**INF8775 – Analyse et conception d’algorithmes**

Rapport TP2 – Automne 2023

|  |  |
| --- | --- |
| **Nom, prénom, matricule des membres** | NOM, Prénom, 1234567  NOM, Prénom, 1234567 |
| **Note finale / 30** | 0 |

# Informations techniques

* Répondez directement dans ce 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 14 novembre à 23h55 pour le groupe 1, 7 novembre à 23h55 pour le groupe 2 et 9 novembre à 23h55 pour le groupe 3.
* Vos fichiers doivent être remis dans une archive zip à la racine de laquelle on retrouve:
  + Ce rapport sous 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.
  + Vous avez le choix du langage de programmation utilisé mais vous devrez utiliser les mêmes langage, compilateur et ordinateur pour toutes vos implantations. Notez que le code et les exécutables soumis seront testés sur les ordinateurs de la salle L-4714 et doivent être compatibles avec cet environnement. En d’autres mots, tout doit fonctionner correctement lorsque le correcteur exécute votre script tp.sh sur un des ordinateurs de la salle.
* Si vous utilisez des extraits de codes (programmes) trouvés sur Internet, vous devez en mentionner la source, sinon vous serez sanctionnés pour plagiat.
* On vous encourage à lire le guide intitulé « guide bash » sur Moodle pour faire vos graphiques. C’est un guide qui a été conçu pour un ancien TP, mais il contient beaucoup d’informations utiles.

# 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.

# Votre principal objectif est de construire la tour la plus haute avec les blocs de béton 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 hbloc , largeur lbloc et profondeur pbloc .

# 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:

# 𝑙𝑛𝑜𝑢𝑣𝑒𝑎𝑢𝐵𝑙𝑜𝑐 < 𝑙𝑏𝑙𝑜𝑐𝑅𝑒𝑐𝑒𝑣𝑒𝑢𝑟 𝑒𝑡 𝑝𝑛𝑜𝑢𝑣𝑒𝑎𝑢𝐵𝑙𝑜𝑐 < 𝑝𝑏𝑙𝑜𝑐𝑅𝑒𝑐𝑒𝑣𝑒𝑢𝑟

# 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 probabiliste

# Algorithme glouton

# Vous devez concevoir un algorithme glouton de votre cru pour résoudre ce problème. Votre critère de choix devra être déterministe et la complexité de votre algorithme devra être inférieure ou égale à O(N²).

# 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 (𝑆𝑢𝑟𝑓𝑎𝑐𝑒𝐵𝑙𝑜𝑐 )𝑖 = 𝑙𝑖 × 𝑝𝑖

# Soit 𝐻(𝑗) la hauteur de votre tour avec le bloc 𝑗 au sommet; on la définit récursivement par 𝐻(𝑗) = 𝑚𝑎𝑥1≤𝑖≤𝑗 : tq 𝑙𝑖 > 𝑙𝑗, 𝑝𝑖 > 𝑝𝑗 {𝐻(𝑖)} + ℎ𝑗

# Pour un exemplaire à 𝑛 blocs la solution se trouve à 𝑚𝑎𝑥1≤j≤n 𝐻(j)

# Algorithme probabiliste

# Enfin vous devez implanter l'approche probabiliste. Pour cela vous devrez utiliser un algorithme de Fitness Proportionate Selection. Chaque bloc i disponible peut être sélectionné avec une probabilité P(bi) = exp(f(bi)) / Σj ∈possibles exp(f(bj)) (softmax).

# Avec f(bi) la fonction critère permettant de classer les blocs par “utilité” dans notre problème. Dans ce TP, on prendra f(bi) = Hauteur(bi) + Aire\_base(bi).

Lorsqu’un bloc est choisi, que le bloc passe ou pas on met sa probabilité à 0. En effet :

* S’il passe il sera utilisé donc plus disponible
* S’il ne passe pas, il ne passera pas plus haut dans la tour donc on peut l’éliminer aussi

Exécutez cet algorithme plusieurs fois et retenez la meilleure solution générée.

# Présentation des résultats

|  |  |
| --- | --- |
|  | / 1,5pts |

## Tableau des résultats

*Exécutez chacun des trois algorithmes et rapportez dans un tableau leur temps d'exécution et la hauteur maximale de votre tour. Pensez à indiquer l'unité de temps utilisée.*

## Précision en fonction du nombre de relances

*Tracez un graphe démontrant l’évolution du rapport Hauteuratteinte / Hauteuroptimale*

*en fonction du nombre de relances de l’algorithme probabiliste pour une taille d’exemplaire donnée.*

**Jeu de données**

Pour tester les algorithmes, vous devez générer un jeu de données avec plusieurs exemplaires pour chaque taille que vous aurez choisie. Vous pouvez utiliser le script gen.py donné avec cet énoncé et modifier la liste des tailles selon votre choix. Comme pour le TP1, prenez des tailles donnant des temps compris entre 1ms et 10s.

Le fichier généré correspondant à la taille n possède n lignes, chacune représentant les trois dimensions d’un bloc (respectivement hauteur, largeur, profondeur)

# Analyse et discussion

### Faites une analyse asymptotique théorique du temps de calcul pour chaque algorithme.

|  |  |
| --- | --- |
|  | / 7,5 pt |

*Justifiez votre analyse. Veillez à indiquer la complexité de chaque étape clé, même de celles qui peuvent devenir négligeables face à d’autres étapes plus complexes. Nous devons voir que vous les avez bien prises en compte.*

### 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.

|  |  |
| --- | --- |
|  | / 7,5 pt |

*La méthode peut varier d'un algorithme à l'autre. Justifiez les choix ici et montrez vos graphiques .*

### 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.

### Selon la courbe d’erreur, combien de fois faudrait-il relancer l’algorithme probabiliste pour obtenir au moins 99 % de la hauteur optimale en moyenne?

### Indiquez sous quelles conditions vous utiliseriez chaque algorithme.

|  |  |
| --- | --- |
|  | / 6,5 pt |

## 

# Autres critères de correction

## Respect de l’interface tp.sh

|  |  |
| --- | --- |
|  | / 1 pt |

## Utilisation

## tp.sh -a [glouton | progdyn | proba] -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.

-h affiche la hauteur totale atteinte par la tour.

## -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

|  |  |
| --- | --- |
|  | / 5 pt |

* + - * 1. Validité des solutions
        2. Qualité de l'implémentation

Présence de commentaires

## Présentation générale

|  |  |
| --- | --- |
|  | / 1 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.