Cours de modélisation mathématique

Guilherme Dias da Fonseca

TP 1 – Bin Packing

1 À propos du cours

Pendant 5 semaines, à chaque semaine on va avoir un projet différent. Les projets seront notés et la note finale sera la moyenne arithmétique de ces 5 notes ainsi que d'une soutenance sur un des 5 projets au choix, qui sera réalisée au dernier jour de cours, à la sixième semaine.

Chaque projet consiste d'un problème d'optimisation différent, ainsi que des fichiers d'entrée de tailles variées. Vous devez coder un logiciel qui obtient une sortie la plus optimisée possible pour chaque fichier. Votre note sera basée sur la qualité des solutions obtenues par vos algorithmes.

Vous devez codez vos logiciels sur Linux, soit en Python 3 (recommandé), soit en Java. Il faut que le logiciel exécute sur les machines des sales de TP et utilisent seulement les libraries distribuées avec le langage. J'ai mis un petit tutoriel de Python (en anglais) sur l'ENT, mais vous pouvez facilement trouver d'autres.

Il n'y a pas d'interface graphique pour les TPs. Seulement un logiciel qui prend un fichier d'entrée et produit un fichier de sortie. Pour des raisons pratiques le temps d'exécution de votre logiciel est limité à 1 minute par fichier. Vous pouvez avoir des paramètres spécifiques pour identifier chaque fichier dans vos codes, par exemple utiliser un algo pour des petits fichier et un autre algo différent pour des gros fichiers.

La date limite à rendre chaque TP est dimanche à midi de la semaine suivante au TP (alors, vous avez presque 2 semaines). Je ne veux pas juger chaque excuse pour ne pas rendre le TP dans le délai (pane d'ordinateur, chat mort...), mais je suis sur qu'il y a des excuses très légitimes. Alors, si vous jugez en avoir une, vous avez le droit de rendre le TP dans un délai supplémentaire de 24h, sans m'expliquer la raison. Par contre, je n'accepterai pas de TP après ce délai, sauf avec un justificatif formel au sein de l'université. Les TPs sont à rendre sur l'ENT et je n'accepte pas de TP par mail. Vous pouvez envoyer plusieurs fois le même TP et seulement le dernier sera stocké sur moodle.

Vous avez le droit de discuter et même de regarder le code de vos collègues, mais vous n'avez pas le droit de copier le code. Le plagiat est une fraude grave. "Vous risquez une sanction pouvant aller du zéro jusqu'à une interdiction d'examens ou une exclusion de l'université de quelques mois à plusieurs années" [1]

1. http://www.studyrama.com/vie-etudiante/se-defendre-vos-droits/triche-et-plagiat-a-l-universite/plagier-c-est-frauder-et-risquer-des-sanctions-74063

J'ai écrit un logiciel pour préparer le fichier zip de vos TP, à rendre sur l'ENT. La notation des codes sera faite de façon automatique, sauf cas spéciaux. Pour cette raison, vos logiciels devront suivre les consignes à la lettre.

2 Bin packing

Vous avez appris à résoudre plusieurs problèmes algorithmiques de manière exacte. Par exemple, comment trier une liste d'entiers. Dans ce cas, c'est pas raisonnable d'obtenir une liste plus ou moins trier, avec quelques éléments mal placés. Par contre, dans le monde réel on a souvent besoin résoudre des problèmes pour lesquels on ne connait pas d'algorithme exacte efficace. Dans ces cas, on fait ce qu'on peut, et on trouve une solution si bonne que possible en pratique. Un exemple de ces problèmes est le *bin packing*.

2.1 Présentation abstraite

Entrée : Un entier c et une liste de n entiers $T = t_1, \ldots, t_n \leq c$.

Sortie : Un entier k et une fonction $f : \{1, \ldots, n\} \longrightarrow \{1, \ldots, k\}$ tel que pour $i = 1, \ldots, k$:

$$\sum_{f(j)=i} t_j \leq c.$$

Objectif : Minimiser k.

En d'autres termes, on est donné un entier c et une liste de n entiers $T = t_1, \ldots, t_n \leq c$. L'objectif est diviser les n entiers parmi k ensembles tels que la somme des éléments d'un ensemble ne dépasse pas c et de façon à minimiser k.

2.2 Présentation appliquée

On a n fichiers de tailles $t_1, \ldots, t_n$. On veut garder les fichiers dans plusieurs clés USB identiques de capacité c. On peut pas diviser un fichier entre deux clés, mais heureusement aucun fichier a une taille supérieur à c. Alors, il faut décider comment distribuer les fichiers parmi un nombre k de clés USB. Pour des raisons économiques, on veut utiliser le minimum de clés USB possible.

À la place de clés USBs et fichiers, ça pourrait être le cas de distribuer n produits parmi des camions sans excéder le poids limite des camions. La formulation abstraite du problème ne change pas. On peut penser à plusieurs autres applications, comme découpe de câbles, ordonnancement de taches parmi plusieurs serveurs, etc.

2.3 Exemple

Disons que l'entrée est $c = 10$ et la liste $3, 4, 3, 5, 7, 6$. Une solution possible utilise $k = 4$ boites :

$$[3, 3, 4], [5], [6], [7].$$

Par contre, on peut faire mieux, en utilisant $k = 3$ boites :

$$[4, 6], [3, 7], [3, 5].$$

3 Projet

C'est à vous de réfléchir sur les stratégies possibles pour trouver des bonnes solutions. En générale, la solution trouvée ne sera pas optimale, spécialement pour les gros fichiers. Personne est capable de trouver toujours la meilleure solution pour ce problème. Faite le mieux que vous pouvez !

Quelques idées :

1. Pensez à comment vous pouvez résoudre le problème sans ordi
2. Pensez à une stratégie rapide et bête pour trouver une solution (glouton)
3. Essayez une stratégie un peu plus intelligente et vérifier si ça marche mieux ou pas
4. Votre algo finit très vite ? Alors, vous pouvez ajouter des choix aléatoire [2] dans votre algo, l'exécuter plusieurs fois et renvoyer la meilleure solution trouvée.
5. Essayez de laisser l'ordinateur faire des petits changements sur votre solution pour l'optimiser un peu (local search).

Surtout, soyez créatif et essayez plusieurs possibilités de façon méthodique.

3.1 Fichiers d'entrée

Le fichier d'entrée est textuel et consiste d'un entier par ligne. Le premier entier est le nombre d'items n. Le deuxième est la capacité c. Ensuite, on a les n entiers $t_1, \ldots, t_n$.

Pour l'exemple on a le fichier :

```
6
10
3
4
3
5
7
6
```

Vous allez trouver 4 fichiers d'entrée sur l'ENT, pour $n = 20, 100, 1000, 10000$. Le fichier sera envoyé au logiciel avec redirection comme l'entrée standard.

3.2 Sortie

La première ligne du fichier de sortie est l'entier k. Ensuite on liste les tailles des fichiers dans chaque clé, en séparant les clés différentes par une ligne vide. Pour l'exemple on a :

2. Un nombre pseudo-aléatoire avec un *seed* choisi est la meilleure solution, pour avoir des résultats toujours reproductibles.

```
3
4
6

3
7

3
5
```

3.3 Interface du logiciel

Les fichiers d'entrée et sortie seront fournis avec redirection des entrée et sortie standards. Si vous voulez voir le progrès de votre logiciel, vous pouvez utiliser *stderr* avec les messages que vous voulez.

Python Le logiciel doit être codé sur un seul fichier appelé `tp.py`. Le logiciel sera exécuté sur linux avec `python3 tp.py <entree >sortie`.

Java Le logiciel doit être codé sans package, pour être compilé avec `javac *.java` et ensuite exécuté avec `java Tp <entree >sortie`.

3.4 Notation

La notation sera basée sur le tableau ci-dessous, qui indique le nombre de boîtes k nécessaire pour avoir la note à gauche pour chaque fichier. La note du projet est la moyenne des notes des 4 fichiers.

Note	20.in1	100.in1	1000.in1	10000.in1
10	6	33	140	1100
12		32	130	1019
14	5	31	127	1018
16			126	1015
18			125	1010
20	4	30	124	1000

La notation est automatique (sauf exceptions) et utilise le logiciel `grader1.py` disponible sur l'ENT. Pour exécuter ce logiciel, il faut être dan un dossier avec les fichiers d'entrée ainsi que votre code. Voici un exemple :

```
$ ls
10000.in1  1000.in1  100.in1  20.in1  tp.py
$ python3 ../grader1.py
Found the tp.py file

***Testing file 20.in1
Running python3 tp.py <20.in1 >20.out1
```

```
Your file passed!
You used 4 bins, when the lower bound is 4
Check the file: 20.out1.html
Your grade for this file is 20

***Testing file 100.in1
Running python3 tp.py <100.in1 >100.out1
Your file passed!
You used 30 bins, when the lower bound is 30
Check the file: 100.out1.html
Your grade for this file is 20

***Testing file 1000.in1
Running python3 tp.py <1000.in1 >1000.out1
Your file passed!
You used 125 bins, when the lower bound is 124
Check the file: 1000.out1.html
Your grade for this file is 18

***Testing file 10000.in1
Running python3 tp.py <10000.in1 >10000.out1
Your file passed!
You used 1004 bins, when the lower bound is 1000
Check the file: 10000.out1.html
Your grade for this file is 18

Your average is 19.0
Your actual grade will depend on how your code performs when ran on the grading computer

Enter your email @etu.udamail.fr: John.Smith@etu.udamail.fr
Enter the email of the other student, if any:
The group is ['John.Smith']
Creating submission file tp1_JSmith.zip
        zip warning: name not matched: *.java
  adding: tp.py (deflated 53%)
  adding: report.txt (deflated 19%)

Please check the file and remember to submit it on time!
```

Le logiciel va créer le fichier zip à envoyer sur l'ENT. Il ne faut pas modifier ce fichier, mais vérifiez que votre code est bien dedans.

Le logiciel va aussi créer automatiquement des fichiers html pour visualiser vos solutions, comme la figure 1.

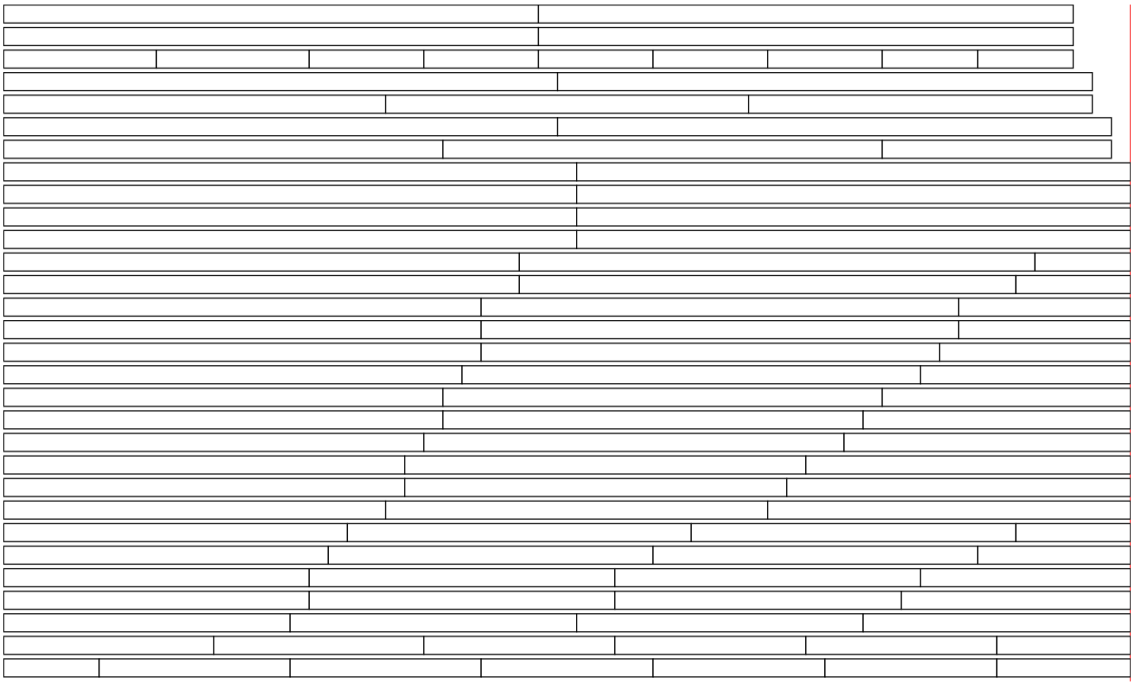

FIGURE 1 – Exemple du fichier `100.in1` avec 30 boites.