Ce challenge est basé sur un système de ligues.

Pour ce défi, plusieurs ligues pour le même jeu sont disponibles. Une fois vos compétences prouvées contre le premier Boss, vous accéderez à une ligue supérieure et des règles supplémentaires seront disponibles.

Dans les premières ligues, votre soumission affrontera uniquement le boss dans l'arène. Gagnez au moins 3 fois sur 5 pour avancer.

Objectif

Marquez plus de points que votre adversaire en traçant des lignes de chemin de fer entre les villes tout en sabotant les efforts de vos adversaires.

Quête annexe

Un classement secondaire **alternatif** est calculé pour ceux qui choisissent de jouer un peu différemment : pouvez-vous construire des voies pour créer des chemins vers des lieux intéressants au lieu de vous battre pour la première place ?

Pour chaque partie dans les soumissions finales du concours, votre bot peut gagner `1` **point de quête annexe** par partie s'il réalise au moins une fois ce qui suit :

- Créer un chemin d'une ville vers un **lieu d'intérêt** sur la carte.
- Le chemin doit être constitué de vos rails et/ou de rails neutres.
- Le chemin doit contenir un rail qui se trouve **sur** la tuile avec le **lieu d'intérêt**.

À la fin du concours, chaque **point de quête annexe** gagné par votre soumission finale comptera dans un total, l'équipe avec la meilleure moyenne de points par partie remporte la side quest.

Règles

Dans ce jeu, les deux joueurs utilisent de la **peinture** pour tracer des voies ferrées sur une carte **magique**. Relier les villes de la carte apportera la prospérité à votre propre monde.

La carte est représentée dans le jeu par une **grille**.

Carte

La grille est composée de cases qui peuvent être de quatre types :

- Type `0` pour les **plaines**.
- Type `1` pour les **rivières**.
- Type `2` pour les **montagnes**.
- Type `3` pour un **lieu d'intérêt**.

La grille est divisée en **régions**. Chaque région est composée de plusieurs cases voisines. Chaque région possède un `regionId` (identifiant de région) unique. Les régions sont sensibles à la **perturbation** par les joueurs.

Plus d'informations sur la perturbation plus bas.

Certaines régions contiendront une **ville**. Les villes ne peuvent se trouver que sur des cases de **plaine**.

Villes

Chaque partie commence avec plusieurs **villes** placées aléatoirement sur la carte. Il y en aura au maximum une par **région** et deux régions partageant une frontière ne contiendront jamais toutes deux une ville.

Chaque ville possède un `townId` (identifiant de ville) unique.

Chaque ville aura une liste de `desiredConnections` (connexions souhaitées) : une liste d'identifiants de villes représentant toutes les **autres villes** auxquelles cette ville souhaite être **connectée** via des **rail** placées par les joueurs.

Fournir à une ville des **rails** la reliant à une ville souhaitée est la façon dont les joueurs marquent des **points**.

Les connexions souhaitées sont **unilatérales**. Cela signifie que si la ville `0` apparaît avec une connexion souhaitée vers la ville `1` , la ville `1` **ne voudra pas** se connecter à la ville `0` .

Une ville peut avoir `zero` `desiredConnections` (connexions souhaitées), mais sera toujours l'objet d'au moins une autre ville ayant des `desiredConnections`.

Placement des rails

À chaque tour, les joueurs reçoivent `3` **points de peinture** qu'ils peuvent utiliser pour placer des **rails** sur la carte. ⚠️ Ces points ne se reportent pas au tour suivant et seront perdus s'ils ne sont pas utilisés.

Coût :

- `1` point de peinture pour placer un rail sur les **plaines**.
- `2` points de peinture pour placer un rail sur une **rivière**.
- `3` points de peinture pour placer un rail sur les **montagnes**.
- `3` points de peinture pour placer un rail sur un **lieu d'intérêt**.

Le `owner` (propriétaire) d'un rail est le `playerId` (`0` - `1`) du joueur qui l'a placée.

Si les deux joueurs placent un rail au **même tour** et au **même endroit**, le `owner` du rail sera `2` , indiquant un rail neutre.

un rail ferrée **ne peut pas** être placée sur une **ville** ni sur un rail existante.

Une fois placée, un rail ferrée se **connectera automatiquement** aux autres **rails** et **villes** adjacentes orthogonalement.

Connexions

Pour chaque paire de villes dont l'une possède l'autre dans ces `desiredConnections` (connexions souhaitée), s'il existe au moins un **chemin** entre les deux, le plus court de ces **chemins** devient la **connexion active** entre ces villes.

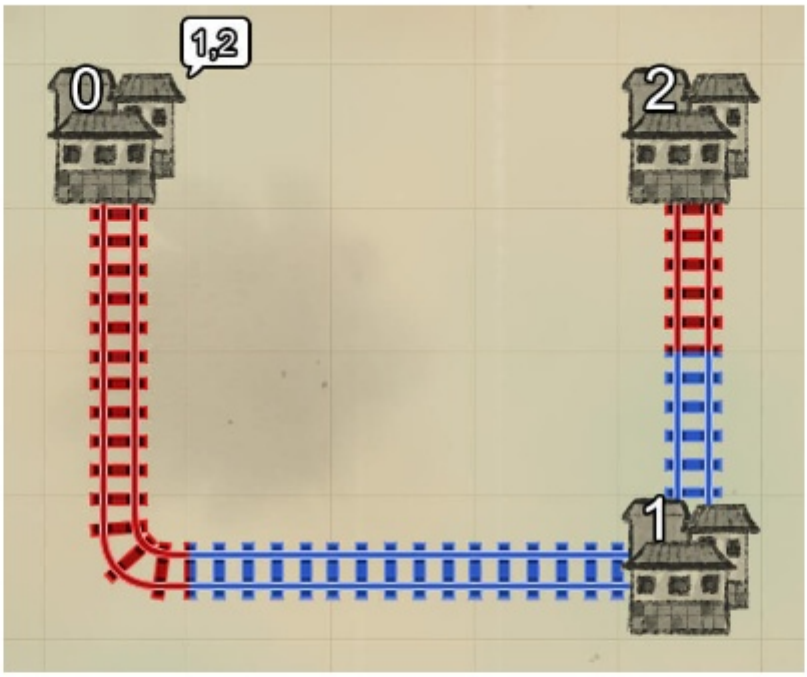
Un **chemin** est une séquence ininterrompue de cases adjacentes orthogonalement contenant un **rail** ou une **ville**.

S'il y a **plusieurs** chemins le plus court possible, le chemin choisi **privilégiera** toujours la direction dans cet ordre lorsqu'on part de la ville demandeuse vers la ville souhaitée :

1. `NORTH`
2. `EAST`
3. `SOUTH`
4. `WEST`

À la fin de chaque tour, chaque **connexion active** rapportera `1` point à chaque joueur pour chaque rail qu'il **possède** dans le **chemin**.

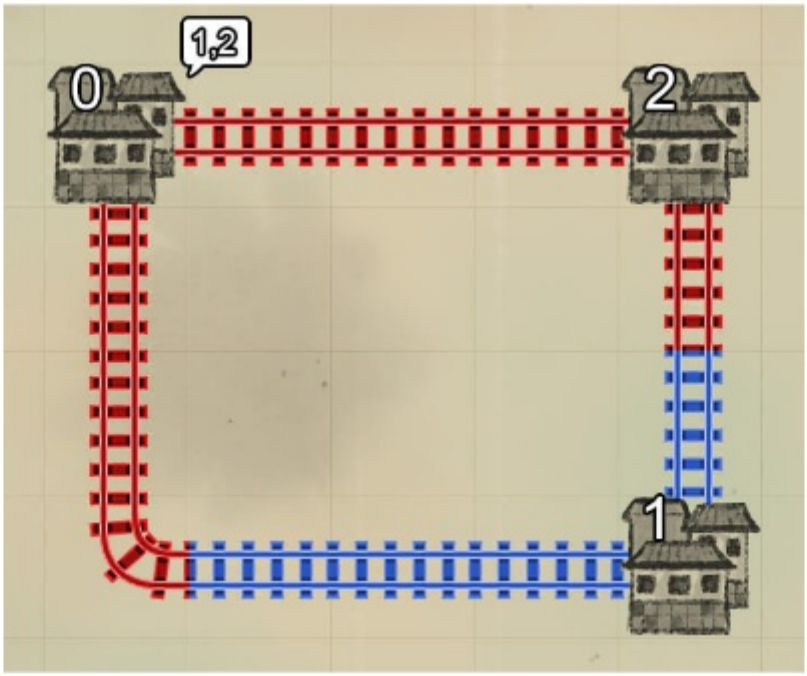
Exemple 1 :



Ici, il y a une **connexion active** de la ville **0** à la ville **1** et une autre de la ville **0** à la ville **2**.

Les joueurs rouge et bleu gagneront tous deux `3` points pour la connexion **0-1** et `4` points pour la connexion **0-2** à la fin du tour.

Exemple 2 :



Ici, seul le plus court chemin de **0** à **2** est utilisé pour la connexion **0-2**. Cela signifie que le joueur rouge gagnera `3` points pour cette connexion, et le joueur bleu n'en gagnera aucun.

Il existe deux chemins de même longueur entre **0** et **1**, mais comme `EAST` a une priorité plus élevée que `SOUTH`, le chemin choisi sera celui qui passe par la ville **2**. Le joueur rouge gagnera `4` points pour cette connexion, et le joueur bleu n'en gagnera que `1`.

💥 Perturbation

De façon similaire aux **points de peinture**, les joueurs peuvent aussi utiliser `1` **point de perturbation** par tour. Il peut être utilisé pour altérer la carte, vous donnant un avantage sur votre adversaire. *Ces points ne sont pas conservés entre les tours.*

Les joueurs peuvent dépenser leur **point de perturbation** à chaque tour pour augmenter l'`instability` (instabilité) de n'importe quelle **région** de `1` .

Une fois que l'`instability` (instabilité) d'une région atteint `3` , cette région est **effacée à l'encre**, supprimant tout **rails** placées et rendant tout placement futur dessus **impossible**. Toute connexion active passant par cette région sera rompue.

Il n'est pas possible de perturber une région qui est déjà **effacée à l'encre**.

Actions

À chaque tour, les joueurs doivent fournir au moins une action sur la sortie standard.

Les actions doivent être séparées par un point-virgule `;` et doivent être l'une des suivantes :

- `PLACE_TRACKS x y` : placer un rail sur une case libre.
- `AUTOPLACE fromX fromY toX toY` : génère automatiquement une liste d'actions pour le chemin **le moins coûteux** des coordonnées `from` au coordonnées `to` en termes de points de peinture. Cela ne fera rien si un chemin existe déjà.
  Les actions générées remplacent cette commande.
- `DISRUPT regionId` : augmente l'instabilité d'une région. *Remarque :* `DISRUPT x y` fonctionne aussi, pour cibler la région dont (x,y) fait partie.
- `WAIT` : ne rien faire.

Conditions de victoire

- Avoir le plus de points après `100` tours.
- Être en tête si toutes les connexions souhaitées deviennent impossibles à réaliser.

Conditions de défaite

Votre programme ne fournit pas de commande dans le temps imparti ou l'une des commandes est invalide.

Détails techniques

- Toutes les actions `PLACE_TRACKS` , y compris celles issues de `AUTOPLACE` , sont effectuées **avant** les actions `DISRUPT` . Les points sont comptés à la toute fin du tour, **après** l'encrage des régions instables.
- Les commandes correspondant à des actions impossibles sont ignorées. Si une action impossible fait partie d'un `AUTOPLACE` , le reste des actions générées est également ignoré, même si elles sont possibles.

Conseils de débogage

- Survolez la grille pour voir des informations supplémentaires sur la case sous votre souris.
- Appuyez sur l'icône d'engrenage du visualiseur pour accéder à des options d'affichage supplémentaires.
- Utilisez le clavier pour contrôler l'action : barre d'espace pour lire/mettre en pause, flèches pour avancer d'une image à la fois.

Cliquez pour développer

Protocole du jeu

Entrée d'initialisation

`myId` : votre identifiant de joueur. `0` ou `1` .

`width` : nombre de cases dans une ligne de la carte.
`height` : nombre de cases dans une colonne de la carte.
Prochaines `height` * `width` **lignes :** deux entiers pour décrire chaque case de la carte, de gauche à droite, de haut en bas :

- `regionId` : l'identifiant de la région dont fait partie cette case.
- `type` : le type de terrain de cette case (`0-3`)

`townCount` : nombre de villes sur la carte.
Prochaines `townCount` **lignes :**

- `townId` : identifiant unique de cette ville.
- `townX` : position X de cette ville (`0` est la plus à gauche).
- `townY` : position Y de cette ville (`0` est la plus en haut).
- `desiredConnections` :
  - Une chaîne de `townIds` séparés par des virgules. ex. " `1,2,4` "
  - " `x` " si cette ville n'a pas de connexions souhaitées.

Entrée pour un tour de jeu

`myScore` : vos points.
`foeScore` : points de votre adversaire.
Prochaines `height` * `width` **lignes :** état de chaque case, dans le même ordre que précédemment :

- `trackOwner` :
  - `-1` si cette case n'a pas de rail.
  - `0` si le joueur `0` possède un rail sur cette case.
  - `1` si le joueur `1` possède un rail sur cette case.
  - `2` si un rail neutre est sur cette case.
- `instability` : instabilité de la région dont fait partie cette case.
- `inked` : `1` si la région a été effacée à l'encre (instabilité ≥ `3`), `0` sinon.
- `partOfActiveConnections` :
  - Une chaîne de paires `townId` séparées par des virgules indiquant que cette case fait partie d'une connexion active entre ces deux villes.
    ex. " `1-2,1-3,4-7` ": *la case fait partie du chemin le plus court entre les villes 1 & 2, villes 1 & 3, et villes 4 & 7.*
  - " `x` " si cette case ne fait partie d'aucune connexion active.

Sortie

Une seule ligne contenant au moins une action et au maximum une seule action `AUTOPLACE`.
Toutes les actions doivent être séparées par un point-virgule `;` et être l'une des suivantes :

- `PLACE_TRACKS` suivie des coordonnées de l'emplacement désiré.
- `AUTOPLACE` suivie de deux paires de coordonnées, pour créer le chemin le moins coûteux entre les deux.
- `DISRUPT` suivie du `regionId` de la région à perturber. *Remarque :* remplacez `regionId` par les coordonnées `x` , `y` pour perturber la région à cet emplacement.
- `MESSAGE` suivie d'un texte, à afficher dans le visualiseur.
- `WAIT`

Contraintes

Temps de réponse par tour ≤ `50` ms
Temps de réponse pour le premier tour ≤ `1000` ms

21 ≤ width ≤ 30
14 ≤ height ≤ 20
4 ≤ townCount ≤ 12

Pour Démarrer

Pourquoi ne pas se lancer dans la bataille avec l'un de ces **IA Starters**, fournis par l'équipe :

- C++ https://gist.github.com/CGjupoulton/bbd4b720ae7e0f1e5f2e970bb42ed066
- JavaScript https://gist.github.com/CGjupoulton/279a0f6ce4995bf4c55391fd40ae8bff
- Java https://gist.github.com/CGjupoulton/17397683833e324a7b8c7c7c642239d3
- Python https://gist.github.com/CGjupoulton/5a73bbd1142af98c6ca6887648b07cc2

Vous pouvez les modifier selon votre style, ou les prendre comme exemple pour tout coder à partir de zero.