TP 2 INFO 503

Simon Pichenot, Hugo Rey

Introduction :

Dans la première partie de ce TP nous avons dû implémenter un tri à bulle ascendant et la version améliorée, le tri de Dobosiewicz. Pour comparer leurs performances.
Dans la seconde partie, nous avons développé un algorithme d'insertion dans un double tas.

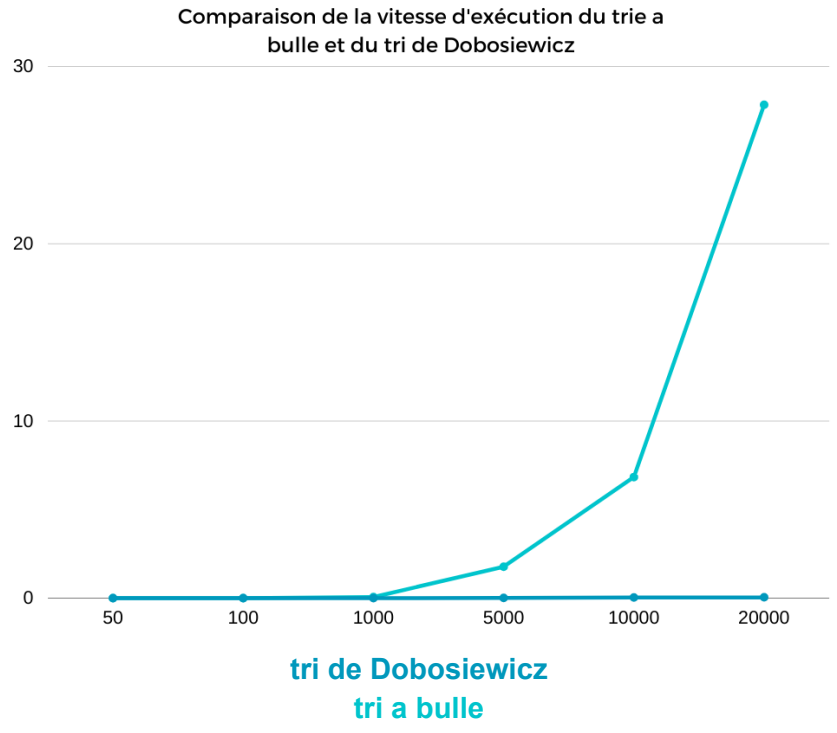
Partie 1 :

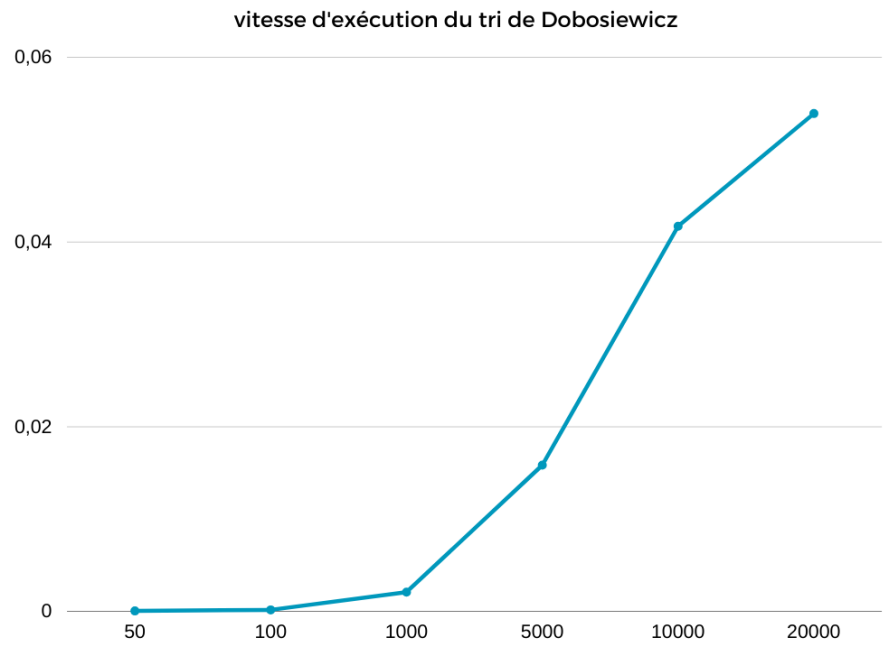
Question 2

Le tri de Dobosiewicz est une amélioration du tri par bulle. Dans le tri par bulle, on compare chaque élément à son voisin immédia. Ici on compare avec des voisins plus éloignés et à chaque nouvelle itération on réduit l'intervalle pour arriver à 1 au final.
L'avantage de cette méthode est que les petits éléménts qui mettaient du temps à "descendre" dans le tri par bulle sont traités plus rapidement.

Question 3

sur les graphiques ci-dessous, en y la vitesse d'exécution (en seconde) et en x le nombre d'éléments à trier.





Conclusion partie 1

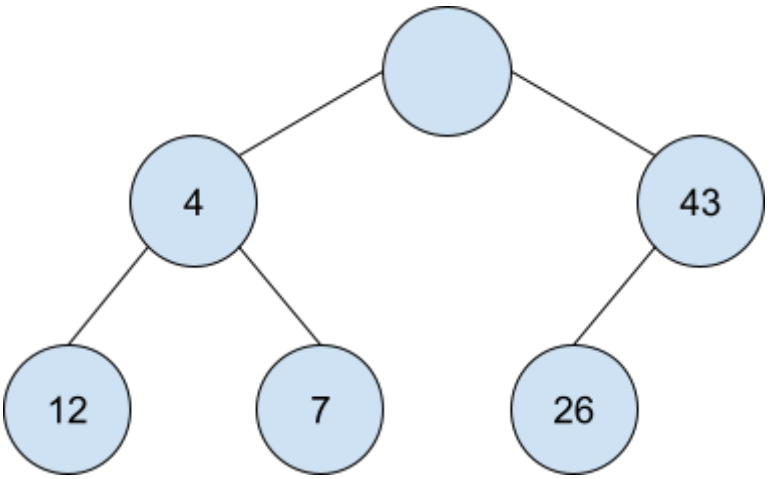
Comme nous pouvons le constater, le trie de Dobosiewicz est bien plus performant. Le temps en secondes n'est pas vraiment intéressant (car il dépent de la machine) mais plutôt l'écart entre les deux algorithmes. Le tri de Dobosiewicz est 450 fois plus rapide pour 20000 éléments triés. et plus le nombre d'éléments à trier augmente, plus l'écart se creuse entre le tri de Dobosiewicz et le tri à bulle

Partie 2 : Les doubles tas

Question 1

On a choisi le langage python pour l'algorithme d'insertion. Ce langage orientée objet nous a permis d'implémenter la structure du double tas comme-ceci :

1. Une classe avec deux attributs, tas de droite et tas de gauche
2. Les deux tas sont des listes pour pouvoir transférer facilement un élément d'un tas à un autre.



La racine vide est la classe python et les listes un des sous arbres.

Question 2

Si la hauteur de l’arbre gauche est égale ou inférieure de 1 à celle de l’arbre droit, on insère dans l’arbre gauche. Sinon dans l’arbre droit.

On se déplace dans l’arbre ou on vient d'insérer (pour l'exemple nous venons d'insérer à gauche). Si le nœud courant est plus petit que son père on les échanges. Sinon si il est plus petit que le nœud a la même position dans l'arbre droit on les échanges (on prend son père si il n’existe pas). On fait la même chose avec l’arbre droit en inversant tous les inférieurs par des supérieurs.

Conclusion partie 2

Cet algorithme d’insertion dans les doubles tas permet de faire remonter les plus petits éléments en haut du tas de droite et les plus grands dans le tas de gauche.