pièce.

Un cavalier noir ♞ se déplace sur l'échiquier. On se restreint au cas où le cavalier commence dans la case $(0,0)$ et ne se déplace jamais vers le bas. Il a donc à chaque coup au plus 4 mouvements possibles, représentés ci-dessous.



Pour une position (i, j) sur un échiquier de taille $n \times n$, on appelle $\mathsf{mouv}_n(i, j)$ la liste des (au plus 4) positions sur l'échiquier qu'un cavalier peut atteindre en un coup depuis (i, j) avec cette restriction.

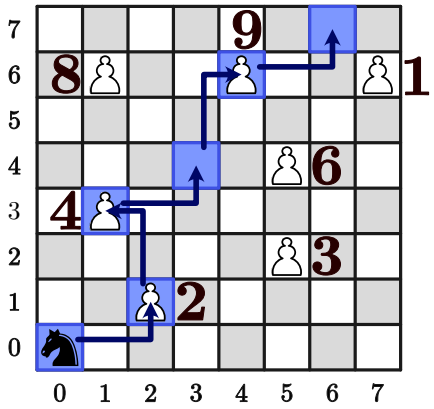
Question 5 *Écrire une fonction qui calcule* mouv. *Pour les valeurs suivantes de* n, p, i, *si l'on note* $\mathsf{mouv}_n(v_{n,p}(i)) = [(i_1, j_1), \ldots, (i_k, j_k)]$, *donner la valeur* $\left(\sum_{1 \leq m \leq k} (i_m^2 + j_m) \right) \bmod 1\,000$.

a) $n = 8, p = 10, i = 6$ ***b)*** $n = 120, p = 148, i = 54$ ***c)*** $n = 600, p = 2\,530, i = 567$

On appellera marche dans l'échiquier $\mathcal{E}$ une succession de mouvements faisables à partir de $(0, 0)$, c'est-à-dire toute séquence $\mathsf{M} = [(i_0, j_0), \ldots, (i_k, j_k)]$ de positions de $\mathcal{E}$ telle que $\forall m \in [\![0, k-1]\!]$. $(i_{m+1}, j_{m+1}) \in \mathsf{mouv}_n(i_m, j_m)$, et $i_0 = j_0 = 0$.
Le score de la marche M dans l'échiquier $\mathcal{E}$, $\mathsf{Score}_{\mathcal{E}}(\mathsf{M})$, désigne la somme des scores de toutes les cases parcourues :

$$\mathsf{Score}_{\mathcal{E}}(\mathsf{M}) = \sum_{1 \leq m \leq k} \mathsf{Score}_{\mathcal{E}}(i_m, j_m).$$

Notons que le score éventuel de la case de départ n'est pas compté.
La figure suivante donne un exemple d'une marche sur un échiquier $\mathcal{E}$ de taille 8×8 contenant 7 pièces dont les scores sont indiqués à côté des cases correspondantes (le score nul des cases vides n'est pas représenté).



Cette marche est $\mathsf{M} = [(0, 0), (2, 1), (1, 3), (3, 4), (4, 6), (6, 7)]$, et son score est $\mathsf{Score}_{\mathcal{E}}(\mathsf{M}) = 2 + 4 + 9 = 15$.

On s'intéresse à un cavalier qui cherche à maximiser le score de sa marche. On se pose le problème suivant : étant donné un échiquier $\mathcal{E}$ de taille $n \times n$, calculer un tableau des scores TS de taille $n \times n$ tel que pour tous i, j, $\mathsf{TS}[i][j]$ contient le score maximal des marches se terminant en (i, j) dans $\mathcal{E}$ (ou 0, s'il n'existe aucune telle marche).

3.1 Recherche exhaustive

On se propose tout d'abord de résoudre ce problème en considérant une par une toutes les marches. On initialise la table TS de façon que toutes ses cases contiennent 0. On énumère ensuite successivement toutes les marches partant de $(0,0)$. Pour chaque marche M se terminant sur une position (i,j), si $\mathsf{Score}_{\mathcal{E}}(\mathsf{M})$ est plus grand que la valeur courante de $\mathsf{TS}[i][j]$, on met à jour TS.

Question 6 *Calculer la table* TS *en énumérant exhaustivement toutes les marches, pour les échiquiers* $\mathcal{E}_{n,p,0}$ *avec les* n, p *suivants. Pour chacun, donner la somme* $\left(\sum_{0\leq i\leq n-1,\, 0\leq j\leq n-1} \mathsf{TS}[i][j]\right) \bmod 10\,000$.

a) $n = 8, p = 17$ ***b)*** $n = 15, p = 175$ ***c)*** $n = 16, p = 190$

Question à développer pendant l'oral 3 *Décrire le fonctionnement de votre programme pour le calcul exhaustif de* TS. *Évaluer sa complexité en temps.*

3.2 Algorithme de Bellman-Ford

La méthode de recherche exhaustive n'est pas très performante. Pour calculer la table TS plus efficacement, on se propose à présent d'utiliser l'algorithme de Bellman-Ford.
Il s'agit d'un algorithme permettant de calculer des chemins de poids maximal dans un graphe. On décrit ici son fonctionnement de façon générale sur un graphe quelconque, et il vous faudra ensuite déterminer comment l'appliquer au problème qui nous intéresse.

Considérons un graphe $\mathcal{G} = (\mathcal{S}, \mathcal{A}, \mathsf{poids})$ et un sommet $s \in \mathcal{S}$. L'algorithme calcule une table PC, telle que pour tout sommet $u \in \mathcal{S}$, $\mathsf{PC}[u]$ est le poids maximal d'un chemin de s à u dans $\mathcal{G}$, s'il en existe, ou une valeur par défaut $\perp$, s'il n'en existe pas.
L'algorithme est le suivant :

```
Initialiser PC à ⊥ : pour chaque u ∈ S  PC[u] := ⊥
PC[s] := 0
répéter jusqu'à ce que PC cesse de changer, ou au plus |S| fois
    pour chaque u ∈ S tel que PC[u] ≠ ⊥
        pour chaque v ∈ succ_G(u)
            si PC[v] = ⊥ ou PC[u] + poids(u, v) > PC[v] alors
                PC[v] := PC[u] + poids(u, v)
            sinon
                ne rien faire
si on a atteint |S| tours et PC a encore changé au tour |S| alors
    renvoyer erreur
sinon
    renvoyer PC
```

Algorithme 2 : Algorithme de Bellman-Ford

La boucle principale est exécutée au plus $|\mathcal{S}|$ fois. Si durant un tour de boucle PC ne change pas, alors il ne changera plus jamais : on s'arrête donc et on renvoie PC. Si on a atteint $|\mathcal{S}|$ tours sans que PC ne se stabilise, l'algorithme renvoie une erreur.

Remarque. Si l'algorithme renvoie erreur, cela signifie qu'il n'existe pas de chemin de poids maximal. En effet, on peut montrer que $\mathcal{G}$ contient alors un chemin revenant d'un sommet u à lui-même dont le poids soit strictement positif – un cycle positif. Un tel cycle peut être emprunté plusieurs fois pour produire des chemins de poids arbitrairement grands.
On ne demande pas de justifier que les graphes auxquels vous appliquerez l'algorithme par la suite ne contiennent pas de tels cycles.

Question 7 *En utilisant l'algorithme de Bellman-Ford, calculer la table* TS *des scores maximaux des marches depuis* $(0,0)$*, pour les échiquiers* $\mathcal{E}_{n,p,0}$ *avec les* n, p *suivants. Pour chacun, donner la valeur*

$$\left(\sum_{0\leq i\leq n-1,\, 0\leq j\leq n-1} \mathsf{TS}[i][j]\right) \bmod 10\,000.$$

a) $n = 8, p = 20$ ***b)*** $n = 50, p = 1\,600$ ***c)*** $n = 100, p = 2\,540$

Question à développer pendant l'oral 4 *Expliquer de quelle façon vous avez modélisé le problème posé sur l'échiquier en un problème de graphes, pour pouvoir lui appliquer l'algorithme de Bellman-Ford.*

Question à développer pendant l'oral 5 *Évaluer la complexité en temps de votre programme.*

L'algorithme de Bellman-Ford peut être étendu pour permettre de calculer, non seulement le poids maximal des chemins depuis s, mais aussi un chemin atteignant ce poids maximal.
On construit, en plus de la table PC, une table Pred : pour un sommet u, Pred$[u]$ désignera un sommet qui précède u dans un chemin de poids maximal de s à u. En utilisant Pred, on peut à l'issue de l'algorithme calculer un chemin de poids maximal vers chaque sommet.

On munit pour cette question les positions de l'ordre suivant : $(i_1, j_1) < (i_2, j_2)$ si et seulement si $i_1 < i_2$ ou ($i_1 = i_2$ et $j_1 < j_2$). On munit alors les marches d'un ordre, appelé ordre lexicographique inverse, défini comme suit : $\mathsf{M} < \mathsf{M}'$ lorsqu'il existe des entiers a, b, c et des positions $(m_i)_i$, $(m'_i)_i$, $(p_i)_i$ tels que $\mathsf{M} = [m_0, \ldots, m_a, p_0, \ldots, p_b]$, $\mathsf{M}' = [m'_0, \ldots, m'_c, p_0, \ldots, p_b]$, et $m_a < m'_c$.

Question 8 *Implémenter cette extension de l'algorithme de Bellman-Ford, pour calculer la marche* M *de* $(0,0)$ *à* $(n-1, n-1)$ *la plus petite pour l'ordre lexicographique inverse parmi celles de score maximal, dans l'échiquier* $\mathcal{E}_{n,p,0}$*, pour les valeurs de* n, p *suivantes. Donner, à chaque fois, la valeur* $\left(\sum_{(i,j)\in\mathsf{M}} i^2 + j\right) \bmod 10\,000$.

a) $n = 8, p = 20$ ***b)*** $n = 50, p = 1\,600$ ***c)*** $n = 100, p = 2\,540$

Question à développer pendant l'oral 6 *Décrire la façon dont votre programme calcule la table des prédécesseurs et la marche de score maximal la plus petite.*

3.3 Calcul efficace

Cette question est indépendante de la suite du sujet.

Question 9 *Écrire un programme implémentant un algorithme plus efficace pour le calcul du score maximal d'une marche de* $(0,0)$ *à* $(n-1, n-1)$ *sur l'échiquier* $\mathcal{E}_{n,p,0}$*. Donner ce score maximal* mod 10 000 *pour les valeurs de* n*,* p *suivantes.*

a) $n = 2\,000, p = 1\,000$ ***b)*** $n = 2\,500, p = 20\,000$ ***c)*** $n = 3\,000, p = 100\,000$

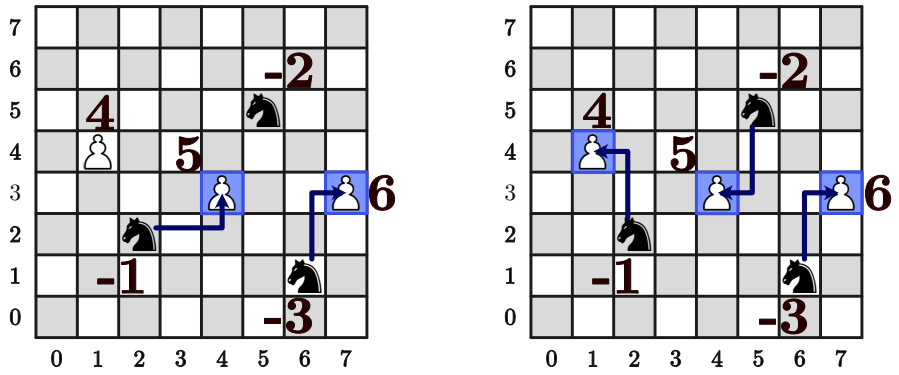
Question à développer pendant l'oral 7 *Décrire le fonctionnement de votre algorithme, et évaluer sa complexité en temps et en espace.*

4 Charge de cavalerie

On s'intéresse dans cette dernière partie à la question de coordonner l'attaque de multiples cavaliers pour maximiser le score obtenu.
Toujours sur un échiquier $\mathcal{E}$ de taille $n \times n$, on considère de nouveau p pièces blanches, sur des cases distinctes, avec chacune un score associé. On ajoute en plus k cavaliers noirs sur d'autres cases, qui ont eux aussi chacun un score associé. Ceci suppose bien sûr que $p + k \leq n^2$. Pour distinguer plus facilement les pièces blanches des cavaliers, on suppose que les scores des pièces blanches sont positifs, et ceux des cavaliers négatifs.

Contrairement à la partie précédente, on ne restreint plus le mouvement des cavaliers à aller vers le haut : ils ont donc de nouveau jusqu'à 8 déplacements possibles à chaque coup. On considèrera que capturer une pièce blanche de score v avec un cavalier de score v' rapporte un score $v + v'$.

Le problème que l'on va chercher à résoudre est le suivant : étant donné un tel $\mathcal{E}$, et un entier naturel $k' \leq k$, quel est le score maximal que l'on peut atteindre en faisant capturer une pièce en un coup chacun, à exactement k' cavaliers ? Deux cavaliers différents ne pouvant bien sûr pas capturer la même pièce.
Par exemple, pour un échiquier 8×8, avec $p = 3$ pièces et $k = 3$ cavaliers avec les positions et les valeurs de la figure suivante, on peut atteindre un score de 7 en jouant $k' = 2$ cavaliers, et 9 en jouant $k' = 3$ cavaliers.



4.1 Algorithme glouton

On souhaite tenter de résoudre ce problème en adoptant une stratégie gloutonne, consistant à choisir à chaque fois le coup le plus rentable parmi les coups disponibles, c'est-à-dire celui rapportant le score le plus élevé, jusqu'à avoir joué k' coups (on renverra une erreur si aucun coup n'est disponible alors que l'on n'a pas encore joué k' coups).

Question 10 *Écrire un programme qui implémente cette approche gloutonne. Donner la valeur absolue, mod 10 000, du score maximal atteignable sur les échiquiers $\mathcal{E}_{n,p,k}$ avec les nombres de captures k' suivants.*

a) $\mathcal{E}_{8,20,20}, k' = 10$

b) $\mathcal{E}_{600,40\,000,40\,000}$, $k' = 10\,000$

c) $\mathcal{E}_{1\,000,120\,000,98\,000}$, $k' = 20\,000$

Question à développer pendant l'oral 8 *Décrire le fonctionnement de votre programme. Évaluer sa complexité en temps.*

Question à développer pendant l'oral 9 *L'algorithme glouton renvoie-t-il toujours le score maximal que l'on souhaitait calculer ? Expliquer.*

4.2 Algorithme de Ford-Fulkerson

Pour finir, on se propose de résoudre le problème en passant de nouveau par l'algorithmique des graphes. Pour cela, on verra le problème de capture des pièces par les cavaliers comme une instance du problème du flot de poids maximal (*cf.* Partie 1.2 pour la définition d'un flot).
Ce problème est le suivant : étant donné un graphe $\mathcal{G}$ avec une source s et un puits t, et $f \in \mathbb{N}$, quel est le poids maximal d'un flot de valeur f de s à t dans $\mathcal{G}$ (s'il en existe) ?

Le problème de capture qui nous intéresse peut être vu comme une instance de ce problème. Pour un échiquier $\mathcal{E}$ de taille $n \times n$ contenant p pièces et k cavaliers, on considère le graphe $\mathcal{G}(\mathcal{E}) = (\mathcal{S}, \mathcal{A}, \mathsf{poids})$ suivant :

— Ses $k+p+2$ sommets sont les positions des cavaliers et pièces sur $\mathcal{E}$, plus une source s et un puits t.

$$\mathcal{S} = \mathsf{Pieces}(\mathcal{E}) \cup \{s, t\}$$

La valeur utilisée pour s et t importe peu, il faut simplement qu'on ne puisse pas les confondre avec les autres sommets.

— Des arcs relient chaque cavalier aux pièces qu'il peut capturer, la source s à chaque cavalier, et chaque pièce au puits t :

$$\begin{aligned}\mathcal{A} = &\left\{((i,j),(i',j')) \in (\mathcal{S}\setminus\{s,t\})^2 \;\middle|\; \mathsf{Score}_{\mathcal{E}}(i,j) < 0 \wedge \mathsf{Score}_{\mathcal{E}}(i',j') > 0 \wedge (i',j') \in \mathsf{captures}(i,j)\right\}\\ &\cup \left\{(s,(i,j)) \;\middle|\; (i,j) \in \mathcal{S}\setminus\{s,t\} \;\wedge\; \mathsf{Score}_{\mathcal{E}}(i,j) < 0\right\}\\ &\cup \left\{((i,j),t) \;\middle|\; (i,j) \in \mathcal{S}\setminus\{s,t\} \;\wedge\; \mathsf{Score}_{\mathcal{E}}(i,j) > 0\right\}\end{aligned}$$

où $\mathsf{captures}(i,j)$ désigne l'ensemble des positions qu'un cavalier en (i,j) peut capturer en un coup.

— les arcs reliant s aux cavaliers et les pièces à t ont un poids nul, l'arc reliant un cavalier à une pièce a pour poids le score correspondant :

$$\forall (u,v) \in \mathcal{A}.\ \mathsf{poids}(u,v) = \begin{cases} \mathsf{Score}_{\mathcal{E}}(i,j) + \mathsf{Score}_{\mathcal{E}}(i',j') & \text{si } u = (i,j) \neq s \text{ et } v = (i',j') \neq t \\ 0 & \text{si } u = s \text{ ou } v = t \end{cases}$$

Intuitivement, cet encodage fonctionne de la façon suivante : trouver un flot de valeur f de s à t dans $\mathcal{G}(\mathcal{E})$ donne un ensemble d'arcs, dont f sortent de s, et entrent donc dans f cavaliers. De chacun de ces f cavaliers sort alors un arc vers une pièce, sélectionnant ainsi la pièce qu'il doit capturer. Le poids de f est égal au score obtenu en capturant ces f pièces.
On admettra que le score maximal atteignable sur $\mathcal{E}$ en k' captures, que l'on recherche, est égal au poids maximal d'un flot de valeur k' de s à t dans $\mathcal{G}(\mathcal{E})$.

On utilisera, pour le calcul numérique des réponses aux questions 11 et 12, l'ensemble $\mathsf{F}_0(\mathcal{E}) = \left\{((i,j),(i',j')) \in \mathcal{A} \;\middle|\; \{(i,j),(i',j')\} \cap \{s,t\} = \emptyset \;\wedge\; i' \bmod 2 = 0\right\}$ des arcs ne reliant ni s ni t qui entrent dans un sommet d'abscisse paire. $\mathsf{F}_0(\mathcal{E})$ est un sous-ensemble des arcs de $\mathcal{G}(\mathcal{E})$. Il ne s'agit pas en général d'un flot, mais ce sera sans importance pour les calculs demandés.

L'algorithme de Ford-Fulkerson permet de calculer un flot de poids maximal. Il repose sur la construction d'un graphe résiduel. Étant donné $\mathcal{G} = (\mathcal{S}, \mathcal{A}, \mathsf{poids})$ et un flot F, le graphe résiduel de $\mathcal{G}$ pour F, $\mathcal{R}_{\mathsf{F}}(\mathcal{G}) = (\mathcal{S}, \mathcal{A}_{\mathcal{R}_{\mathsf{F}}(\mathcal{G})}, \mathsf{poids}_{\mathcal{R}_{\mathsf{F}}(\mathcal{G})})$, possède les mêmes sommets que $\mathcal{G}$. Ses arcs sont ceux de $\mathcal{G}$, à ceci près que l'on inverse le sens et le poids de ceux qui sont présents dans F. C'est-à-dire que $(u,v) \in \mathcal{A}_{\mathcal{R}_{\mathsf{F}}(\mathcal{G})}$ si et seulement si

— $(u,v) \in \mathcal{A} \setminus \mathsf{F}$, et alors $\mathsf{poids}_{\mathcal{R}_{\mathsf{F}}(\mathcal{G})}(u,v) = \mathsf{poids}(u,v)$;

— ou bien $(v,u) \in \mathsf{F}$ et alors $\mathsf{poids}_{\mathcal{R}_{\mathsf{F}}(\mathcal{G})}(u,v) = -\mathsf{poids}(v,u)$.

Remarque. Cette construction est bien définie, car on a imposé en 1.2 que $\mathcal{A}$ ne contienne jamais à la fois (u,v) et (v,u).

Question 11 *Écrire un programme qui, étant donné $\mathcal{E}$ et F, calcule le graphe $\mathcal{R}_{\mathsf{F}}(\mathcal{G}(\mathcal{E}))$. Pour les $\mathcal{E}$ suivants, avec $\mathsf{F} = \mathsf{F}_0(\mathcal{E})$, donner la valeur absolue*

$$\left| \sum_{u \in \mathcal{S}_{\mathcal{R}_{\mathsf{F}}(\mathcal{G}(\mathcal{E}))}} \sum_{v \in \mathsf{succ}_{\mathcal{R}_{\mathsf{F}}(\mathcal{G}(\mathcal{E}))}} \mathsf{poids}_{\mathcal{R}_{\mathsf{F}}(\mathcal{G}(\mathcal{E}))}(u, v) \right| \bmod 10\,000.$$

a) $\mathcal{E}_{8,20,20}$ ***b)*** $\mathcal{E}_{60,250,140}$ ***c)*** $\mathcal{E}_{110,850,400}$

En utilisant cette notion de graphe résiduel, nous pouvons maintenant présenter l'algorithme de Ford-Fulkerson pour le calcul du poids maximal d'un flot de valeur f dans $\mathcal{G} = (\mathcal{S}, \mathcal{A}, \mathsf{poids})$.

```
F := []
tant que valeur(F) < f
    C := chemin de s à t de poids maximal dans R_F(G)
    si un tel C existe alors
        mettre à jour F comme suit :
            pour chaque arc (u, v) de C
                si (u, v) ∈ A alors ajouter (u, v) à F
                si (v, u) ∈ A alors retirer (v, u) de F
    sinon
        renvoyer erreur
renvoyer poids(F)
```

Algorithme 3 : Algorithme de Ford-Fulkerson

F est à chaque étape un flot, initialement vide, et que l'on va remplir progressivement jusqu'à atteindre la valeur f cherchée. À chaque étape, on cherchera un chemin qui permettra d'augmenter la valeur de F. Si c'est impossible, alors on admet qu'il n'existe aucun flot de valeur f, et on renverra une erreur.
Le calcul du chemin de poids maximal dans $\mathcal{R}_{\mathsf{F}}(\mathcal{G})$ peut être réalisé avec l'algorithme de Bellman-Ford, vu dans la partie 3.2.

Question 12 *Écrire un programme qui, étant donné $\mathcal{E}$, F, calcule un chemin de poids maximal de s à t dans $\mathcal{R}_{\mathsf{F}}(\mathcal{G}(\mathcal{E}))$ avec l'algorithme de Bellman-Ford. Donner le poids de ce chemin pour les $\mathcal{E}$ suivants, avec $\mathsf{F} = \mathsf{F}_0(\mathcal{E})$.*

a) $\mathcal{E}_{8,20,20}$ ***b)*** $\mathcal{E}_{60,250,140}$ ***c)*** $\mathcal{E}_{110,850,400}$

Indication. On pourra bien sûr réutiliser le programme écrit pour la question 8, en prenant soin de l'adapter au graphe utilisé ici. On admettra que $\mathcal{R}_{\mathsf{F}}(\mathcal{G}(\mathcal{E}))$ ne contient pas de cycle positif – c'est le cas au cours de l'algorithme de Ford-Fulkerson.

Question 13 *Écrire un programme qui, étant donné $\mathcal{E}$ et k', calcule le score maximal atteignable en capturant k' pièces blanches par k' cavaliers. On utilisera l'algorithme de Ford-Fulkerson pour calculer le poids maximal d'un flot de valeur k' sur le graphe $\mathcal{G}(\mathcal{E})$ décrit précédemment. Donner ce score maximal pour les entrées suivantes.*

a) $\mathcal{E}_{8,31,31}, k' = 25$ ***b)*** $\mathcal{E}_{10,47,45}, k' = 38$ ***c)*** $\mathcal{E}_{20,150,150}, k' = 120$

Question à développer pendant l'oral 10 *Évaluer la complexité en temps de votre programme.*

♢
♢ ♢
♢

Fiche réponse type : Charge de cavalerie

$\widetilde{u_0}$: 1530

Question 1

a) 669

b) 392

c) 224

d) 391

Question 2

a) 216

b) 789

c) 201

Question 3

a) 39

b) 503

c) 5056

Question 4

a) 411

b) 5390

c) 1573

Question 5

a) 92

b) 752

c) 716

Question 6

a) 1980

b) 6127

c) 6372

Question 7

a) 2334

b) 7362

c) 9893

Question 8

a) 115

b) 2873

c) 7666

Question 9

a) 474

b) 7317

c) 922

Question 10

a) 234

b) 8575

c) 8539

Question 11

a) 254

b) 4024

c) 5314

Question 12

a) 62

b) 121

c) 11992

Question 13

a) 215

b) -2

c) 6772

Fiche réponse : Charge de cavalerie

Nom, prénom, u_0 : .....

Question 1

a)

b)

c)

d)

Question 2

a)

b)

c)

Question 3

a)

b)

c)

Question 4

a)

b)

c)

Question 5

a)

b)

c)

Question 6

a)

b)

c)

Question 7

a)

b)

c)

Question 8

a)

b)

c)

Question 9

a)

b)

c)

Question 10

a)

b)

c)

Question 11

a)

b)

c)

Question 12

a)

b)

c)

Question 13

a)

b)

c)