Strip Packing

Algo1

6 octobre 2014

1 Introduction

On s'intéresse dans ce projet à la résolution d'un problème connu : le problème du strip packing.

On dispose d'un ruban, de largeur fixe, donnée ; et de longueur infinie. On considère également un ensemble d'objets rectangulaires, chacun défini par un couple (largeur,hauteur). Le problème consiste à trouver un placement des objets sur le ruban, sans effectuer de rotations et sans que deux objets se chevauchent. On cherche à minimiser la longueur de ruban utilisée.

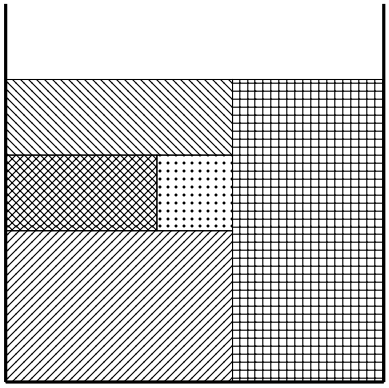

FIGURE 1 – Exemple de strip packing (solution optimale)

Il n'existe pas à l'heure actuelle d'algorithmes rapides de résolution exacte de ce problème qui a néanmoins de nombreuses applications industrielles.

On vous demande pour ce projet de :

1. lire un fichier contenant des objets ;
2. calculer un placement sur le ruban ;
3. afficher le placement à l'aide d'un fichier svg.

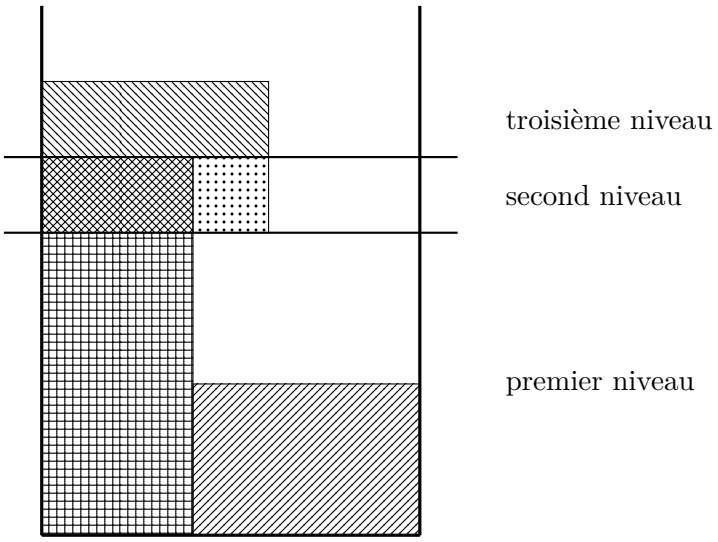

FIGURE 2 – Exemple de strip packing par Next Fit Decreasing Height

2 Next Fit Decreasing Height

L'algorithme *Next Fit Decreasing Height* fonctionne en remplissant le ruban niveau par niveau. Tous les objets d'un même niveau sont alignés horizontalement sur une même hauteur de base. Les "hauts" des objets d'un niveau en revanche ne sont pas forcément alignés ; chaque objet pouvant être d'une hauteur différente. Au sein d'un niveau, les objets sont placés successivement de gauche à droite jusqu'à ce que l'espace libre sur la droite soit trop petit pour accueillir l'objet suivant, ce qui clot le niveau.

Données : Objets ; Largeur du ruban
Trier les objets par hauteur décroissante;
pour chaque *objet* **faire**
  si *l'objet ne peut pas rentrer dans le niveau courant* **alors**
    Passer au niveau suivant;
  fin
  Placer l'objet à la fin du niveau;
fin

Algorithme 1 : Next Fit Decreasing Height

3 Benchmarks

Il existe de nombreux benchmarks permettant de comparer les performances de différents algorithmes.

Nous vous fournissons le benchmark *ZDF*[1] de l'*ESICUP*.
Chaque fichier est structuré de la manière suivante :

- la première ligne contient le nombre d'objets contenus dans l'instance ;
- la seconde ligne contient la largeur du ruban ;
- chaque ligne suivante contient un objet :
  - le numéro identifiant l'objet ;
  - la largeur de l'objet ;
  - la hauteur de l'objet.

4 SVG

La structuration du fichier SVG est libre. On doit pouvoir voir d'un coup d'œil l'ensemble des objets placés sur le ruban. Il est sans doute intéressant d'avoir un peu de couleur pour les différents objets.

5 Fichiers fournis

Nous vous fournissons le fichier principal du programme :

```
--paquetages classiques
with Ada.Command_line;
use Ada.Command_Line;
with Ada.Text_IO;
use Ada.Text_IO;

--paquetages "maison"
with Objet_Packing;
use Objet_Packing;
with Parseur;
with Packing;
with Svg;

procedure Strip is
        Nombre_Objets : Natural;
        Largeur_Ruban : Natural;
        Hauteur_Ruban : Natural;
begin

        if Argument_Count /= 2 then
                Put_Line(Standard_Error, "utilisation : strip benchmark.txt packing.svg");
                return;
        end if;

        Parseur.Lecture_En_Tete(Argument(1), Nombre_Objets, Largeur_Ruban);

        declare
                Objets : Tableau_Objets(1..Nombre_Objets);
        begin
                Parseur.Lecture(Argument(1), Objets);
                Packing.Next_Fit_Decreasing_Height(Objets, Largeur_Ruban, Hauteur_Ruban);
                Svg.Sauvegarde(Argument(2), Objets, Largeur_Ruban, Hauteur_Ruban);
        end;

exception
        when Parseur.Erreur_Lecture_Benchmark
                => Put_Line(Standard_Error, "le fichier d'entree est illisible");
end;
```

1. http://paginas.fe.up.pt/~esicup/tiki-list_file_gallery.php?galleryId=3

On peut imaginer que ce fichier a été réalisé par votre chef de projet qui est désormais parti en vacances en vous demandant de terminer le programme.

Attention ! Il vous est demandé de *ne pas modifier* le code existant. Le fichier *strip.adb* en lui même est assez clair mais vous ne disposez malheureusement pas des fichiers *.ads* correspondants aux paquetages maison à réaliser.

Nous vous demandons explicitement de porter une attention particulière à contraindre au maximum les prototypes de fonctions et procédures. Interdiction donc de mettre en *in out* des arguments qui ne sont pas modifiés par exemple.

Nous vous posons également comme contrainte de déclarer l'enregistrement d'un objet comme un type privé (à l'aide du mot clef *private*). De cette manière un utilisateur externe du paquetage fournissant les objets ne peut utiliser directement les structures internes au paquetage.

Enfin, essayez dans la mesure du possble de remonter l'exception prévue en cas fichier de benchmark incorrect.

6 Travail attendu

6.1 Code

Il est important de garder un code le plus clair possible et la clarté de ce que vous écrivez sera le critère le plus important pour la notation. N'hésitez donc pas à structurer votre projet correctement : pas de procédures trop longues ; des commentaires sur les parties complexes ; des noms de variables faisant sens ; un code indenté (attention : soit tabulations soit espaces) ; des paquetages permettant de bien organiser le projet.

Attention, n'attendez pas d'avoir terminé votre projet pour commencer à déboguer. Il vaut mieux vérifier au fur et à mesure ce que l'on fait afin de construire sur du solide. N'hésitez donc pas à créer des programmes de test.

Vous rendrez une archive contenant le code source et le rapport via *TEIDE*. Attention à ne pas inclure les fichiers issus de la compilation dans l'archive, les correcteurs recompileront votre code source pour obtenir l'exécutable. Attention aussi aux fichiers cachés (ceux dont le nom commence par un ".")

6.2 Rapport

Nous vous demandons 2 pages (max) de rapport au format *pdf* que vous incluerez dans l'archive que vous rendrez électroniquement sur *TEIDE*.

Le rapport a pour objectif de permettre aux correcteurs une plongée plus facile dans votre code.

6.3 Rendu et évaluation

Le projet est à rendre au plus tard le vendredi 23 octobre 2014 à 23h59. Tout retard est sujet à une pénalité.

L'accent du projet est mis sur la clarté. Il est possible de réaliser plus de fonctionnalités que prévues pour le projet (plus d'algorithmes, meilleur rendu svg ...). Néanmoins ces fonctionnalités ne seront pas un critère déterminant de votre note. C'est par contre l'occasion de vous faire plaisir ou de profiter du retour des correcteurs sur votre programme.

6.4 FAQ

Est-il possible de réaliser le projet seul ?

Oui, mais nous souhaiterions limiter le nombre de projets à corriger et donc nous n'accepterons que très peu de demandes.

Est-il possible de réaliser le projet à 3 ?

Non.

Est-il possible de réaliser le projet avec quelqu'un d'un autre groupe ?

Éventuellement. Préférez plutôt quelqu'un de votre groupe.

Est-il possible de modifier les fichiers .ads et .adb fournis ?

Non. Pour ce projet, nous vous fournissons la contrainte forte de ne pas modifier le fichier fourni. De plus vous devez impérativement déclarer la structure codant un objet comme une structure privée (private) au paquetage *objet_packing*.

Est-il possible de prendre du code sur internet ?

Oui, dans une certaine mesure. Il est possible de s'inspirer des exemples du cours. Vous pouvez également profiter des procédures fournies par Ada (pour éviter de re-programmer vous même un tri par exemple). Enfin vous pouvez vous servir de *rosettacode*. Un emprunt de quelques lignes ne porte pas à conséquence (en cas de doute, merci de signaler dans un commentaire l'origine du code). Par contre il est interdit de reprendre du code lié au packing.

Est-il possible de prendre du code d'un autre groupe d'étudiant ?

Non. En cas de copie, les 2 groupes (le copieur et le copié) auront 0. Il est donc à charge de chacun de sécuriser ses fichiers.