Concours de méca-fourmis - 2023

Date limite de rendu pour le tournoi : dimanche à 12h (dernier push)

Compétiteurs, compétitrices, welcome to the annual méca-fourmis tournament !

Vous y voilà, le plus grand tournoi de méca-fourmis de la capitale du monde : Bruz. Pendant les prochaines **36 heures**, votre tâche sera de concocter le programme le plus performant pour le tournoi final.

Les méca-fourmis peuvent faire tourner un language très simpliste, le Fourmissembleur. Mais travailler avec risque d'être insuffisant si vous souhaitez vous confronter aux meilleurs. La solution est donc toute trouvée : **désignez un language plus haut niveau dans lequel implémenter votre startégie, et implémentez un compilateur de ce langage vers le Fourmissembleur.**

Un exemple de language ainsi qu'un compilateur correspondant vous sont fournis comme point de départ. Vous n'êtes absolument pas limités à améliorer ce compilateur. Libre à vous d'utiliser un autre language, un autre parser, ChatGPT ou que sais-je encore. Seul votre language et le programme compilé rendus nous intéressent !

Ce document vous servira de référence. Gardez-le précieusement et lisez-le attentivement pour ne rien louper. Si le document manque de clarté, n'hésitez pas à solliciter les organisateurs (promis, on ne le prendra pas mal).

1 Règles du jeu

Un match confronte deux équipes de méca-fourmis sur une carte au hasard. Le but est simple : **ramener à votre base plus de nourriture que l'équipe adverse.**

Tous les coups sont permis pour y arriver (excepté incapaciter les autres participants et/ou soudoyer le jury) !

1.1 Méca-fourmis

Le seul contrôle que vous avez sur vos unitées se fait via **un unique programme fourmissembleur partagé par toutes vos méca-fourmis**.

Chaque unitée est définie par :

- **Son équipe**, qui définit le programme qu'elle suivra.

- **Un identifiant** attribué au hasard. Les unitées bougent une-à-une durant un tour de jeu dans l'ordre croissant de leur identifiant.
- **Sa position** dans le programme de son équipe, pour savoir quelle instruction elle doit exécuter ensuite.
- **Son statut actuel**. Dans une optique de green IT, les méca-fourmis fonctionnent à l'énergie solaire et doivent se recharger plus ou moins longtemps avant d'effectuer leur prochaine action (le nombre de tours de repos par instruction est indiqué en annexe). De plus, elles peuvent être désactivées temporairement par les autres unitées, les laissant vulnérables aux attaques (plus d'informations sur le combat plus bas).
- **Son orientation courante**, parmi est, ouest, nord-est, nord-ouest, sud-est et sud-ouest. Toutes les unitées commencent le jeu orientées vers l'est.
- Si oui ou non elle **transporte actuellement de la nourriture**.

1.1.1 Phéromones

Les méca-fourmis peuvent marquer la carte par des phéromones, pour un total de 8 bits d'information ! De quoi rendre jaloux n'importe quel super-calculateur !

Libre à vous de les utiliser efficacement pour votre stratégie.

Les phéromones de chaque équipe sont complètement séparées. Il est possible pour une unitée de détecter si une case a été marquée par l'équipe ennemie, mais aucune autre interaction n'est possible.

Ces phéromones n'ont pas de durée de vie, elles restent jusqu'à ce qu'une méca-fourmi les modifie.

Au début du jeu, aucune case n'est marquée.

1.1.2 Art martial des méca-fourmis

Quand il s'agit de récupérer de la nourriture, les méca-fourmis sont tout sauf pacifistes. Durant l'année passée, elles ont parfait leur art martial et sont désormais plus meurtrières que jamais… pour peu qu'elles puissent se coordonner !

Il est possible d'**attraper** l'unitée en face de soi pour la désactiver pendant **40 tours**. Ce n'est pas sans risque pour l'aggresseuse, qui elle-même se retrouve désactivée durant **20 tours** avant une pause de **30 tours** (c'est fatiguant d'être méchante).

Toute méca-fourmi peut **attaquer** l'unitée en face d'elle, mais cette attaque ne réussit que si l'unitée en face est déjà désactivée. Si l'attaque réussit, feu la méca-fourmi est tuée, et laisse derrière elle 3 unitées de nourriture.

Attention, même si les méca-fourmis sont très entraînées, elles ne peuvent toujours pas faire la différence entre alliées et ennemies quand il s'agît d'attraper ou de porter le coup fatal ! En gros, **attention au friendly-fire** !

1.2 Cartes

Les cartes sont des pavages hexagonals encodés dans des fichiers ASCII. Quelques fichiers de base vous seront fournis afin d'entraîner vos méca-fourmis dans les meilleurs conditions.

Une carte est définie par un header de la forme `largeur hauteur profondeur` sur la première ligne, puis des cellules elles-mêmes.

Il existe 5 types de cellules :
- Les **rochers** (`#`), infranchissables et indestructibles.
- Les **cellules vides** (`.` ou $x \in 0, \ldots, 9$ si elles contiennent de la nourriture).
- Les **fourmillières** ($c \in a, \ldots, h$ indiquant l'index de l'équipe), situées uniquement dans l'étage le plus profond.

- Les **cellules creusables** (`x`) disposées uniquement en sous-sol.
- Les **tunnels** de la surface vers le sous-sol (`o`), permettant aux unitées de passer d'un étage à l'autre.

Il ne peut se trouver qu'une méca-fourmi par cellule. Initialement, **une unitée est créée sur chaque cellule "fourmillière"**.

Il n'existe sur une carte qu'une fourmillière unie par équipe. Aucune garantie n'est donnée sur la taille de cette fourmillière aka le nombre de méca-fourmis, si ce n'est qu'il est le même pour chaque équipe.

1.2.1 Sous-sols

Les méca-fourmis ont encore une fois évoluées, et se sont rendu compte que placer leur base en surface n'était pas une très bonne stratégie contre les prédateurs. Elles sont devenues des expertes du minage.

Les cartes sont désormais composées d'**au moins deux étages**. En surface, les unitées se déplacent librement. En sous-sol cependant, **il va falloir creuser**.

Il est possible pour une méca-fourmi de creuser les cellules creusables, mais également de remplir les cellules vides du sous-sol si aucune unitée ne s'y trouve. Dans ce cas, les phéromones et la nourriture présentes sur la cellules sont enfouies, et il faudra creuser pour les retrouver. Il reste tout de même possible de les sentir.

Pour passer d'un étage à l'autre, les méca-fourmis peuvent utiliser des trous déjà creusés, ou bien les creuser elles-mêmes. Il leur est aussi possible de boucher des trous pour ne laisser aucune trace de leur passage ou détruire les entrées de l'équipe adverse.

Notez que creuser un trou depuis la surface libère automatiquement la cellule de située en dessous. De la même manière, reboucher une cellule accessible par un trou le rebouche en surface.

2 Simulateur

Un simulateur, **MechAnts** (c'est rigolo ça fait "méchants"), vous a été fourni pour tester vos programmes au fur et à mesure de la compétition.

Il vous est possible de récupérer l'exécutable ou le code source directement ici : https://gitlab.com/aloisrtr/mechants.

Le simulateur peut-être invoqué via la ligne de commande et propose plusieurs sous-commandes.

Le fonctionnement général est décrit ci-dessous, mais l'intégralité des commandes/paramètres peut être récupéré en utilisant `mechants --help`.

2.1 Paramètres généraux

Les paramètres suivants doivent être passés avant la sous-commande et affectent le simulateur de manière générale.

- `--turns <tours>`, `-t <tours>` : permet de spécifier le nombre de tours à jouer avant la fin de la partie. Par défaut, une partie dure 100 000 tours.
- `--seed <seed>`, `-s <seed>` : fixe la seed du générateur aléatoire. Deux parties jouées avec la même seed donneront forcément le même résultat. La seed utilisée dans une partie est indiquée dans l'output avec l'option `--verbose`.
- `--verbose` : montre plus d'informations dans le terminal.

2.2 Match

```
mechants match <carte> <programmes> [--display | -d]
```

Cette commande vous permet de lancer un match entre deux équipes sur une carte et des programmes passés en paramètres.

Si `--verbose` est passé en paramètre, la seed utilisée sera retournée.

Le flag `--display` ou `-d` permet de visualiser le match en direct.

Au moment où le sujet est imprimé, cette visualisation se fait en imprimant la carte en UTF-8 dans le terminal à chaque tour.

2.3 Round-Robin

```
mechants round-robin <carte> <programmes | dossiers>
```

Organise un tournoi round-robin entre plusieurs programmes donnés avec une carte fixée. Passer un ou plusieurs dossiers en paramètre incluera tout fichier `.brain` ou `.bbrain` présent dans ceux-ci.

Chaque programme jouera contre tous les autres, puis un classement final sera généré. Si `--verbose` est passé en paramètre, les résultats précis de chaque match ainsi que la seed utilisée seront donnés en plus des résultats finaux.

2.4 Tournament

```
mechants tournament <carte> <programmes | dossiers> [seeding]
```

Organise un tournoi à double élimination entre plusieurs programmes donnés sur une carte fixée. L'ouverture des programmes fonctionnne de la même manière que le round-robin.

Les programmes seront matchés deux-à-deux, suivant un seeding ou au hasard si aucun n'est passé en paramètre. Une équipe perdant un première fois passera en looser bracket.

Si `--verbose` est passé en paramètre, les résultats de chaque match, l'arbre de tournoi final et la seed utilisée seront donnés en plus des résultats finaux.

2.5 ToBin et Check

```
mechants to-bin <programme> [output]
mechants check <programme>
```

Check permet (comme le nom l'indique) de simplement vérifier si un fichier `.brain` contient des erreurs.

Les fichiers `.brain` peuvent devenir… plutôt lourds (de l'ordre d'une dizaine de gigaoctets si des petits malins se mettent à implémenter des variables).

La théorie, c'est beau, mais nos pauvres ordinateurs ont une quantité de mémoire un peu limitée, et il faut pouvoir charger deux programmes en même temps !

Compiler vos programmes `.brain` vers des `.bbrain` à l'aide de cette commande peut réduire le poids du programme d'un facteur 2 environ. Nous vous demanderons de bien vouloir n'envoyer que des `.bbrain` pour réduire le temps de téléchargement.

Si votre programme est trop lourd pour être chargé lors du tournoi, il sera compté comme incorrect, faites bien attention !

N'oubliez pas de vous amuser, bonne chance et à vos claviers !