Testing BitVectors

Overview

Test the `BitVector` classes that have been written by your colleagues for PA1 and PA2.

Context

Assume that you are a software engineer in a large company that needs a BitVector package. You do not know much about the actual workloads as the system that will use the package is not yet implemented, but you have been given two usage scenarios by the guy two cubicles down. Several competing implementations are available but you must choose one.

The goal of this assignment is to make you think about the different facets of software quality:

1. Correctness: by exhaustively testing the interface of the two classes implemented in this assignment.
2. Efficiency: by developing a benchmark suite and measuring performance of different aspects of the runtime behavior of the code.
3. Code quality: by assessing the quality of the code written by your colleagues and listing the 10 most common mistakes.

Requirements

1. You must write (a) a test suite, (b) a suite of performance benchmarks, and (c) a short (3 pages) report that a technically savvy manager could use as basis to argue for her choice of a BitVector package. (This report should be technical, no hype required.)
2. The test suite should be minimal. No redundant test should be included (in other words don't test the same bug twice).
3. The performance benchmark suite should evaluate the efficiency of relevant operations. It is up to you to determine what 'efficiency' and 'relevant' mean. But you should motivate the choice in your report. You should also motivate any assumptions made during testing.
4. The report should (at least) contain the following information: (I) a ranking of the performance and correctness of each implementation, together with an explanation of the formula used to compute it, (II) a summary

of test and benchmark results, with a short explanation of the tests and benchmarks (e.g. you can group the test and give a paragraph per group), (III) a section describing the ten most common problems found in the source code and arguing about their importance. Finally, (IV) the report should recommend one implementation of serialization and motivate the choice. Any additional data is welcome, and clarity is essential: anything that can help the reader understand the data in question will be appreciated.