INF1008 – Analyse d'algorithmes

Session HIV-23

TRAVAIL PRATIQUE #2 – ÉCHEC ET MATH

Indications principales	
Date de présentation de l'énoncé	31 mars 2023
Date de remise du travail	28 avril 2023, 23h55
Politique sur remises tardives	-25% par jour de retard
Éléments et format de remise	Rapport et fichiers complet de la solution ou du projet remis sous forme électronique sur le portail de cours dans un seul fichier compressé (.zip)
Pondération	15%
Nb d'étudiants par équipe	4 (conseil) à 5

INTRODUCTION

Lors du chapitre sur les graphes, nous avons vu que nous pouvons utilisé une technique dite de « retour arrière » comme stratégie pour explorer un espace problème à la recherche d'une solution. L'avantage de la dite approche, en outre, est qu'on peut évaluer une solution partielle et s'assurer qu'avant d'aller de l'avant vers une solution globale, cette solution partielle soit prometteuse. Si on peut évaluer que la solution partielle ne conviendra pas, on peut alors reculer d'un peu et tenter d'autres combinaisons de donner qui elles convergent de plus en plus près d'une solution. On évite ainsi une grande partie de la combinatoire des jeux de données à tester.

L'exemple que nous avons exploré ensemble concernait un jeu d'échec 8 cases par 8 cases pour lequel on désirait placer 8 reines sur le plateau sans qu'aucune reine ne « menace » l'autre, c'est-à-dire de telle sorte qu'il n'y ait qu'une seule reine par ligne, colonne et diagonale. Nous avions démontré comment on pouvait passer d'une solution de type force brute à plusieurs milliards de possibilités à tester jusqu'à une solution de retour arrière. Vous avez aussi dans les notes de cours un algorithme et un exemple graphique avec un plateau 4 cases par 4 cases et 4 reines.

Dans le cadre de ce travail, vous êtes appelés à implémenter dans un langage .NET de votre choix (C++, C#, VB.NET, F#) ou Java cet algorithme de retour arrière selon les modalités qui seront décrites dans les sections à suivre.

SPÉCIFICATIONS DU TRAVAIL DEMANDÉ

Caractéristiques/fonctionnalités

Votre application doit permettre, grâce à un algorithme de *backtracking*, de pouvoir résoudre le problème du positionnement de ***n*** reines sur un plateau de jeu ***n*** x ***n*** et, au choix de l'utilisateur, afficher la première solution, ou l'ensemble des solutions de placement possible.

Vous devrez compter le nombre de tuples k-prometteurs produits avant d'obtenir chacune des solutions possibles, ainsi que la quantité de solution pour chacun des n. Les solutions devront être affichées/affichables.

Application (12%)

Votre application console ou avec UI doit implémenter la solution à la suite décrite précédemment, soit :

- Une classe *JeuEchec* (ou l'équivalent) adéquate qui doit **(12%)**:

  - Pouvoir lancer une **méthode de résolution** par retour arrière pour une taille de plateau ***n*** donnée en paramètre ***(1.5%)***;

  - Appeler à partir de cette méthode de résolution une **méthode récursive** qui fouillera l'arborescence de recherche ***(5%)***;

    - Cette méthode récursive doit contenir les structures de données nécessaires pour conserver en mémoire le jeu d'essai k-prometteur, les colonnes déjà couvertes par d'autres reines et les diagonales positives et négatives également couvertes par d'autres reines *(2%)*;

    - Elle doit évidemment implémenter l'équivalent sémantico-logique de l'algorithme spécifié dans les notes de cours *(3%)*.

  - Faire en sorte de rendre les méthodes précédentes assez malléables (ou en créer des similaires) afin qu'elles puissent retourner la première solution trouvée ou toutes les solutions ***(2.5%);***

  - Contenir des instructions nécessaires pour interpréter le ou les jeux d'essais n-prometteurs retournés comme solution(s) à des fins d'affichage, notamment. ***(1.5%)***.

  - Intégrer des instructions qui comptent le nombre de tuples k-prometteurs produits avant d'obtenir chacune des solutions possibles et total, ainsi que le nombre de solutions trouvées au total si applicable ***(1.5%)***.

- Une classe *Main* ou un formulaire qui doit **(3%)** :

    - Permettre à un utilisateur (par un menu qui boucle si vous faites cela en application console) de : ***(1%)***

        - Demander une solution possible pour une taille de plateau donnée; *(0.5%)*

        - Demander toutes les solutions possibles pour une taille de plateau donnée. *(0.5%)*

    - Afficher à la console ou dans des contrôles adéquats : ***(2%)***

        - La ou les solutions identifiées; *(1%)*

        - Le nombre de tuples k-prometteurs produits avant de trouver chacune des solutions possibles; *(0.5%)*

        - Le nombre de tuples k-prometteurs total pour explorer toute l'espace de recherche. *(0.5%)*

CONSEILS :
Ne négligez pas d'ajouter tout commentaire pertinent à votre code. Ils ne seront pas évalués comme tel, mais ils pourront tout de même aider lors de la correction à comprendre ce que vous vouliez faire en cas d'erreur.

Rapport (3%)

Vous devez également remettre un **rapport Word ou PDF** qui doit contenir les éléments suivant :

- Page de présentation ***(0.5% les 3 premières exigences)***;
- Problèmes et difficultés rencontrés (s'il y a lieu);
- Instructions spéciales d'exécution du programme (s'il y a lieu);
- Une présentation des résultats obtenus en lançant l'application pour un ***n*** allant de 1 jusqu'à 10 ***(2.5%)***.

  Vous pouvez présenter les résultats dans un tableau. Vous êtes tenus de donner pour chaque valeur de *n* toutes les solutions si elles existent **(les 4 premières seulement s'il y en a plus que 4)**, après combien de tuples k-prometteurs elles sont trouvées, combien au total il en existe, et combien de tuples doit-on parcourir pour explorer tout l'espace de recherche. Vous pouvez utiliser des imprimes-écran de votre application, ou reproduire les plateaux dans des tableaux (comme les notes de cours).

EXIGENCES ET INSTRUCTIONS SUPPLÉMENTAIRES

Langage de programmation

Au niveau du langage de programmation, vous devez utiliser soit l'une ou l'autre des options suivantes, **sans aucune exception non approuvée** :

- C#/VB/C++/F# avec Microsoft Visual Studio .NET 2017+
- Java avec NetBeans ou Eclipse

Éléments à remettre

Vous devrez remettre l'ensemble des extrants exigés qui seront énumérés ci-dessous dans un seul fichier compressé (en format .zip) dans la section de dépôt des travaux sur le portail du cours **avant le début du cours de la remise.**

Dans le cas où vous remettiez le travail en retard, vous ne pourrez alors le déposer sur le portail et vous devrez me le retourner via courriel. Une pénalité de 25% par jour de retard, à compter de l'heure de remise, serait alors appliquée.

Les éléments à remettre sont les suivants :

- Tous les fichiers relatifs au projet de l'application :
    - Avec Visual Studio .NET, vous devez remettre le répertoire contenant la solution, le ou les projets associés, les fichiers contenant le code source, les exécutables, etc.
    - Une vidéo de courte durée montrant l'exécution de votre code

    Dans tous les cas, assurez-vous que le projet soit prêt à être exécuté directement à l'intérieur de la plateforme de développement, c'est-à-dire sans erreur de compilation et sans configuration spéciale non explicitement donnée. Aucun débogage ne sera fait lors de la correction. C'est TRÈS important. Si je dois reconstruire un nouveau projet et importer les classes, les fichiers de configuration et configurer des paramètres pour vous, il y aura d'importantes pénalités appliquées.
- Le rapport PDF.