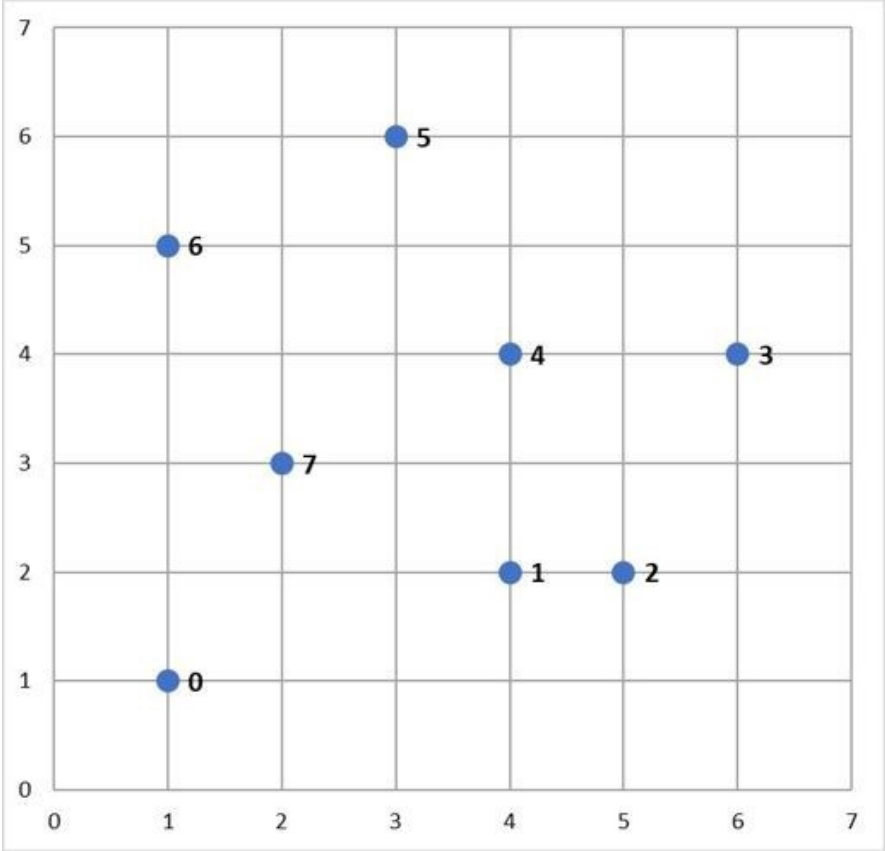
Projet : Le robot

Contexte

Vous êtes un petit robot. Vous devez faire le tour de tous les points pour réussir.

- Votre point de démarrage est défini (ici, ce sera 0)
- Vous devez passer par tous les points une fois
- Vous devez finir par le point de démarrage

Le but du jeu est de faire le circuit le plus court.



But du jeu

Vous devez implémenter trois algorithmes différents ainsi que deux fonctions utilitaires. Ces trois algorithmes ont les mêmes paramètres d'entrées, et renvoient la même chose (cf code).
Si vous êtes plus à l'aise, vous pouvez tout à fait coder le troisième algorithme avant le deuxième, mais il est conseillé de commencer par le premier.

Fonctions utilitaires

Il y a deux fonctions utilitaires à implémenter.

- Une fonction calculant la distance entre deux points
- Une fonction calculant la distance parcouru par votre robot dans le circuit

Vous pouvez aussi développer d'autres fonctions utilitaires. Par exemple :

- Une fonction générant des cas de tests
- Une fonction d'affichage de votre chemin

Premier algorithme : la naïveté

Le premier algorithme à implémenter est une résolution naïve :

En commençant par le premier point qu'on appellera *p0*, on se dirige vers le point le plus proche *p1*. Puis de *p1* on se dirige vers son point le plus proche non visité. On répète cela pour visiter tous les points. Puis on reviens à *p0* pour fermer le circuit.

```
NearestNeighbor(P)
    Pick and visit an initial point p0 from P
    p = p0
    i = 0
    While there are still unvisited points
        i = i + 1
        Select pi to be the closest unvisited point to pi-1
        Visit pi
    Return to p0 from pn-1
```

Deuxième algorithme : le meilleur !

Pour cette algorithme, fini la naïveté, nous cherchons une meilleure solution, sans pour autant parcourir toutes les différentes possibilités. Ici vous n'êtes pas guidé.

Le but est de faire un meilleure score que le premier, tout en étant moins long que le troisième. Soyez créatif !

Troisième algorithme : l’optimal

L’idée est de considérer et parcourir tous les chemins possible, pour retourner le circuit ayant le chemin le plus court. Différentes solutions s’offre à vous, mais une structure arborescente semble une bonne idée.

```
OptimalRobot(P)
    d = infinite
    For each of the n! permutations Pi of point set P
        if (cost(Pi) <= d) then d = cost(Pi) and Pmin = Pi
    return Pmin
```

Règles du jeu

En utilisant le code python fourni. Vous devez implémenter les différentes fonctions.

Collaboration

- Vous pouvez être un ou deux, mais pas trois
- Vous pouvez vous aider les uns les autres, en particulier dans la recherche du meilleur algorithme. Sans copier évidement.
- Vous pouvez vous échanger des fonctions/classes utilitaires non obligatoire. Mais si cela est fait, vous devrez le mentionner.

Respect du format

Tout manquement à ces règles sera sanctionné !

- Vous devez respecter le format d'entrée et de sortie
- Vous pouvez travailler dans d'autres fichiers
- Vous pouvez ajouter des tests, mais n'avez pas le droit de modifier les tests existants
- Vous devez coder en Python 3.6 ou inférieur
- Vous pouvez utiliser toutes les librairies intégrées au Python, à l'exception de pytest qui est autorisé, voir très fortement conseillé.
  Liste des librairies intégrées à Python 3.6 : https://docs.python.org/3.6/library/index.html
  Si vous avez besoin d'une autre librairie, contactez moi et on en parlera.

Livraison

Vous devez me partager, au plus tôt, votre code dans un repository privé sur github (https://github.com/sophisur) Afin que j'ai accès à tout l'historique de votre code.

La version utilisée pour la correction sera celui de la branche principale le 21/12/2019.
Une pré-correction sera faite lors d'une soutenance le 20/12/2019. Vous permettant d'éviter des petits accidents le jour du rendu.

Score final

Votre score final sera basé sur plusieurs éléments. Le barème est à titre indicatif, il vous sert de guide.

Est ce que tous les tests passent ? (14)

Tout simplement, est ce que tout fonctionne ? Plus ou moins bien ?

- Un algorithme : 5 points
- Deux algorithmes : 10 points
- Les trois fonctionnent : 14 points

Qualité du code : Le code respecte t'il les normes Python ? (4)

Ici je vais noter si vous respectez la norme PEP 8. Beaucoup d'IDE permettent de vérifier automatiquement si votre code respecte cette norme.

https://www.python.org/dev/peps/pep-0008/

Mais aussi la lisibilité du code, le nommage et la longueur de vos fonctions/méthodes...

La perfection ! La compétition ! (2)

Ici, je vais comparer la performance de votre algorithme, en temps, et sur la qualité du résultat.

- Le temps d'éxecution des trois algorithmes
- La qualité du résultat de votre "bonne algorithme"
- Ceux qui auront fait les meilleures fonctions utilitaires supplémentaires

Conseils

Une petite liste non exhaustive :

- Je vous conseille de coder en pair, au moins sur la mise en place du premier algorithme.
- N’attendez pas pour exécuter ! Dés que vous récupérez le code, avant même de commencer à coder. Allez y pas à pas, en exécutant au fur et à mesure. Le TDD est une bonne pratique https://fr.wikipedia.org/wiki/Test_driven_development
- Faites des recherches
- Posez moi des questions
- Mettez en place un environnement qui vous convienne, pour ma part, j’ai codé le projet sur Pycharm, en utilise la librairie pytest.
- Faites des dessins