TP Algo2 – session 11 – Projet-tournoi

15 mai 2025

Objectif d'apprentissage : écrire un programme qui joue à un jeu de poursuite/évasion. Il s'agit de la seconde séance préparatoire sur les trois.

1 Algorithmes

Le fichier `algo.h` ne changera plus et il reste donc deux évaluations blanches sur le fichier `algo.c`. Profitez-en !

Cette séance apporte deux précisions sur cette partie :

— la fonction `board_read_from` doit être complétée pour éviter tous les avertissement et pour retourner `false` lorsque le format d'entrée n'est pas respecté ;

— l'algorithme de Floyd-Warshall doit prendre en compte la connexité du graphe : si deux sommets ne sont pas connectés, `board_dist` et `board_next` doivent retourner `INT_MAX`. Le pseudo-code du TP précédent à été modifié dans ce sens (au niveau de l'initialisation).

2 Format d'entrée des plateaux de jeu

Le format des plateaux de jeu est le suivant :

```
Cops: NOMBRE_DE_GENDARMES
Robbers: NOMBRE_DE_VOLEURS
Max turn: NOMBRE_DE_TOURS_MAXIMUM
Vertices: NOMBRE_DE_SOMMETS
COORDONNEES_DES_SOMMETS
Edges: NOMBRE_D_ARETES
ARETES
```

Chaque partie en capitale doit être remplacée par des valeurs numériques. Il s'agit d'une valeur entière strictement positive dont le sens est direct lorsque cela commence par `NOMBRE_DE_` (à l'exception de `NOMBRE_D_ARETES` qui peut être nul).

Pour `COORDONNEES_DES_SOMMETS`, il s'agit d'une liste de couples de valeurs flottantes qui décrivent les abscisses et ordonnées de chaque sommet. Ces valeurs (deux par ligne) doivent être comprises entre -1 et 1. Ces coordonnées sont utilisées par l'affichage graphique de `server.py` pour positionner les sommets. Elles doivent donc être ignorées par `board_read_from` : il faut qu'il y ait le bon nombre de lignes, mais on ne vérifie pas leur validité.

Pour `ARETES`, il s'agit des arêtes qui connectent les sommets. Chaque ligne représente une arête avec les indices des deux sommets connectés.

Il est possible de proposer des plateaux personnalisés qui pourront être inclus dans le tournoi et donner lieu à un bonus.

3 Programme de jeu

Le fichier `game.c` que vous réaliserez doit fournir un programme qui respecte l'interface définie à la séance précédente. En plus, il ne doit jamais prendre plus d'une seconde pour calculer le prochain déplacement : `server.py` utilise un *timeout* d'une seconde pour chaque lecture sur l'entrée d'un des

programmes (réinitialisé à chaque tour, le temps de calcul cumulé pour une seule partie peut donc dépasser cette limite).

Le programme `game.c` s'appuie sur deux structures : `vector` et `game`. La première permet de représenter les positions soit des gendarmes, soit des voleurs. Il s'agit d'un tableau de pointeurs vers les sommets occupés. La seconde structure regroupe un plateau, les positions des gendarmes et des voleurs, le nombre de tours restants et le rôle que joue le programme (gendarme ou voleur).

Parmi les fonctions fournies, celle qui calcule les positions est `game_next_position`. C'est celle-ci qu'il faudra ajuster pour y insérer vos algorithmes :

— placement initial des gendarmes ;
— placement initial des voleurs, sachant les positions des gendarmes ;
— prochaines positions des gendarmes ;
— prochaines positions des voleurs.

Toutes les fonctions nécessaires devront être ajoutées dans `game.c` et il sera possible de s'appuyer sur les fonctions définies dans `algo.h` et implémentées dans `algo.c`.

Plutôt que de repartir du fichier `game.c` fourni, il est aussi possible de repartir de zéro, mais en garantissant le respect de l'interface : arguments lors de l'appel du programme et entrées/sorties pour chaque tour de jeu.

4 Seconde séance

À l'issue de cette seconde séance, l'incrément suivant est attendu :

— découvrir le sujet et l'archive fournie ;
— réviser les algorithmes élémentaires associés à `algo.h` ;
— réviser `game.c` pour implémenter une stratégie du même niveau que `mid` : les gendarmes se déplacent vers les voleurs et les voleurs s'éloignent des gendarmes.

Une évaluation facultative blanche vérifiera ces premiers éléments. Elle est vivement recommandée quelque soit l'état d'avancement. Il faudra soumettre trois fichiers : `algo.c` contenant les implémentations, `algo_tests.c` contenant les tests unitaires (à compléter pour chaque méthode) et `game.c`. Si une fonction auxiliaire paraît utile pour les tests, c'est dans le fichier de tests qu'il faudra la rajouter.

Les trois fichiers doivent être soumis **tels quels** pour permettre le traitement et l'évaluation automatique (non compressés, non archivés, non renommés).