Software Engineering Project 2025

Data compressing for speed up transmission

The transmission of integer arrays is one of the central problems of the internet.
In this project, you are asked to study different transmission modes based on integer compression. The idea is to compress an integer array in order to have fewer integers to transmit, then decompress the integers after transmission.

We want to use a compression method that does not lose direct access to the elements. In other words, even after compression, we must be able to easily find the i-th integer in the original array. You are therefore asked to implement a compression method based on the number of bits used, called Bit Packing.
Thus, if each element can be represented using k bits, then we can create a global representation that will use n*k bits instead of 32n bits if we use conventional integers.

You must implement the BitPacking class that performs this compression. You must implement two different versions:
- one can write compressed integers on two consecutive integers
- the other must not write compressed integers on two consecutive integers

For example, if we find that 12 bits are needed to represent 6 elements, then the first representation will write:
- the first integer compressed onto the first 12 bits of the first integer in the output,
- the second integer compressed onto the next 12 bits of the first integer in the output,
- the third integer compressed on bits 25 to 32 on the first integer in the output and on the first 4 bits of the second integer in the output
- the fourth integer compressed on bits 5 to 16 on the second integer in the output
- the fifth integer compressed on bits 17 to 28 on the second integer in the output
- the sixth integer compressed over bits 29 to 32 of the second integer output and over the first 8 bits of the third integer output

the second representation will write
- the first integer compressed over the first 12 bits of the first integer output,
- the second integer compressed over the next 12 bits of the first integer output,
- the third compressed integer on the first 12 bits of the second output integer,
- the fourth compressed integer on the next 12 bits of the second output integer,
- the fifth compressed integer on the first 12 bits of the third output integer,
- the sixth compressed integer on the next 12 bits of the third output integer,

For each type of compression, you must implement the following functions:
- compress(array), which compresses an array
- decompress(array), which decompresses and places the result in the array passed as a parameter

In addition, you must implement the function
- int get(int i) which returns the value of the ith integer in the compressed array.

You must implement measures to time the execution of each function. Be sure to implement a protocol that you explain carefully, which measures as accurately as possible the time taken by each function. Based on these times, you must calculate the transmission time for a latency t at which compression becomes worthwhile.

We now want to perform compression with overflow areas.

Indeed, if a single number in the initial array requires a large number of bits k and the other numbers require
k' bits with k' < k, it is a waste to represent all numbers with k bits. In this case, we can assign a special value to a compressed integer that indicates that the true value is located elsewhere at a certain position in my table, called the overflow area.

For example, if we want to encode the numbers 1, 2, 3, 1024, 4, 5, and 2048. We can encode 1, 2, 3, and 4 using 3 bits and the other numbers using 11 bits at the end. Since we don't want to lose direct access, we must precalculate the number of integers in the overflow area and then integrate this into our encoding.
Here, we have 2 integers in the overflow area, which requires 1 bit to be encoded. We will use 1 bit of the encoding to express the fact that we are not directly representing a number but a position in the overflow area. If 1 corresponds to the overflow area, and x-y means that the first bit is x and the others are y, we will represent the sequence of numbers 1, 2, 3, 1024, 4, 5, 2048
as 0-1, 0-2, 0-3, 1-0, 0-4, 0-5, 1-1, 1024, 2048

Arrange your code and create a factory to manage the creation of the compression type based on a single parameter.

You must include some benchmarks, explaining their relevance.

The project must
- be developed in java or python (no jupiter python)
- be associated with a github
- the github must contain an explanation (written in markdown) of how to use the program
- the github must contain a report in pdf format. The report must describe the problems, present your solution and comment on the choices made.
- project end date is 2 Nov 2025 23:59 AoE
- the github repository web address must be send be email to JC Regin (jcregin@gmail.com) before the end of the project. The subject of the email must only "SE github".
- The name of the authors must be added in the github and in the report

Bonus: consider negative numbers and explain the problem they cause and propose some solutions

The project is individual, and must be designed, implemented and documented entirely and exclusively by the author.

I reserve the right to ask for an oral presentation to obtain further information.