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

Rapport TP2 – Hiver 2022

|  |  |
| --- | --- |
| **Nom, prénom, matricule des membres** | Pucci-Barbeau, Vincenzo, 1948994  Bossert, Matthieu, 2161168 |
| **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 28 Mars à 23h59 en suivant les instructions suivantes :
  + Vos fichiers doivent être remis dans une archive zip à la racine de laquelle on retrouve :
    - Ce 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.

# Présentation des résultats

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

## Tableau des résultats

*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.*

*Pensez à indiquer l'unité de temps utilisée.*

## Temps d’exécution moyen en secondes :

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithme  Taille exemplaire | Glouton | Prog. Dyn. | Tabou |
| 100 | 0.000103 | 0.0115 | 0.299 |
| 500 | 0.000474 | 0.282 | 4.078 |
| 1’000 | 0.00116 | 1.15 | 11.6 |
| 5’000 | 0.00657 | 32.2 | 135 |
| 10’000 | 0.0140 | 143 | 397 |

## Hauteur moyenne de la tour :

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithme  Taille exemplaire | Glouton | Prog. Dyn. | Tabou |
| 100 | 5513 | 7306 | 6610 |
| 500 | 63807 | 86747 | 75445 |
| 1’000 | 177609 | 252132 | 212600 |
| 5’000 | 2046460 | 2960840 | 2457718 |
| 10’000 | 5704276 | 8399209 | 6832425 |

# Analyse et discussion

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

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

*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. 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.*

## Glouton :

* Trier par aire :
* On parcoure la liste de n blocs :
  + Opération baromètre : vérifier si le bloc « fit » sur celui du dessous    
    => pour chaque bloc

La boucle est en

On voit donc que l’algorithme glouton est en

## Programmation Dynamique :

* Trier par aire :
* On parcoure la liste de n blocs :
  + Parcoure la liste de bloc de 1 à i :
    - Opération baromètre : vérifier si le bloc « fit » sur celui du dessous    
      => à chaque passage de boucle
    - Donc pour la boucle
  + Vérifier si la liste est vide et ajouter un bloc à la liste si c’est le cas :
  + Sinon :
    - Trouver le maximum des hauteurs :
    - Ajouter la hauteur maximum à la liste des hauteurs maximums :
* Retracer la liste des résultats en sens inverse : où m est la hauteur de la tour

On a donc

Comme m < n et , n donc que l’algorithme glouton est en :

## Tabou :

Soit :

* n : la taille de l’échantillon initiale
* k : le nombre de voisin à la solution initiale
* m : la taille de la solution initiale
* t : la taille de la liste tabou

*Coût :*

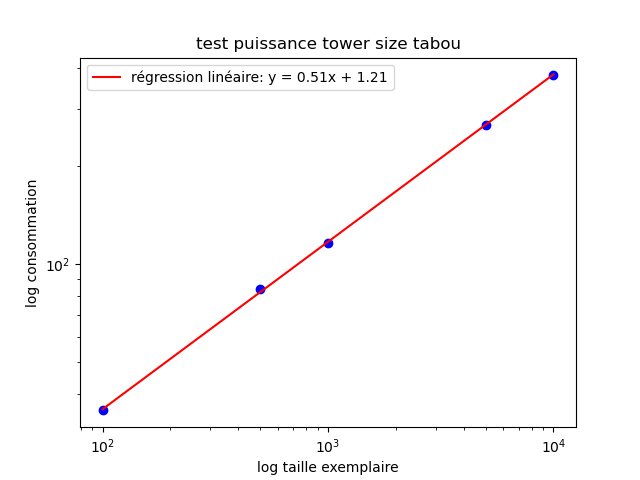
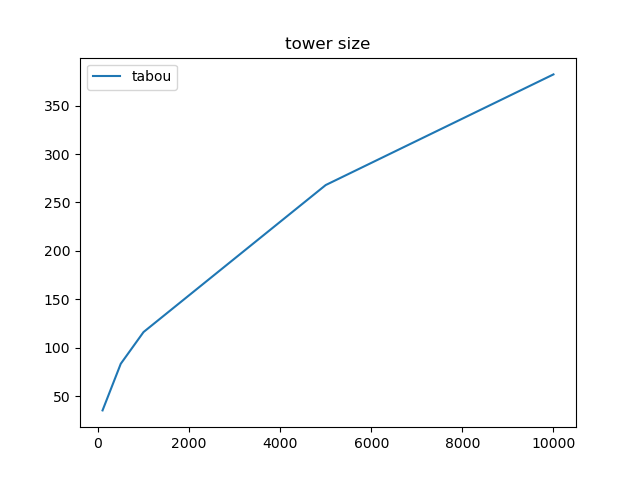
* Exécuter l’algo glouton :
* Exécuter 100 fois :
  + Trouver les voisins : cf. analyse ci-dessous
  + Trouver le voisin avec la plus grande hauteur :
  + Remplacer la solution courante par le meilleur voisin :
  + Retirer les éléments tabous de la liste de blocs : car il y a au plus m éléments dans la liste tabou. On peut facilement supposer cependant que le nombre de blocs tabou sera proportionnel à la taille de la tour donc
  + Actualiser la file d’attente tabou (ajouter les nouveaux éléments et réintégrer les éléments arrivés au début de la file :

Trouver les voisins :

* Différence entre la solution initiale et la liste de blocs :
* Exécuter (n-m) fois :
  + Parcourir une partie de la solution initiale depuis le haut : car c’est proportionnel à la taille de la tour
  + Insérer le bloc :
  + Checker si les blocs sont toujours légaux : car on doit checker tous les blocs au-dessus du bloc inséré

On a donc : T(n) = alpha m + (n-m) \*(beta m + c + gamma\* m) est en O((n-m) \* m) = O(nm)

On pourrait assumer que la taille de la solution (nombre de blocs) proportionnelle à n. Cependant, en pratique, nous trouvons que la taille de la solution évolue plutôt en comme le montre les graph si dessous :



On a donc que l’algorithme de recherche des voisins est en

L’algorithme Tabou au complet est en :

Le nombre de voisin k est au plus n-m, donc :

Au final, l’algo Tabou est donc en :

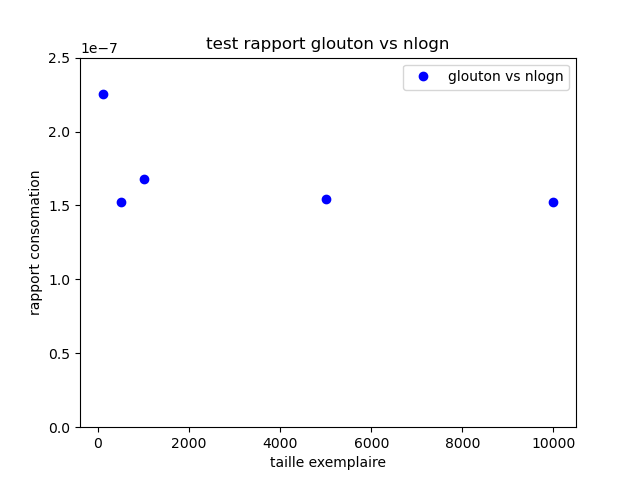
### Servez-vous de vos temps d'exécution pour confirmer et/ou préciser l'analyse asymptotique théorique de vos algorithmes avec la méthode hybride de votre choix.

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

*La méthode peut varier d'un algorithme à l'autre. Justifiez les choix ici et montrez vos graphiques. Attention, vous devrez vous poser des questions pour l’algorithme tabou.*

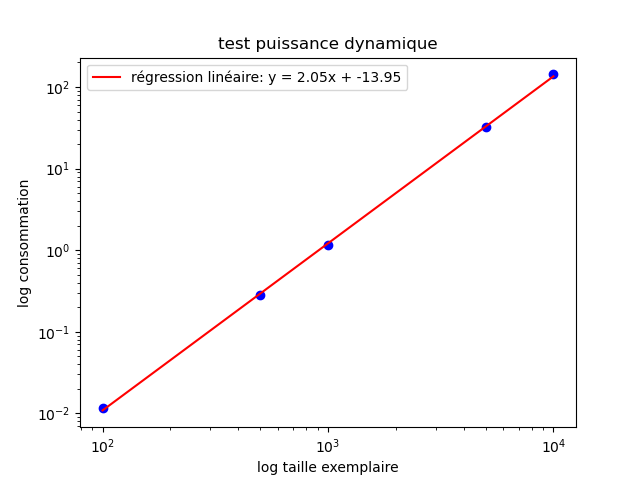
## Glouton :

On utilise le test du rapport afin de déterminer si l’algorithme est bien en . On plot donc . Le rapport converge effectivement vers une valeur de constante de 1.5e-7



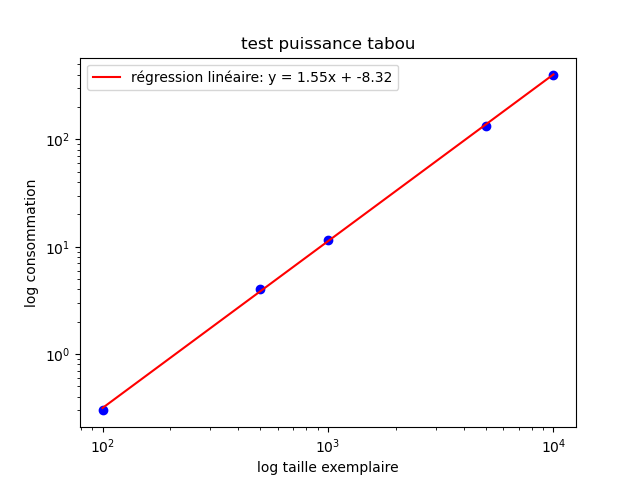
## Programmation Dynamique :

On utilise le test de puissance afin de vérifier le degré de l’algorithme polynomial. On plot donc . On vérifie que l’algorithme est bien en . La constante multiplicative est . On a donc approximativement :



## Tabou :

On utilise le test de puissance afin de vérifier le degré de l’algorithme polynomial. On plot donc . On vérifie que l’algorithme est bien en . La constante multiplicative est . On a donc approximativement :



### 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 algorithme.

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

## 

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithme : | Avantage : | Inconvénients : | Cas d’utilisation : |
| Glouton | Rapidité et complexité temporel moindre ())  Très facile à implémenter  Consommation mémoire faible ()) | Solution approximative  Aucune garantie sur la qualité de la solution, celle-ci peut être arbitrairement mauvaise (heuristique) | Première approximation rapide de la solution  Problème de très grande taille lorsque l’on a pas assez de temps pour exécuter un des deux autres algo |
| Prog dyn. | Solution exacte  Relativement rapide pour de petits exemplaires grâce à une constante multiplicative faible  Consommation mémoire faible ())  Modérément difficile à implémenter | Complexité temporel importante ()), ce qui pénalise les très grands exemplaires | Si on souhaite obtenir la solution exacte au problème  Si le problème est de taille raisonnable (<10^5) |
| Tabou | Complexité temporelle intermédiaire )  Solution meilleure que glouton  Possibilité de trouver un compromis temps de calcul/qualité de la solution en influant sur le nombre maximal d’itérations | Solution meilleure que glouton mais toujours approximative  Pas de garantie de qualité, le maximum local peut être arbitrairement mauvais  Difficile à implémenter  Complexité mémoire la plus importante  Constante multiplicative bien plus grande que prog. dyn. Ce qui fait qu’en pratique, l’algorithme tabou est moins rapide que prog. dyn pour des tailles d’exemplaires moyens, malgré une complexité moins importante. | Si on souhaite avoir une solution de qualité intermédiaire pour de très grands exemplaires (dans tel cas est significativement moins important que |

# Autres critères de correction

## Respect de l’interface tp.sh

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

## 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 |

* -15% de la note / journée de retard, arrondi vers le haut. Les TPs ne sont plus acceptés après 3 jours.