INFO-F103 – Algorithmique 1
Projet 1 : *Bin packing*

Jérôme De Boeck

Année académique 2017–2018

Enoncé

Le problème de *bin packing* consiste à emballer un ensemble d'objets donné dans un ensemble de boîtes, le but étant de minimiser le nombre de boîtes utilisées. Ce problème est semblable au problème de knapsack, chaque objet a un poids donné et chaque boîte peut contenir un poids maximum. Toutes les boîtes disponibles pour emballer des objets ont la même capacité et il n'est pas nécessaire que les boîtes utilisées soient remplies au maximum de leur capacité dans une solution du bin packing. Comme exemple d'instance, considérons dix objets respectivement de poids 1,3,4,4,5,6,7,8,8,10 et des boîtes de taille 20. Comme solutions admissibles nous avons par exemple :

- Solution 1 :
  - Boite 1 : 1,3,4,4,5 $\longrightarrow$ poids = 17
  - Boite 2 : 6,7 $\longrightarrow$ poids = 13
  - Boite 3 : 8,8 $\longrightarrow$ poids = 16
  - Boite 4 : 10 $\longrightarrow$ poids = 10
- Solution 2 :
  - Boite 1 : 1,4,5,8 $\longrightarrow$ poids = 18
  - Boite 2 : 3,4,6,7 $\longrightarrow$ poids = 20
  - Boite 3 : 8,10 $\longrightarrow$ poids = 18

La solution 2 utilise 3 boîtes et est meilleur que la solution 1 qui en utilise 4. Etant donné le poids total des objets à emballer et les tailles des boîtes disponibles, on peut facilement prouver qu'il est impossible d'utiliser moins de 3 boîtes.

Implémentation

Nous vous demandons d'écrire une fonction `bin_packing(file)` dans un fichier `projet1.py` où `file` est un nom de fichier de données contenu **dans le même dossier** que `projet1.py` (vous pouvez supposer que l'instance passée en paramètre existe bien dans le dossier). Cette fonction doit résoudre le problème de bin packing pour l'instance `file` et commencera par afficher les données du problème. Durant son exécution, dès qu'une nouvelle meilleure solution est trouvée, la fonction doit afficher le nombre boîtes utilisées. A la fin de son exécution, elle

affichera la meilleure solution ainsi que le temps total de résolution. L'exécution sur l'instance donnée ci-dessus pourrait afficher sur le terminal :

```
Capacité des boites : 20
Nombre d'objets : 10
Poids des objets : 1,3,4,4,5,6,7,8,8,10
Résolution...
Solution à 4 boites trouvée...
Solution à 3 boites trouvée...
Meilleure solution :
3 boites
Boite 1 : 1 4 5 8 ; total : 18
Boite 2 : 3 4 6 7 ; total : 20
Boite 3 : 8 10 ; total : 18
Temps de résolution : 0.0000015 s
```

Vous êtes libre d'afficher d'autre informations pertinentes à la bonne exécution de votre algorithme.
Lors de l'appel depuis un terminal du fichier `projet1.py`, votre code appellera la fonction `bin_packing` sur une instance dont le nom sera passé en paramètre (vous pouvez supposer que l'utilisateur encodera un nom d'instance correcte), par exemple :

```
python3 projet1.py bin20-10.txt
```

Vous êtes libre d'utiliser d'autres fonctions. Veillez à ce que votre code soit efficace d'un point de vue algorithmique, c-à-d que la bonne réponse soit retournée et que les variables et opérations qui n'apportent rien à l'efficacité de vos fonctions soient évitées. Seront évalués : la lisibilité du code, la bonne utilisation des mécanismes algorithmiques vu au cours et le temps de résolution sur les instances données.
Aucune librairie ne peut être utilisée en dehors des librairies `sys` et `time`. Les variables globales ne sont pas autorisées.

Fichier de données

Un jeu d'instances de bin packing est disponible sur l'UV. Les nom des instances est écrit sous la forme `binT-O.txt` où `T` et `O` représentent respectivement la taille des boites et le nombre d'objets. Dans chaque fichier, la première ligne indique le poids maximum que peut contenir chaque boite, les lignes suivantes contiennent le poids des différents objets. Le fichier correspondant à l'instance du premier exemple de l'énoncé est :

```
20
1
3
4
```

```
4
5
6
7
8
8
10
```

Pour donner une idées des temps de résolutions, toutes les instances peuvent être résolues en moins de deux minutes.

Consignes de remise

Les "Consignes de réalisation des projets" (cf. http ://www.ulb.ac.be/di/consignes_projets_INFO1.pdf) sont d'application pour ce projet individuel. Tout votre code doit être contenu dans un seul fichier `projet1.py`. Veuillez respecter le nom, la casse et le format des variables, fonctions et du fichier demandé.
Votre fichier sera importé dans un autre fichier lors de la correction, veillez donc à ce que rien ne soit exécuté depuis votre fichier lors d'un `import`.
Nous vous demandons de

- créer localement sur votre machine un répertoire de la forme NOM_Prenom (exemple :DUPONT_Jean) dans lequel vous mettez le fichier à soumettre,
- compresser ce répertoire via un utilitaire d'archivage produisant un .zip (aucun autre format ne sera accepté),
- de soumettre le fichier archive .zip, et uniquement ce fichier, sur l'UV.

Les versions électronique et papier sont à remettre pour le lundi 19 mars 2018 à 12h sur l'UV et au secrétariat étudiant, aucun retard n'est toléré. **Tout manquement aux consignes sera sanctionné directement d'un 0/10.**