INF8775 – Analyse et conception d'algorithmes

TP2 – Automne 2020

Nom, prénom, matricule des membres	Nom, Prénom, matricule Nom, Prénom, matricule
Note finale / 17	0

Informations techniques

- Répondez directement dans ce document ODT avec LibreOffice. Veuillez ne pas inclure le texte en italique servant de directive.
- La correction se fait à même le rapport.
- Vous devez faire une remise électronique avant le 13 novembre 23h59 en suivant les instructions suivantes:
  - Le dossier remis doit se nommer matricule1_matricule2_tp2 et doit être compressé sous format zip.
  - À la racine de ce dernier, on doit retrouver :
    - Ce rapport sous format ODT.
    - 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
- 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 compatible 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.

Mise en situation

Ce travail pratique se répartit sur deux séances de laboratoire et porte sur l'analyse et la conception d'algorithmes développés suivant différents patrons de conception.

Vous participez à une compétition dont le principal objectif est de construire la tour la plus haute avec les blocs de béton étant à votre disposition. Vous devez les empiler afin que votre tour soit la plus haute possible. Vous disposez d'un ensemble de blocs de dimensions variées, qui sont définies comme suit : hauteur h_{bloc}, largeur l_{bloc} et profondeur p_{bloc}. Afin de garantir la stabilité de votre tour, vous devez vous assurer que le bloc que vous ajoutez sur votre tour repose entièrement sur le précédent.

Plus formellement, vous devez respecter la règle suivante:

$$l_{nouveauBloc} < l_{blocReceveur} \text{ et } p_{nouveauBloc} < p_{blocReceveur}$$

qui garantit une inclusion stricte de la base du nouveau bloc dans la surface du bloc receveur.

Votre objectif est donc de maximiser la hauteur H de votre tour. Une solution au problème est un sous-ensemble de blocs ordonné, de la base au sommet, tel que chaque bloc peut supporter le suivant.

Implantation

Trois algorithmes seront implantés, mettant en pratique les patrons de conception glouton, programmation dynamique et recherche avec tabou.

Algorithme glouton

Vous devez concevoir un algorithme glouton de votre cru pour résoudre ce problème. Votre critère de choix devra être déterministe.

Algorithme de programmation dynamique

Vous devez également résoudre ce problème à l'aide de la programmation dynamique. Tout d'abord, pour que l'approche fonctionne, vous devez trier les blocs disponibles en ordre décroissant de surface. Cela revient à calculer l'aire pour chacun des blocs ($\mathrm{SurfaceBloc}_i = l_i \times p_i$).

Soit $\mathrm{H(j)}$ la hauteur de votre tour avec le bloc j au sommet; on la définit récursivement par

$H(j) = max_{1 \leq i < j \,:\, l_i > l_j,\ p_i > p_j}\{H(i)\} + h_j.$

Pour un exemplaire à n blocs la solution se trouve à $\max_{1 \leq j \leq 3n} H(j)$.

Recherche avec tabou

Enfin vous devez implanter l'approche de recherche avec tabou avec le voisinage suivant :

> Choisir un bloc parmi ceux ne faisant pas partie de la tour ni de la liste taboue puis le placer sur le bloc le plus haut dans la tour courante qui puisse le recevoir. Ce mouvement permet en particulier de modifier la base de la tour. Cette insertion peut engendrer le retrait de certains blocs directement au-dessus pour conserver l'équilibre de la tour: chacun de ces blocs sera alors tabou pour un nombre d'itérations choisi uniformément au hasard dans l'intervalle $[7,10]$. Le voisin choisi est celui qui maximise la hauteur de la tour.

Démarrez avec une tour vide et arrêtez après 100 itérations sans amélioration de la meilleure solution trouvée.

Jeu de données

Pour tester les algorithmes, vous devez générer un jeu de données avec 10 exemplaires pour les tailles: 100, 500, 1000, 5000, 10000, 50000 et 100000. Vous pouvez utiliser le script suivant:

```
#!/bin/bash
for n in {100,500,1000,5000,10000,50000,100000}; do
    for i in {1..10}; do
        # Génération d'une permutation de 1 à 3n
        shuf -i 1-$((3*$n)) |
        # Regroupement par bloc de 3
        awk 'BEGIN {i=0} {printf $1; if (++i%3==0) printf "\n";  else printf " "}' |
       # On génère trois orientations différentes
        awk '{
               printf $1; if ($2 < $3) print " "$2" "$3; else print " "$3" "$2;
               printf $2; if ($1 < $3) print " "$1" "$3; else print " "$3" "$1;
               printf $3; if ($1 < $2) print " "$1" "$2; else print " "$2" "$1;
       }' > b${n}_${i}.txt
    done
done
```

Chaque ligne représente les trois dimensions d'un bloc (hauteur, largeur, profondeur). Chaque taille *n* contient *3n* blocs. Si vous regardez le script, on génère trois blocs à partir de chaque groupe de trois entiers généré par *shuf* + *awk*.

Présentation des résultats

0	/ 5 pt

Tableau des résulats

Exécutez chacun des trois algorithmes en notant leur temps d'exécution et la hauteur maximale de votre tour, mais ne rapportez dans un tableau que la moyenne de chacune des séries de dix exemplaires. Pour le vorace probabiliste, exécutez chaque exemplaire dix fois et utilisez la moyenne de ces exécutions.

Graphs pour analyse hybride

Voir questions plus bas.

Analyse et discussion

0	/ 8 pt

Tentez une analyse asymptotique du temps de calcul pour chaque algorithme.

Si vous préférez écrire vos équations en Latex, vous pouvez ajouter un pdf à la remise avec la réponse à cette question et le mentionner ici.

Servez-vous de vos temps d'exécution pour confirmer et/ou préciser l'analyse asymptotique théorique de vos algorithmes avec la méthode hybride de votre choix.

La méthode peut varier d'un algorithme à l'autre. Justifiez les choix ici et mettez les graphiques dans la section précédente. Attention, vous devrez vous poser des questions pour l'algorithme tabou.

Discutez des trois algorithmes en fonction de la qualité respective des solutions obtenues, de la consommation de ressources (temps de calcul, espace mémoire) et de la difficulté d'implantation.

Indiquez sous quelles conditions vous utiliseriez chaque algorithmes.

Autres critères de correction

Respect de l’interface tp.sh

0	/ 1 pt

Utilisation

tp.sh -a [vorace | progdyn | tabou] -e [path_vers_exemplaire]

Arguments optionnels

-p affiche les blocs utilisés dans la construction de la tour chacun sur une ligne (hauteur, largeur, profondeur) en commençant par le bas.

-t affiche le temps d’exécution en ms, sans unité ni texte superflu

Important: l’option -e doit accepter des fichiers avec des paths absolus.

Qualité du code

0	/ 1 pt

Présentation générale

0	/ 2 pt

- Concision
- Qualité du français

Pénalité retard

0

- -1 pt / journée de retard, arrondi vers le haut. Les TPs ne sont plus acceptés après 3 jours.