INFO-F103 – Algorithmique 1
Projet 1 : Polyominos

Nil Fernandez Lojo

Année académique 2019-2020

Énoncé

L'objectif de ce projet est d'implémenter en python un jeu utilisant des polyominos et d'écrire un programme qui puisse trouver une solution optimale. Un polyomino est une forme composée de n carrés unitaires adjacents. La liste de polyominos de taille inférieure ou égale à 5 est donnée dans la figure 1. Par exemple si $n = 4$, ils sont dénommés tétraminos et sont les pièces utilisées dans le jeu tetris. Le jeu que vous devez implémenter, consiste à trouver le rectangle à l'aire la plus petite dans lequel entre un ensemble de polyominos donné. Chaque polyomino peut subir des rotations de 90 degrés et une réflexion sur son axe vertical ou horizontal avant d'être placé sur le tableau.

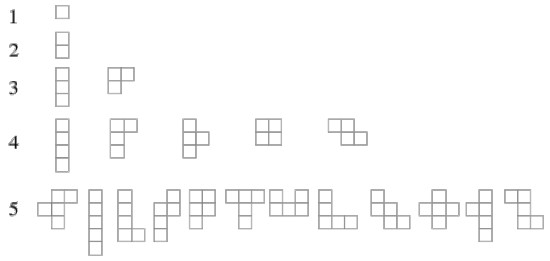


Figure 1: Liste de polyominos de taille inférieure ou égale à 5[1].

Lorsque l'ensemble de polyominos à placer dans le rectangle contient exactement 2 copies de chacun des 5 tétraminos, il existe 156864 [2] rectangles différents d'aire minimum qui peuvent contenir ces pièces. Ces rectangles sont de dimensions 4×10 ou 8×5. Deux exemples de solutions sont donnés dans la figure 2.

Dimensions 8×5. Dimensions 4×10

Figure 2: Exemples de solutions lorsque l'ensemble de pièces à placer contient exactement 2 copies des 5 tétraminos[2].

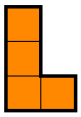
[1] Source: `http://mathworld.wolfram.com/Polyomino.html`
[2] Source: `https://en.wikipedia.org/wiki/Tetromino`

Veuillez noter que l'aire de la solution est égale ou supérieure la taille totale des polyominos à placer. S'il n'est pas possible de les placer dans un rectangle de cette taille, la solution optimale va contenir des carrés vides. Par exemple, un rectangle ne peut pas être formé par une seule copie de chaque tétraminos (taille totale des pièces = 20). La solution optimale sera un tableau de dimension 3×7 avec un carré vide. La stratégie recommandée pour ce projet est de d'abord essayer de faire rentrer les polyominos dans un tableau d'aire égale à la taille des pièces. Si cela n'est pas possible, augmentez l'aire totale de votre tableau d'une unité et ressayez de faire rentrer les pièces dans le tableau. Réitérez jusqu'à trouver une solution.

Le programme que vous devez implémenter doit trouver toutes les solutions à dimensions différentes mais une seule solution par dimension de rectangle. Par exemple, si pour un ensemble de pièces donné, le plus petit rectangle qui puisse les contenir a une aire de 20 carrés et qu'il existe 15 solutions différentes, 10 formant un rectangle de dimensions 4×5 et 5 formant un rectangle de dimensions 2×10, le programme ne devrait donner qu'une solution de dimensions 4×5 et une solution de dimensions 2×10. De plus, la rotation de 90 degrés d'une solution ne constitue pas une nouvelle solution, dans l'exemple précédent, cela veut dire que le programme ne devrait pas donner de solutions de dimensions 5×4 ni 10×2. Dans ce travail, on utilisera la convention suivante: la hauteur du tableau doit être plus courte que sa largeur. Par exemple, la solution sera de dimensions 4×5 et pas 5×4.

Structure Input

Chaque polyomino est représenté par un nom qui est un caractère ASCII et une matrice carrée binaire. Pour un ensemble de pièces, la taille de la matrice carrée est la même pour chaque pièce. Par exemple une représentation possible d'une pièce en forme de L est donnée dans la figure 3.



Tétramino en forme de L

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

Possible représentation matricielle

Figure 3: Exemple de représentation matricielle d'un polyomino.

Les fichiers contenant une liste de polyominos ont une extension .poly et leur structure est la suivante:

- La première ligne contient deux nombres séparés par un espace. Le premier nombre N indique le nombre de polyominos dans le fichier et le deuxième nombre T indique la taille des matrices.
- La ligne 2 contient le nom du premier polyomino (un caractère ASCII). Les lignes 3 à $T+2$ contiennent la matrice carrée binaire du premier polyomino où les nombres sont séparés par un espace.
- La ligne $(n-1)T+n+1$ contient le nom du polyomino numéro n. Les lignes $(n-1)T+n+2$ à $nT+n+1$ contiennent la matrice carrée binaire du polyomino numéro n.

Un exemple de fichier contenant 5 polyominos représentés par des matrices de taille 3 est montré dans la figure 4.

```
5 3
a
1 1 0
1 0 0
0 0 0
b
1 1 1
0 0 0
0 0 0
c
0 1 0
1 1 1
0 1 0
d
1 0 0
1 1 1
0 0 0
e
1 0 0
1 1 1
0 1 0
```

Figure 4: Exemple de fichier .poly

Trois exemples de fichiers .poly vous sont donnés. Vous êtes encouragés à en créer d'autres mais ne modifez pas ceux fournis car ils seront utilisés dans les tests unitaires.

Structure Output

Le format d'impression d'un tableau est donné dans la méthode `imprimer(self)` de la classe Tableau. Ne modifiez pas cette méthode car ce format sera utilisé pour la correction de votre code.

Si plusieurs solutions de dimensions différentes existent, imprimez les en ordre croissant de leur hauteur. Par exemple, si des solutions de dimensions 2×10 et 4×5 existent, imprimez la solution de dimensions 2×10 et ensuite celle de 4×5. N'imprimez rien d'autre et n'insérez pas de lignes vides entre les différents tableaux.

Implémentation

Un template du code vous est fourni sous le nom *Polyominos.py*. Il contient toutes les classes, méthodes et fonctions nécessaires pour la résolution de ce projet. Vous pouvez en utiliser d'autres si vous le désirez mais celles qui sont dans le template doivent rester présentes dans votre code. Certains tests unitaires pour celles-ci sont fournis dans le fichier *tests_unitaires.py*. Le script prend un argument qui est le numéro du test (nombre entre 1 et 13). La description des tests est donnée dans les premières lignes du script. Uniquement certains cas sont testés, vous êtes encouragés à tester plus de cas. Il est important que vous respectiez la description fournie dans le template pour ces classes et méthodes car une partie de votre note sera déterminée par des tests unitaires sur vos fonctions et méthodes.

En dehors des définitions de classes et fonctions, vos lignes de code doivent être précédés du statement `if __name__ = '__main__':`. Ceci évitera de lancer votre script en entier lors des tests unitaires.

Vous n'êtes pas autorisés à utiliser des librairies python sauf `sys`.

Consignes générales

Tout votre code doit être contenu dans un seul fichier *Polyominos.py*. Veuillez respecter le nom, le format des fonctions et du nom du fichier. Votre fichier doit pouvoir se lancer en ligne de commande avec la commande suivante :

```
python3 Polyominos.py data.poly
```

Veillez à ce que votre code soit commenté de manière concise pour mettre en valeur ses fonctionnalités et les optimisations appliquées.

La code doit être soumis sur l'UV avant le 8 mars à 22h. Tout manquement aux consignes ou retard sera sanctionné directement d'un 0/10.

Si vous avez des questions, vous pouvez envoyer un mail à l'adresse nil.fernandez.lojo@ulb.be.