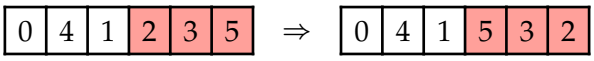
Sujet de TP n°1 du module « Méthodes de résolution des problèmes »

Année universitaires : 2023/2024

1. Introduction

Il existe plusieurs algorithmes de tri, allant des plus simples et naïfs aux plus complexes et efficaces. La plupart de ces algorithmes (par bulles, par insertion, dichotomique, fusion, par tas) reposent sur l'opération `exchange(arr,i,j)` signifiant intervertir les éléments des cases `i` et `j` dans le tableau `arr`.

Dans un système informatique, nous supposons que la seule possibilité pour modifier les éléments d'un tableau est l'opération `reverse(arr,i)` signifiant inverser l'ordre des éléments du tableau `arr` à partir de la position `i`. La figure suivante donne un exemple d'une inversion à partir de la position 3 (en considérant une indexation commençant de 0).



On voudrait implémenter le tri d'un tableau quelconque à l'aide des inversions uniquement. On souhaite trouver le nombre minimal d'inversions pour trier un tableau. On désire également minimiser le temps nécessaire à la résolution du problème.

On modélise le problème selon le formalisme expliqué au cours :

- Les états du problème : un tableau (une liste en Python).
- L'état initial : le tableau initial.
- L'état objectif : le tableau trié.
- Les actions : une seule action `reverse(arr,i)` avec `i` allant de 0 jusqu'à la taille du tableau - 2.

Pour résoudre le problème, il est demandé d'implémenter les méthodes de recherche suivantes :

- Recherche en profondeur.
- Recherche en largeur.
- Recherche aléatoire.
- Recherche A avec $f = g + h$ avec :
  - $g = 0$, h : pour chaque élément, calculer la somme du nombre d'éléments supérieurs à gauche et le nombre d'éléments inférieurs à droite.
  - $g = profondeur$ et h identique au cas précédent.
- Proposez votre propre heuristique.

2. Travail demandé

- Télécharger et comprendre le code Python fourni.
- Le travail présenté doit **impérativement** étendre le code fourni (tout autre code ne sera pas accepté).
- Une fois la solution trouvée, vous devrez afficher les étapes de tri.
- Tester les codes écrits pour différentes tailles du tableau (5,…15, ou plus). Comparez les performances des différentes méthodes.
- Le travail se fait par binôme.
- Le TP doit être hébergé sur Github.
- Afin d'améliorer les performances de recherche, utilisez l'interpréteur `pypy3` au lieu de `python3`.
- Optionnel : les binômes sont invités à développer le TP en utilisant l'approche du *Pair programming*. Dans ce cas, il devrait y avoir une collaboration via la plateforme Github.