TP C#14 : Packing Challenge

Consignes de rendu

À la fin de ce TP, vous devrez rendre un dépôt git respectant l'architecture suivante :

```
csharp-tp14-prenom.nom
    |-- README
    |-- PackingChallenge/
        |-- PackingChallenge.sln
        |-- PackingChallenge/
            |-- Everything except bin/ and obj/
```

N'oubliez pas de vérifier les points suivants avant de rendre :

— Remplacez *prenom.nom* par votre propre login.
— Votre code **DOIT** compiler !
— Le fichier `README` est obligatoire.
— Pas de dossiers `bin` ou `obj` dans le projet.

README

Vous devez écrire dans ce fichier tout commentaire sur le TP, votre travail, ou plus généralement vos forces et vos faiblesses. Vous devez lister et expliquer tous les bonus que vous aurez implémentés. Un `README` vide sera considéré comme une archive invalide (malus).

1 Introduction

1.1 Objectifs

Cette fois vous allez être évalués sur votre capacité à écrire du code et à penser par vous-même. Nous vous donnons un problème à résoudre et vous devez essayer de le résoudre de la meilleure façon possible. De ce fait, vous pouvez utiliser toutes les notions que vous avez apprises cette année et nous ne vous imposons pas d'implémentation spécifique quant à la résolution du problème.

2 Cours

2.1 Fichiers (Rappel)

Vous pouvez lire des fichiers de la manière suivante :

```
try
{
    using (StreamReader sr = new StreamReader("InFile.txt"))
    {
        string line;
        while ((line = sr.ReadLine()) != null)
        {
            Console.WriteLine(line);
        }
    }
}
catch (IOException e)
{
    Console.WriteLine("The file could not be read:");
    Console.WriteLine(e.Message);
}
```

Vous pouvez écrire des fichiers de la manière suivante :

```
try
{
    using (StreamWriter sw = new StreamWriter("OutFile.txt"))
    {
        sw.WriteLine("Hello World!");
    }
}
catch (IOException e)
{
    Console.WriteLine("The file could not be written:");
    Console.WriteLine(e.Message);
}
```

3 Exercices

3.1 Problème

Plusieurs pièces sont données à votre programme. Ces pièces ont des formes spécifiques et le but est d'en placer le plus possible sur un plateau.

3.1.1 Les formes

Une forme est représentée par une hauteur et une largeur. Chaque pixel à l'intérieur d'une forme a deux états possibles : "on" ou "off".

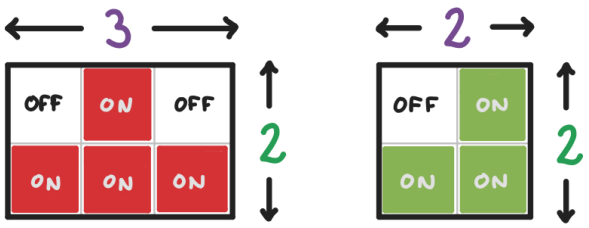

FIGURE 1 – Exemple de formes

Une forme peut pivoter par multiples de 90°.

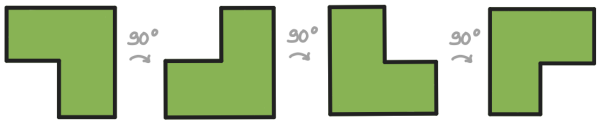

FIGURE 2 – Exemple de rotations possibles

3.1.2 Les pièces

Une pièce correspond à une forme. Une forme peut donc générer plusieurs pièces.

3.1.3 Le plateau

Le plateau a une forme rectangulaire et est représenté par une largeur et une hauteur.

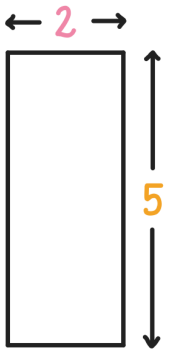

FIGURE 3 – Exemple de plateau

3.2 Format d'entrée

L'entrée est un fichier qui contient des valeurs séparées par des espaces.

Dans la première ligne vous trouverez :
- Le nombre de formes uniques
- La largeur du plateau
- La hauteur du plateau

Chaque prochaine ligne correspond à une forme. Dans ces lignes vous trouverez :
- Le nombre de pièces de cette forme
- La largeur de cette forme
- La hauteur de cette forme
- Les états (on/off) des parties de cette forme d'en haut à gauche à en bas à droite.

Par exemple, cette entrée

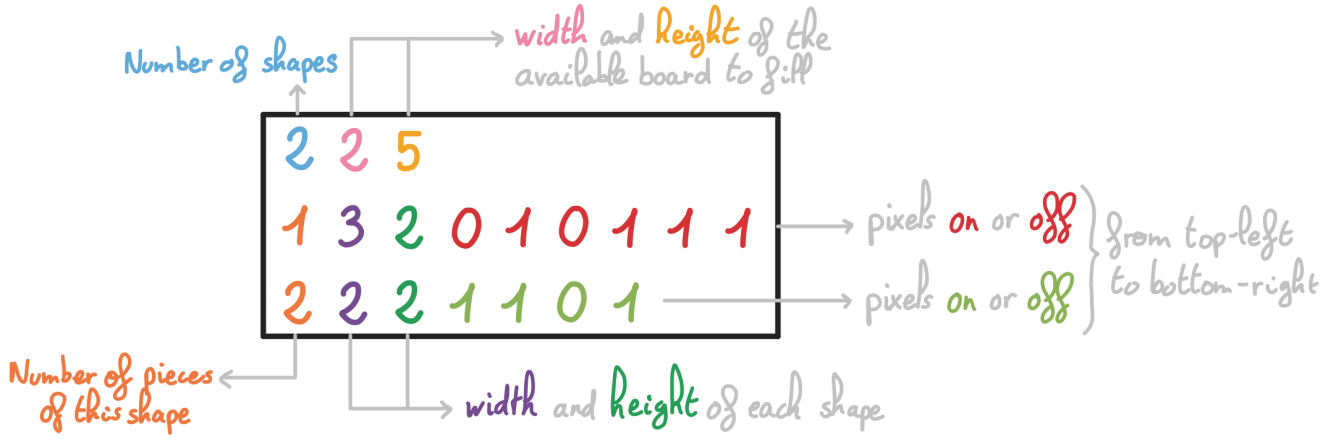

FIGURE 4 – Exemple de fichier d'entrée (basic.in)

Vous donne ces pièces et ce plateau :

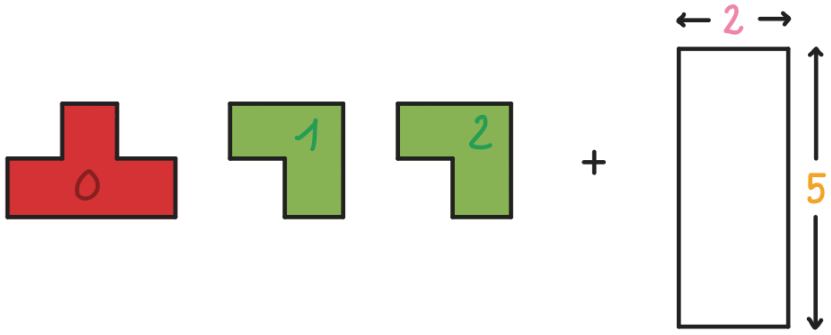

FIGURE 5 – Exemple d'entrée

Info

Votre code ne sera PAS testé avec des fichiers d'entrée invalides.

Important !

Chaque pièce reçoit un identifiant unique. Il DOIT commencer à 0 et être positif.
L'ordre des identifiants n'a pas d'importance : seule leur propriété unique est importante.

3.3 Fichier de sortie

La sortie doit être un fichier contenant des valeurs séparées par un seul espace. Chaque valeur correspond à l'identifiant unique d'une pièce placée dans le tableau final.

Avec notre exemple, ce résultat :

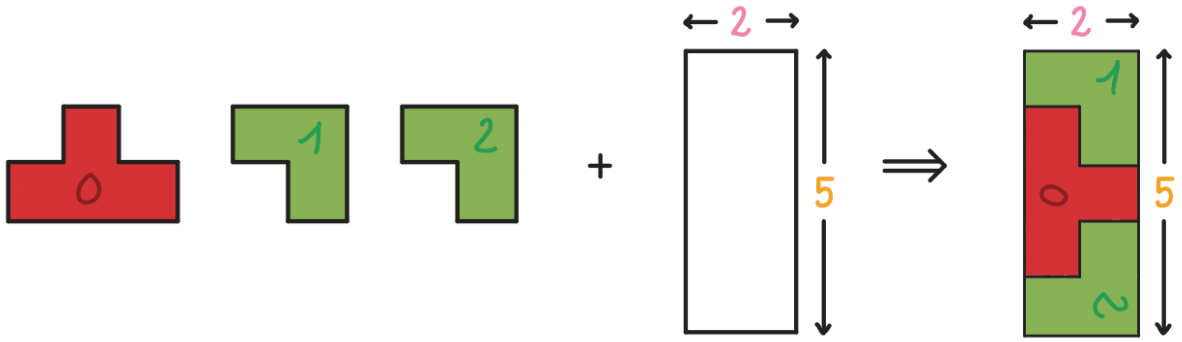

FIGURE 6 – Exemple de sortie

Vous donnera cette sortie :

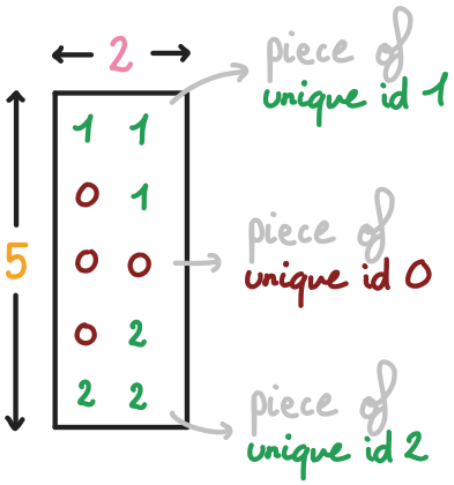

FIGURE 7 – Exemple d'un fichier de sortie (basic.out)

Important !

Un pixel qui ne contient aucune pièce DOIT être représenté par la valeur -1.

3.4 Notation

Contrairement à tous les autres projets que vous avez réalisés cette année, celui-ci sera noté à la fois en fonction de l'évaluation de votre code et du résultat de votre soumission.

3.4.1 Évaluation du code

L'évaluation du code se fera comme d'habitude mais étant donné que vous êtes libre de choisir l'implémentation de votre choix, nous vous conseillons vivement de commenter et de partitionner votre code autant que possible. De cette façon, il sera plus facile pour le correcteur de comprendre et de noter votre travail.

Un travail plus propre et mieux organisé est synonyme de meilleure note.

3.4.2 Évaluation du programme

D'autre part, la notation de votre rendu reflétera à quel point votre solution est optimale. Il existe plusieurs ensembles de données représentant des instances distinctes du problème. Le score de chaque ensemble de données sera calculé de la manière suivante : (Nombre total de pixels du tableau) - (Nombre de pixels du tableau non remplis par une pièce). Votre score final correspondra à la somme de vos meilleurs scores sur les ensembles de données individuels. Vous pourrez faire un rendu à tout moment pour voir le score obtenu.

Important

Si votre code ne compile pas, vous n'obtiendrez aucun point !

Attention

L'exécution de votre programme sera restreinte par une contrainte de temps. Cela signifie que votre algorithme devra être suffisamment rapide et qu'il ne pourra pas tourner indéfiniment. Le temps maximal autorisé sera différent pour chaque ensemble de données testées.

3.4.3 Lire et écrire

Le fichier d'entrée se trouvera exactement dans le même répertoire que le binaire (programme actuel) et s'appellera "input.in". Pour écrire votre solution, vous devez également créer (et remplir) un fichier nommé "output.out" dans le même répertoire.

3.5 Conseil

Le problème est difficile, mais faisable. Avant de coder, prenez le temps de bien comprendre le sujet et réfléchissez à la manière dont vous allez organiser votre code. Coupez le travail en petites parties sur lesquelles vous pouvez facilement vous concentrer.

3.5.1 Comment démarrer

Vous êtes libre de faire comme vous voulez, mais si vous ne voyez pas comment commencer, considérez le programme comme un casse-tête comportant plusieurs parties. Une partie est l'endroit où vous lirez le fichier d'entrée. Un autre est l'endroit où vous allez écrire le fichier de sortie. Ainsi, commencez simplement par créer un programme capable de lire le fichier d'entrée. Et puis, implémentez la partie où le programme écrit le fichier de sortie. Après cela, vous pouvez vous concentrer sur la résolution du problème.

3.5.2 Que faut-il utiliser

Vous pouvez utiliser tout ce que vous avez appris cette année. En plus des StreamReader/StreamWriter, il serait judicieux d'utiliser les objets. Avec ceux-ci, vous pouvez représenter un plateau, une pièce, une solution, etc.

4 Bonus

4.1 Visualisation en SVG

Si vous êtes un génie de la programmation et que votre solution est parfaite, vous pouvez essayer de visualiser le résultat de votre programme en générant un fichier au format svg. Des informations sur ce format peuvent facilement être trouvées sur Internet.

I see no point in coding if it can't be beautiful.