Reconstruction de puzzles

Sujet proposé par Philippe Chassignet

Philippe.Chassignet[at]polytechnique.fr

1 Introduction

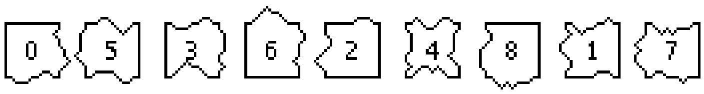


Figure 1: Les pièces d'un puzzle

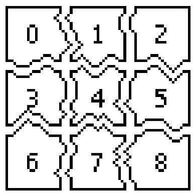


Figure 2: Le puzzle reconstruit

On se donne les pièces d'un puzzle et il s'agit de le reconstruire. Pour plus de clarté, dans l'exemple des figures 1 et 2, les pièces sont numérotées mais en réalité on n'aura pas cette information. Les pièces sont décrites par leurs contours et la reconstruction du puzzle va donc consister à apparier ces contours pour trouver les pièces adjacentes.

Les difficultés de ce sujet résident essentiellement dans la représentation des contours et dans la manière de rechercher une pièce correspondant à un contour donné. La méthode proposée est basée sur un codage simplifié des contours, ce qui restreint un peu la forme des pièces, par rapport à celles d'un puzzle classique.

Pour de grands puzzles, un dessin détaillé comme celui de la figure 2 ne tiendrait pas à l'écran. Chaque pièce correspondra à un pixel d'une image, avec un niveau de gris, et en remettant les pièces au bon endroit on pourra reconstituer cette image pixel par pixel. C'est alors cette image qui sera affichée et on peut facilement envisager des puzzles de 256×256 pièces, voire plus. Les moyens pour construire et visualiser des images sont indiqués dans une page web annexe.

L'entrée du programme sera un fichier qui donne une description des pièces et on devra produire deux sortes d'affichages, un dessin du style de celui de la figure 2, pour des petits exemples, et l'image en niveaux de gris pour les plus gros. Un certain nombre de ces fichiers sont donnés dans la page web annexe.

2 Détail du sujet

2.1 Codage d'un contour

On considère 3 déplacements élémentaires, codés respectivement `+ = -` :

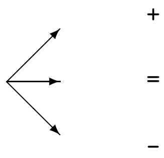
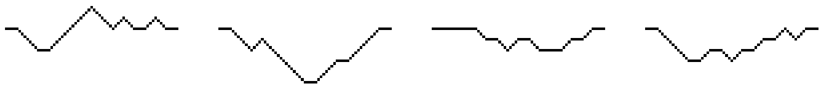


Figure 3: Les 3 déplacements élémentaires

Si on le généralise à 8 directions, ce type de codage est appelé codage de Freeman et il permet de décrire toute courbe discrète. Un mot sur l'alphabet {`+ , = , -`} permet de décrire un côté "horizontal", implicitement orienté de la gauche vers la droite, la figure 4 nous en donne quatre exemples.



`=--=++++--+-=+-=`   `=--+----=++=+++=`   `====-=-+=-==+=+=`   `=---=+=-+=+=+-+=`

Figure 4: Exemples de codage des côtés

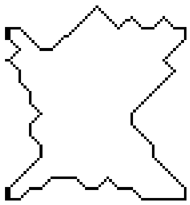


Figure 5: Dessin de la pièce correspondante

Les 3 directions `+ = -` nous suffisent. Une pièce du puzzle sera décrite par ses 4 côtés, enroulés dans le sens des aiguilles d'une montre. Ainsi la suite des 4 mots donnés figure 4, pris dans cet ordre, correspond à la pièce de la figure 5. On remarquera qu'une permutation circulaire des 4 mots décrirait la même pièce mais tournée d'un certain nombre de quarts de tours.

2.2 Format d'un fichier puzzle

Un fichier de description d'un puzzle commence par une ligne qui donne 3 entiers `N`, `M`, `L`, séparés par un espace. Il s'agit respectivement de la largeur et de la hauteur du puzzle (en nombre de pièces) et de la longueur d'un mot.

Viennent ensuite `N`×`M` lignes, chacune décrivant une pièce. Chaque ligne comporte ainsi 4 mots de longueur `L` sur l'alphabet {`+ , = , -`} puis un entier qui sera le numéro de gris. Ces éléments sont séparés par un espace. Les pièces sont données dans un ordre et avec une orientation quelconques, sauf la première pièce qui est toujours celle à placer en haut et à gauche, ce qui permet d'orienter le puzzle.

L'exemple introductif correspond au fichier suivant (avec le numéro de pièce en guise de numéro de gris) :

```
3 3 10
========== =-+=+=+--= =-=+=+==-= ========== 0
==+=---++= ========== =+=--===+= ==++-===-= 5
=-==+=+=-= =+===-=+-= ==--=-+++= ========== 3
=+++-=--== =+=-==-+== ========== ========== 6
=====+==-= ========== ========== =++---=+== 2
=+--=++=-= =-===-++== ==--+-=++= =-+=-===+= 4
========== ========== =+==+-+--= =+===--=+= 8
=--+=+=+-= ========== =-==+===== =-=++=--+= 1
=--+-+==+= ========== ==+-==-=+= =++=-+--== 7
```

On pourra utiliser le fait qu'il n'y a pas de côté plat à l'intérieur du puzzle.

2.3 Reconstruction du puzzle

La reconstruction commencera par placer la première pièce en haut à gauche, on sait que c'est la première dans le fichier. On placera ensuite les autres de proche en proche. Cela revient à trouver une pièce dont un des côtés coïncide avec celui considéré sur une pièce déjà placée, placer cette nouvelle pièce, etc. Pour orienter convenablement chaque pièce, il faut aussi identifier le côté qui a permis de la retrouver.

On a donc besoin d'une structure de table, dans laquelle chaque clé est formée avec le code d'un côté et les informations associées donnent la référence de la pièce et le rang de ce côté dans cette pièce.

En fait, ceci ne fonctionne que si l'association est bijective, c'est-à-dire, s'il n'y a pas deux pièces ayant un côté de même code (autre que le pourtour). De manière plus générale, la table devra donc associer plusieurs pièces à une même clé et on devra explorer plusieurs possibilités de placement. Pour limiter le coût de l'exploration, il faut utiliser au mieux les contraintes connues.

Pour traiter des gros puzzles, il faudra une structure de table efficace.

3 Travail demandé

Pour ce projet, il s'agira donc d'écrire un programme qui lit un fichier de description d'un puzzle et qui le reconstruit. Ce programme aura deux modes (on peut aussi faire deux programmes séparés) :

- visualisation dans le style de la figure 2, on ne l'utilisera que pour des puzzles relativement petits,
- visualisation de l'image définie par ses pixels.

4 Extensions possibles

Voici une suggestion de trois extensions pour ce sujet. La première est un préalable aux les deux autres.

- Construction d'un puzzle : il s'agit de produire, dans le format donné ci-dessus, un fichier de description pour un puzzle réalisable.
- Généralisation des formes : il s'agit de compliquer les formes des pièces en s'autorisant le code de Freeman complet. La difficulté est de produire des pièces consistantes.
- Puzzle en couleur : l'entier en fin de chaque ligne est remplacé par un triplet donnant les composantes rouge-vert-bleu de chaque pixel.

5 Annexe

cf. http://www.enseignement.polytechnique.fr/profs/informatique/Philippe.Chassignet/04-05/PUZZLES/index.html