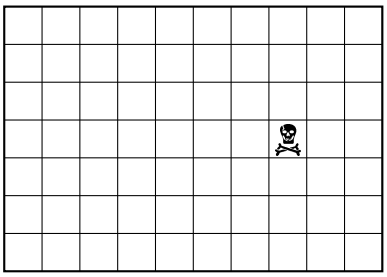
TP Programmation dynamique: le jeu du chocolat

Description du problème

Dans ce TP, on s'intéresse au jeu suivant. On prend une tablette de chocolat, c'est-à-dire une grille de m colonnes et n lignes, dans laquelle un des carrés est marqué d'un signe spécial. C'est le `carré de la mort`, et on note (i, j) ses coordonnées. Pour fixer les idées, le carré en haut à gauche est le carré de coordonnées $(0, 0)$, tandis que le carré en bas à droite est le carré de coordonnées $(m-1, n-1)$.

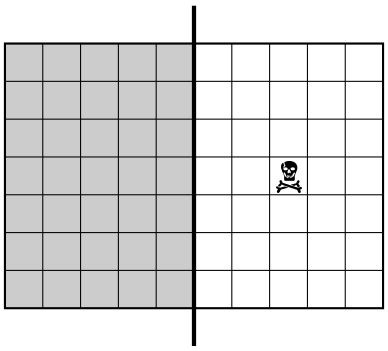


$(m, n) = (10, 7)$ et $(i, j) = (7, 3)$

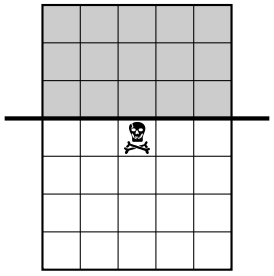
Le jeu se joue de la façon suivante. Le premier joueur prend la tablette, la casse en deux (horizontalement ou verticalement), et donne la partie qui contient le carré mortel à l'autre joueur (la partie restante ne sert plus à rien et peut être consommée). L'autre joueur procède de même. Au fur et à mesure que la partie progresse, la tablette rétrécit, et fatalement au bout d'un moment il ne reste plus que le carré de la mort tout seul. Celui qui reçoit des mains de l'autre le carré de la mort a perdu.

Exemple

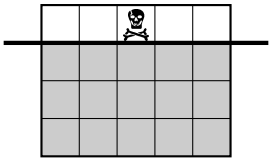
Par exemple, le joueur 1 reçoit la configuration suivante et commence pas casser verticalement la tablette en deux moitiés égales :



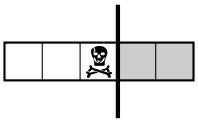
Le joueur 2 reçoit des mains du joueur 1 la moitié de la tablette contenant le carré de la mort. Dans cette nouvelle configuration il décide de casser horizontalement après la 3^{e} ligne :



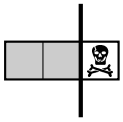
Le joueur 1 décide lui-aussi de la casser horizontalement après la première ligne de la configuration qu'il reçoit :



Le joueur 2 casse verticalement le long de la 3^{e} colonne :



Et pour finir, le joueur 1 casse verticalement le long de la 2^{e} colonne, et gagne car il donne le carré de la mort au joueur 2.



Objectif du TP

L'objectif est d'utiliser la programmation dynamique pour calculer la stratégie optimale de jeu en fonction de la configuration courrante de la tablette, c'est-à-dire déterminer quel coup jouer pour s'assurer de gagner le plus rapidement lorsque c'est possible. Lorsqu'il est impossible de gagner compte tenu de la configuration de la tablette, le coup optimal consiste à retarder sa défaite au maximum.

En effet, dans un jeu comme celui-ci où il n'y a pas de match nul, on est forcément dans l'une de ces deux situations :

1. Soit le joueur qui commence peut, en jouant correctement, gagner quoi que fasse l'autre.
2. Soit le joueur qui commence, quoi qu'il fasse, va perdre si l'autre joue correctement.

A chaque configuration de la tablette il est possible d'associer une *valeur*, qui exprime à quel point la configuration est favorable au joueur dont c'est le tour. La valeur d'une configuration est l'entier $+k$ si le joueur dont c'est le tour peut gagner en maximum k coups, quoi que fasse l'autre. Il peut éventuellement gagner plus vite si l'autre joue mal.

La valeur est l'entier $-k$ si le joueur dont c'est le tour ne peut pas gagner mais qu'il peut survivre au moins k coups face à un adversaire optimal. Il peut toutefois gagner si son adversaire commet des erreurs.

Déterminer la stratégie optimale revient donc simplement à être capable de calculer la valeur d'une configuration pour guider le choix de la coupe à effectuer. Dans un premier temps vous allez travailler sur le calcul de la valeur d'une configuration qui constitue le coeur des algorithmes que vous allez coder.

Pour mesurer l'intérêt d'utiliser la programmation dynamique et les modifications qu'elle implique dans les algorithmes, vous commencerez par coder une version récursive simple, c'est-à-dire *sans* faire appel à ce paradigme algorithmique.

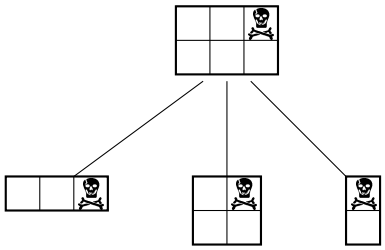
Vous coderez ensuite une première version utilisant la programmation dynamique, avant de réfléchir à des améliorations possibles basées sur les particularités du problème traité.

Q 1. Quels paramètres permettent de définir de manière unique une configuration de la tablette ?

Valeur d'une configuration

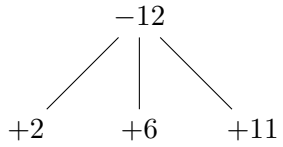
La première étape du TP est de déterminer une formule mathématique récursive exprimant le calcul de la valeur d'une configuration en fonction des valeurs de l'ensemble de ses configurations successeures.

Par exemple, la configuration suivante reçue par l'un des joueurs peut déboucher pour son adversaire sur 3 configurations différentes suivant le choix de découpe qu'il va faire :

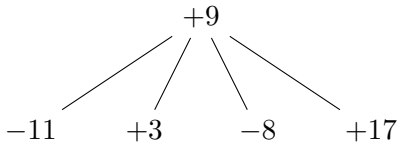


En fonction des valeurs de ces 3 configurations successeurs, le joueur va pouvoir déterminer la valeur de la configuration qu'il a reçue et quelle coupe est optimale dans sa situation.

Pour comprendre ce calcul, considérons une configuration (fictive) qui a 3 successeurs, dont les valeurs sont respectivement $+2, +6$ et $+11$.

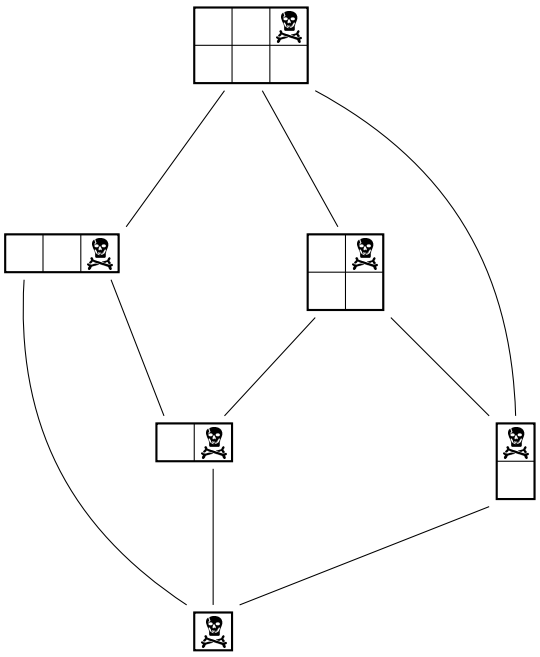


Cela signifie qu'il y a 3 choix de coupe possibles, mais que toutes les coupes offrent une configuration gagnante à l'adversaire. La configuration de départ est donc perdante, et sa valeur est -12. En effet, en choisissant le 3e successeur, l'adversaire ne pourra pas gagner en moins de 11 coups, et donc comme ça on pourra « tenir » 12 coups en tout avant de rendre l'âme (en espérerant une erreur de la part de l'adversaire au cours de ses 12 coups !).



Maintenant considérons une configuration qui a 4 successeurs de valeurs $-11, +3, -8$ et $+17$. Cette configuration est gagnante, parce qu'en choisissant (par exemple) la première coupe, on place l'adversaire dans une situation perdante. Mais pour gagner le plus vite possible, il vaut mieux choisir le 3e successeur, car alors on est sûr de gagner en 9 étapes maximum. La valeur de la configuration de départ est donc $+9$.

Q 2. Déterminez à la main les valeurs des positions suivantes, et faites-les figurer dans votre rapport.



Q 3. Etant donnée une configuration (cf Q1), comment obtenir ses configurations successeures ? *Dans cette question on considère les configurations mais pas encore leur valeur.*

Q 4. Donnez une formule mathématique permettant de calculer la valeur d'une configuration à partir des valeurs de ses successeurs. Il peut être utile de distinguer le cas où un des successeurs au moins à une valeur négative du cas où ils sont tous positifs.

Version récursive « naïve »

Q 5. Coder une fonction récursive calculant la valeur d'une configuration dans le langage de votre choix. Vous pouvez prendre en ligne de commande les arguments représentant la configuration initiale de la tablette dans votre programme principal pour pouvoir faire des tests facilement par la suite.

Q 6. Testez votre fonction sur les configurations de petite taille (3×2) du début du TP. Combien de temps met votre code à résoudre l'instance 10×7 figurant sur la première page du TP, puis sur l'instance dans laquelle le carré de la mort est décalé de 2 colonnes vers la gauche (colonne 5 au lieu de 7).

Q 7. Comment expliquer la différence entre les deux résolutions ? La complexité de cette procédure est-elle polynomiale ou exponentielle (justifiez) ?

Version « dynamique »

Pour passer de la version récursive simple à une version dynamique il est nécessaire de réfléchir au stockage des valeurs de toutes les configurations successeures de la configuration initiale.

Q 8. Proposez une structure de données et le cas échéant une manière de représenter une configuration permettant de stocker les valeurs calculées.

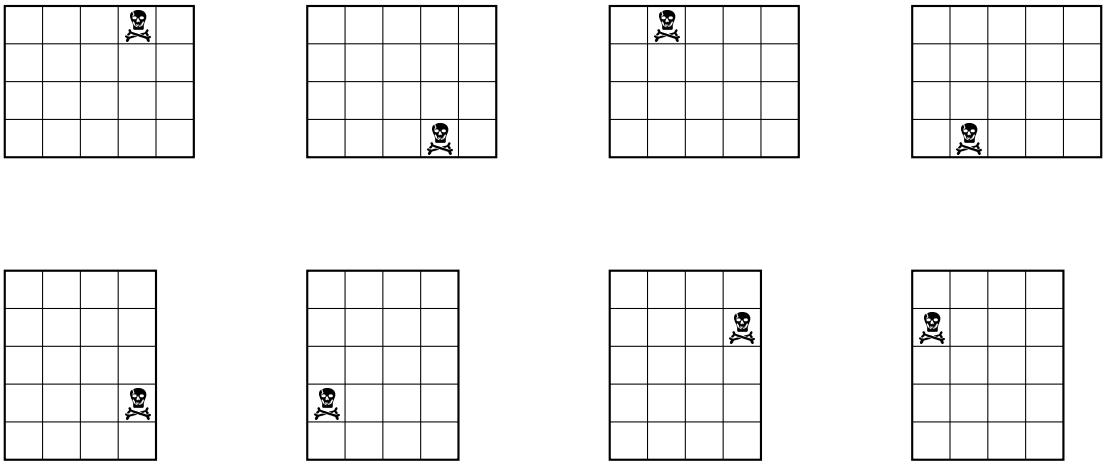
Q 9. Déterminez les valeurs des configurations $(100, 100, 50, 50)$ et $(100, 100, 48, 52)$.

Q 10. Déterminez toutes les paires (i, j) telles que la valeur de la configuration $(127, 127, i, j)$ soit égale à 127.

Q 11. Donnez l'ordre de grandeur de la complexité de votre procédure en fonction de m et n. Justifiez.

Accélération

Il est possible d'accélérer sensiblement la procédure, en remarquant que le problème possède de nombreuses symétries. En effet, les 8 configurations suivantes ont la même valeur.



Q 12. Expliquez pourquoi toutes ces configurations ont la même valeur.

Q 13. Trouvez un moyen d'en tirer parti et incorporez-le à votre code. Est-ce que le temps de résolution diminue ?

Q 14. (question facultative) Utilisez les symétries pour réduire la complexité spatiale de votre programme. Quel est l'ordre de grandeur du gain ?

Programmation du moteur de jeu

Q 15. Écrivez un programme qui prend sur la ligne de commande les 4 arguments décrivant la configuration initiale, et... qui joue (on peut admettre qu'il commence). Quand c'est son tour il doit afficher son coup (dans un format intelligible par un être humain), mais aussi la configuration résultante de son coup, ainsi que son évaluation de cette configuration.

Quand c'est au tour de l'adversaire, il doit demander à l'adversaire d'entrer son coup, calculer la configuration résultante et afficher son évaluation.

Il doit évidemment détecter la fin de la partie.

Q 16. (question facultative) Un joueur attentif a remarqué la ressemblance avec les jeux de Nim. Qu'en pensez-vous ?