Software Engineering Project

Compression de données pour accélérer la transmission

Daniel Carriba Nosrati

UE Software Engineering
Semestre 1 Master 1 Informatique
Université Côte d'Azur

2025

Sommaire

1 Introduction

Ce projet, réalisé pour l'UE "Software Engineering" du Semestre 1 du Master 1 Informatique de l'Université Côte d'Azur, est un projet sur le thème de compression de données pour accélérer la transmission.

La transmission de tableaux d'entiers est un problème majeur de l'internet. Ce projet propose une réponse à ce problème en implémentant une méthode de compression de tableaux d'entiers positifs, basés sur le nombre de bits utilisés. Cette méthode de compression est appelée "Bit Packing". Plusieurs versions de cette méthode ont été implémentés par ce projet. L'utilisateur peut ainsi compresser un tableau d'entiers positifs, ainsi que le décompresser. L'accès direct aux éléments n'est pas perdu lors de la compression, l'utilisateur peut toujours avoir un accès immédiat à un i-ème élément du tableau.

Ce projet à été réalisé en Java. Les différentes versions la méthode de compression "Bit Packing" ainsi que leurs implémentations et fonctionnements seront présentés dans ce rapport. Ce projet propose également des benchmarks, sous forme de mesures de performances et de temps, pour chaque versions de la méthode de compression. L'implémentation ainsi que la pertinence de ces benchmarks seront également expliquer dans la suite de ce rapport.

2 Bit Packing

La méthode de compression "Bit Packing" à été implémenté en trois versions différentes dans ce projet. Chaque version sera présenté en détails dans sa section dédiée.

Pour l'implémentation j'ai choisi donc de créer une classe par versions. Ces classes héritent de la classe abstraite *BitPacking*, qui elle contient les champs et méthodes et communs à toutes les versions différentes de "Bit Packing". Chaque classe qui hérite donc de *BitPacking* implémente sa propre méthode pour compresser, décompresser ainsi que pour accéder au i-ème élément du tableau compressé.

Voici un schéma UML montrant une vue d'ensemble de la structure de ces classes :

(TODO: mettre schéma UML)

2.1 Bit Packing with overlap

Cette version de "Bit Packing" est une version nommé "with overlap". Cela signifie que une valeur, du tableau à compresser, peut être compressé sur deux entiers consécutifs au sein du tableau compressé.

2.1.1 Implémentation

Cette version de "Bit Packing" a été implémenté par la classe *BitPackingWithOverlap* qui hérite de la classe abstraite *BitPacking*.

BitPackingWithOverlap implémente les méthodes abstraites de *BitPacking*, soit :

```
@Override
public void compress(int[] array) {
        ...
}
@Override
public void decompress(int[] array) {
        ...
}
@Override
public int get(int i) {
        ...
}
```

Une classe *BitPackingWithOverlapTest* a également été implémenté contenant des tests unitaires pour tester le bon fonctionnement des méthodes montrés ci-dessus.

2.1.2 Fonctionnement

2.2 Bit Packing without overlap

Cette version de "Bit Packing" est une version nommé "without overlap". Cela signifie que une valeur, du tableau à compresser, sera toujours compressé sur un seul, et non deux, entier du tableau compressé.

2.2.1 Implémentation

2.2.2 Fonctionnement

2.3 Bit Packing with overflow areas

2.3.1 Implémentation

2.3.2 Fonctionnement

3 Benchmarks