INF8775 – Analyse et conception d'algorithmes

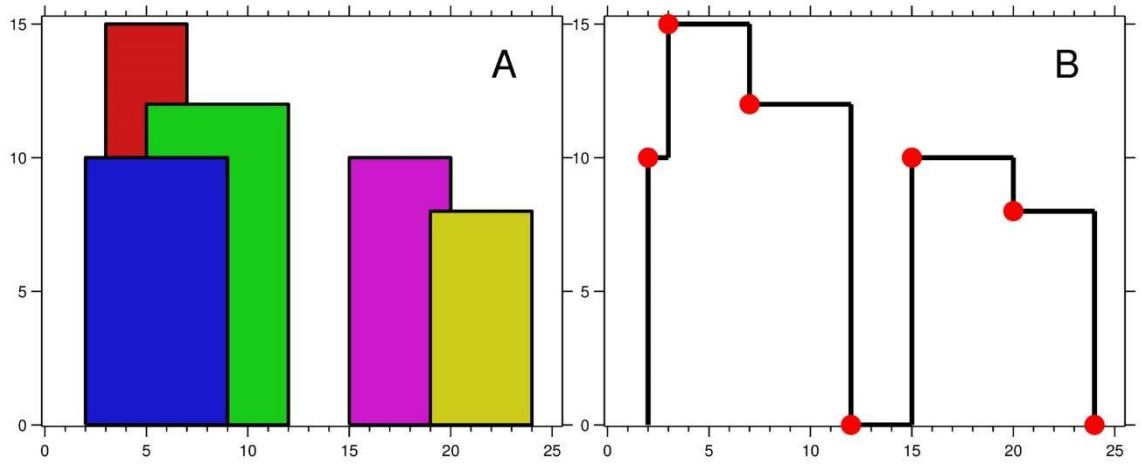
TP1 – Hiver 2022

Mise en situation

Ce travail pratique se répartit sur deux séances de laboratoire et porte sur l'analyse empirique et hybride des algorithmes. Dans les capsules vidéo de la semaine 3, trois approches d'analyse de l'implantation d'un algorithme sont décrites. Vous les mettrez en pratique pour des algorithmes de résolution d'un problème connu.

Description du problème

On vous demande de résoudre le problème de la ligne d'horizon[1] (*The Skyline Problem*) qui consiste à dessiner la silhouette de bâtiments lorsqu'ils sont vus de loin. Ces bâtiments sont juxtaposés l'un à l'autre et il est possible que l'un en cache un autre. Soit l'exemple suivant avec 5 bâtiments :



La figure B représente la silhouette (et donc la solution) tracée par les bâtiments colorés de la figure A.

Concrètement, chaque bâtiment est défini par le triplet (x_1, x_2, h) avec h la hauteur du bâtiment et x_1 et x_2 les abscisses des murs gauche et droit, respectivement. La solution quant-à-elle représente une suite de couples (x, h) représentant les coordonnées des points définissant la silhouette des bâtiments.

Pour l'exemple ci-dessus :

[1] D'après https://leetcode.com/problems/the-skyline-problem/

- Les données du problème sont telles que : (2, 9, 10), (3, 7, 15), (5, 12, 12), (15, 20, 10),
(19, 24, 8).
- La solution est telle que : (2, 10), (3, 15), (7, 12), (12, 0), (15, 10), (20, 8), (24, 0).

Algorithmes à implanter

On vous demande de résoudre ce problème de 3 façons différentes :

1. En utilisant un algorithme force brute simple ;
2. En utilisant un algorithme diviser pour régner ;
3. En utilisant un algorithme diviser pour régner avec seuil de récursivité non élémentaire.

Pour l'algorithme 3, vous devez déterminer un seuil de récursivité expérimentalement. Les exemplaires dont la taille est en deçà de ce seuil ne sont plus résolus récursivement mais plutôt directement avec l'algorithme 1.

Jeu de données

Pour tester les algorithmes, vous devez générer un jeu de données avec au moins 5 exemplaires pour chacune des tailles suivantes : 1000, 5000, 10000, 50000, 100000 et 500000. On vous fournit un script python `inst_gen.py` pour générer vos exemplaires. Son usage est le suivant :

```
$ ./inst_gen.py [-h] -s NB_BATIMENTS [-n NB_EXEMPLAIRES]
```

On suggère d'utiliser le script bash suivant pour automatiser la génération des exemplaires :

```
for n in {"1000","5000","10000","50000","100000","500000"}; do
    ./inst_gen.py -s $n -n 5
done
```

La première ligne de chaque exemplaire représente le nombre de bâtiments N (qui est aussi la taille du problème). Chacune des N lignes subséquentes représente les triplets discutés dans « Description du problème » avec leurs valeurs entières séparées par des espaces. Par ailleurs, on garantit que tous les triplets générés sont triés selon x_1 et que chacun d'eux est tel que $x_2 - x_1 \geq 1$.

Contraintes sur les solutions

Vos algorithmes doivent donner des réponses où les couples (x, h) sont triés de façon croissante selon x (cf. exemple plus haut). Par ailleurs, ils ne doivent pas donner de solutions avec couples redondants, i.e. deux couples qui se suivent ne peuvent pas avoir la même hauteur ni la même abscisse.

Interface d'exécution

Utilisation :

```
$ ./tp.sh -a {brute, recursif, seuil} -e CHEMIN_EXEMPLAIRE [-p] [-t]
```

Arguments optionnels :

> `[-p]` affiche, sur chaque ligne, les couples définissant la silhouette de bâtiments, triés selon l'abscisse et sans texte superflu (les deux valeurs d'un couple sont séparées d'un espace) ;
> `[-t]` affiche le temps d'exécution en millisecondes, sans unité ni texte superflu.

Important : l'option `-e` doit pouvoir accepter des chemins absolus.

Affichage de la solution

Vous devez afficher tous les couples (x, h) les uns à la suite des autres, en mettant un couple par ligne. Sur chaque ligne affichez *x* en premier, et séparez *x* et *h* d'un espace. Si l'on reprend l'exemple de la description du problème, voici la sortie que vous devez obtenir :

```
[aubur@m3409-01 tp1]$ ./tp.sh -a brute -e N5_0  -p
2 10
3 15
7 12
12 0
15 10
20 8
24 0
```

Un script *check_sol.py* vous est fourni avec le sujet. Vous devez vous assurez que vos solutions sont correctement lues par ce script. Dans le cas contraire, cela signifiera que le script que nous utiliserons pour la correction ne sera pas capable de lire vos solutions. Vous risquerez donc de perdre des points de validité de solution.

Informations techniques

> Répondez dans le document DOCX. Veuillez ne pas inclure le texte en italique servant de directive.
>
> La correction se fait sur ce même rapport.
>
> Vous devez faire une remise électronique sur Moodle avant le 23 Février à 23h59 en suivant les instructions suivantes :
>
> - Vos fichiers doivent être remis dans une archive zip à la racine de laquelle on retrouve :

- Le rapport au format DOCX.
- Un script nommé *tp.sh* servant à exécuter les différents algorithmes du TP. L'interface du script est décrite à la fin du rapport.
- Le code source et les exécutables.
- Si le langage que vous utilisez nécessite une phase de compilation, veuillez joindre un Makefile afin que nous puissions le compiler en cas de problème avec vos exécutables. Si nous ne sommes pas en mesure de tester votre code, vous perdrez des points de respect d'interface et de qualité de code !

Vous avez le choix du langage de programmation utilisé mais vous devrez utiliser les mêmes langage, compilateur et ordinateur pour toutes vos implantations. Le code et les exécutables soumis devront être compatibles avec les ordinateurs de la salle L-4714.

Si vous utilisez des extraits de codes (programmes) trouvés sur Internet, vous devez en mentionner la source, sinon vous serez sanctionnés pour plagiat.