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

TP3 – Automne 2021

| **Nom, prénom, matricule des membres** | FALICOFF, Maximiliano, 2013658  Houtart, Thomas, 1895556 |
| --- | --- |
| **Note finale / 15** |  |

**Q1 – Description de votre algorithme**

*Décrivez en quelques phrases votre algorithme.*

| **0** | / 1 pt |
| --- | --- |

Notre algorithme organise une ligne d'étudiants selon leur taille et leurs conflits. Tout d'abord, on trie les étudiants par taille décroissante. Ensuite, pour chaque étudiant, on tente de le placer dans la ligne sans créer de conflit. Nous avons une classe étudiant qui pour chaque étudiant possède un attribut fights\_with qui a une liste avec les index des étudiants qui il est en conflit avec. Si un placement direct crée un conflit, il trouve une position alternative basée sur la taille en le plaçant à une position où il ne bloque personne. Finalement, il retourne l'ordre des étudiants dans la ligne en respectant ces critères.

**Q2 – Présentation de votre algorithme**

*Sous forme de pseudo-code et incluant une analyse de complexité théorique des principales fonctions. 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. Pas besoin de faire une analyse empirique de la complexité.*

| **0** | / 4 pt |
| --- | --- |

voir fichier a la fin

**Q3 – Justification de l’originalité de votre algorithme**

*La conception de votre algorithme sera jugée avec les critères suivants :*

* *Lien avec le contenu du cours*
* *Originalité*
* *Initiative*

| **0** | / 3 pt |
| --- | --- |

La nature du problème étant un problème d’optimisation, nous avons commencé avec une approche gloutonne naive, donc on place les étudiants selon une fonction fitness naïve telle que la taille ou le nombre de conflits. Cela a donné des résultats proches du baseline minimum.

Le problème avec ces approches était que nous nous concentrions juste sur l’une des deux contraintes, soit la taille ou les conflits. Or, cela posait un problème car si on valorise les conflits, on peut se retrouver avec une personne qui se situe au début de la file qui bloque tous les étudiants, cela n’est pas tant un problème lorsque le nombre d'étudiants est petit mais avec beaucoup d'étudiants cela entraîne a un loss beaucoup plus grand.

L'originalité vient de comment on a pu concevoir un moyen de respecter au maximum les deux contraintes en même temps, donc d'éviter cette situation où un grand étudiant se retrouve au début de la file.

Le tri décroissant initial permet de commencer le tri avec la valeur minimale possible de point généré par la taille, sachant qu’aucun étudiant se cache. Choisir par la suite de placer les étudiants en suivant cet ordre permet d’enlever un grand nombre de conflits tout en limitant la perte de point par la taille. Bien que cette solution offre un score correct, soit meilleur que le bon baseline défini du laboratoire, il reste que cet algorithme n’est pas parfait. La nature de notre approche fait en sorte que les dernières valeurs trié auront une plus forte probabilité d’être en conflit, vu qu’ils auront moins d’option d’être placé et pose le risque principale au niveau de notre score. Cependant, le tri décroissant diminue permet de réduire l’impact de ce risque sa réduction du score qu’il offre, tel que prouvé par nos résultats obtenus qui sont meilleurs que le baseline.

**Q4 – Votre algorithme est-il assuré de trouver la solution optimale ?**

*Répondez simplement “oui” ou “non”. Si oui, justifiez.*

| **0** | / 1 pt |
| --- | --- |

Non