Projet 2 : L'énigme des carrés

Présentation - objectif

Dans certains livres de jeux pour enfants, ont trouve des jeux à l'énoncé très simple : un dessin est présenté et l'enfant doit déterminer le nombre de triangles ou carrés dessinés.

On vous propose ici de faire jouer l'ordinateur à la version où l'on détermine le nombre de carrés : on trace des verticales et des horizontales et la question est de savoir combien de carrés ont été tracés sur la feuille.

Pour les verticales, on vous donne une liste d'abscisses entières (par exemple : 0, 2, 5 et 10) et pour les horizontales, une liste d'ordonnées entières (par exemple : 0, 3, et 5). Dans cet exemple, on a tracé 4 carrés :

- un de côté 2 (IJKL), construit sur les horizontales $y = 3$ et $y = 5$ et les verticales $x = 0$ et $x = 2$,
- un de côté 3 (MNOK), construit sur les horizontales $y = 0$ et $y = 3$ et les verticales $x = 2$ et $x = 5$,
- deux de côté 5 (IPNQ et QNTR), construits sur les horizontales $y = 0$ et $y = 5$ et les verticales $x = 0$ et $x = 5$ et
- sur les horizontales $y = 0$ et $y = 5$ et les verticales $x = 5$ et $x = 10$.

La réponse pour cet exemple est donc 4.

Vous devrez, par groupe de 2 ou 3, écrire des programmes qui donnent la réponse de façon automatique à ce jeu à partir des listes d'abscisses et d'ordonnées.

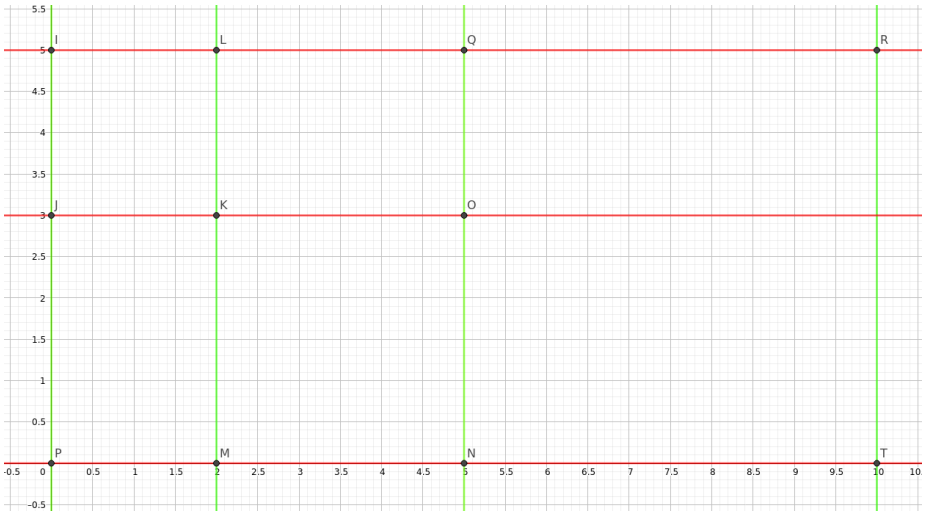

Figure 1: Exemple de jeu

Jeux de test

Les jeux de données sont fournis dans le fichier `donnees.py`. Vous ne devez pas modifier ce fichier, mais l'importer comme une bibliothèque avec la commande `from donnees import jeux`.

`jeux` est une liste de 9 jeux. chaque élément de la liste contient un jeu :

- la première liste contient les abscisses des verticales,
- la seconde contient les ordonnées des horizontales,
- la troisième contient la solution attendue, afin que vous puissiez tester la correction de vos algorithmes

Les valeurs des abscisses et ordonnées sont toutes positives.

Dans un premier temps, vous pouvez travailler avec un seul jeu de test, le premier de la liste, qui est aussi l'exemple proposé au début de ce document. Il peut vous permettre de mettre au point vos algorithmes.

Les jeux suivants serviront pour évaluer les performances de vos algorithmes.

Progression

Lors de ce projet, vous proposerez différents algorithmes afin d'améliorer les performances de ceux-ci.

Il est nécessaire de commencer par une version très simple sans essayer d'optimiser vos résultats, puis, progressivement, de modifier votre stratégie de résolution afin d'améliorer la complexité de vos algorithmes et donc leurs performances en temps. La complexité spatiale nous importe peu ici.

A rendre

Pour chaque version, vous proposerez :

- un algorithme (en français ou en pseudo-code)
- un programme python
- un calcul de complexité

Le travail sera à rendre en trois étapes; pour chacune, vous aurez un retour de l'étape précédente, d'où l'obligation de respecter les délais imposés. A chaque étape, vous pourrez proposer plusieurs versions de votre résolution.

Les étapes sont à rendre sur pronote pour les :

- 23 décembre (première version) - un fichier algorithmes et calculs de complexité, un fichier python de vos programmes sous forme de fonction
- 7 Janvier (seconde version) - un fichier algorithmes et calculs de complexité, un fichier python de vos programmes sous forme de fonction
- 24 Janvier (version finale) - un fichier algorithmes et calculs de complexité, un fichier python de vos programmes sous forme de fonction

Remarque : les programmes doivent calculer la solution, pas l'extraire du fichier `donnees.py` !

Comparaison de vos programmes

L'objectif est de construire progressivement des programmes plus rapides sur les jeux de données. Pour comparer vos algorithmes, vous pourrez comparer leurs complexités, mais aussi mesurer la durée de l'exécution des programmes associés grâce aux programmes du fichier `mesures.py`. Vous l'importerez grâce à la commande `from mesures import *`. Cela vous permettra alors d'utiliser la fontion `mesure(algo,jeu)` où `algo` est la fonction dont vous souhaitez connaître la durée d'exécution en nanoseconde et `jeu`, le jeu de test sur lequel vous souhaitez connaître cette durée.

Pour utiliser ce fichier, vos fonctions doivent avoir un prototype de la forme : `def algo(jeu)` où algo est votre fonction implantant l'algorithme à tester et `jeu` une liste telle que proposée dans `donnees.py`: `[x,y,solution]` où `x` est la liste des abscisses des verticales, `y` la listes des ordonnées des horizontales et `solution`, le nombre des carrés.

Evaluation des productions

Ce projet est un challenge entre vous, mais surtout contre vous-même : vous pouvez comparer vos différents programmes et déterminer lequel est le plus rapide pour l'ensemble des jeux de test.

D'autre part, ce projet sert d'évaluation sur la partie algorithmique et la rédaction des algorithmes en français, la spécification (docstring) des fonctions produites, les commentaires du code produit, mais aussi les calculs de complexité seront évalués.

Chaque groupe doit proposer au moins deux algorithmes distincts, avec leur code et le calcul de leur complexité.

Travail de l'oral

Pour le 31 Janvier, vous déposerez sur un lien fourni dans pronote une vidéo de 3 minutes au maximum dans lequel vous présenterez votre algorithme le plus performant. Vous expliquerez ses points forts et ses éventuels point faibles. La clarté de l'exposé ainsi que votre capacité à convaincre de la proposition faite seront mises en valeur.