Adding Data-Aware Sort Optimizations to the Python Interpreter

Gorokhovsky, Elliot

Python's list.sort() is uses an extremely sophisticated and highly tuned algorithm, designed to minimize the number of comparisons necessary to sort lists with real-world structure. However, no thought is given to the cost of the individual comparisons: the only compare function available to the sort routine is the default object compare. This is unfortunate, since comparisons in any dynamic language, Python included, are extremely expensive: before two objects can be compared, they must first be type-checked, and possibly subjected to other safety checks, to ensure the appropriate compare implementation is used. In sorting, this is usually entirely unnecessary, since real-world lists with well-defined orderings are almost always type-homogeneous. We demonstrate, operating under weak assumptions about the distribution of real-world data, that by instead safety-checking in a single pre-sort pass, it is possible to overcome much of the overhead of dynamic dispatch in comparisons. While this idea could be implemented for any dynamic language, in this project, it was implemented in the CPython interpreter as a simple patch, with benchmarks demonstrating speed-ups of 30-50% for data types commonly encountered in practice. These numbers suggest that taking the time to prove (at runtime) assumptions about real-world data, in the case where all the relevant objects are available for examination in advance, is an extremely useful method for optimizing the execution of algorithms in dynamic languages. As of this writing, these changes have been approved for inclusion in Python 3.7.0 alpha 1.

Awards Won:
First Award of $5,000
Fondazione Bruno Kessler: Award to Travel to Trento, Italy to participate in summer school "Web Valley"
Association for Computing Machinery: First Award of $1,000