Foursum Report

Exhaustive search

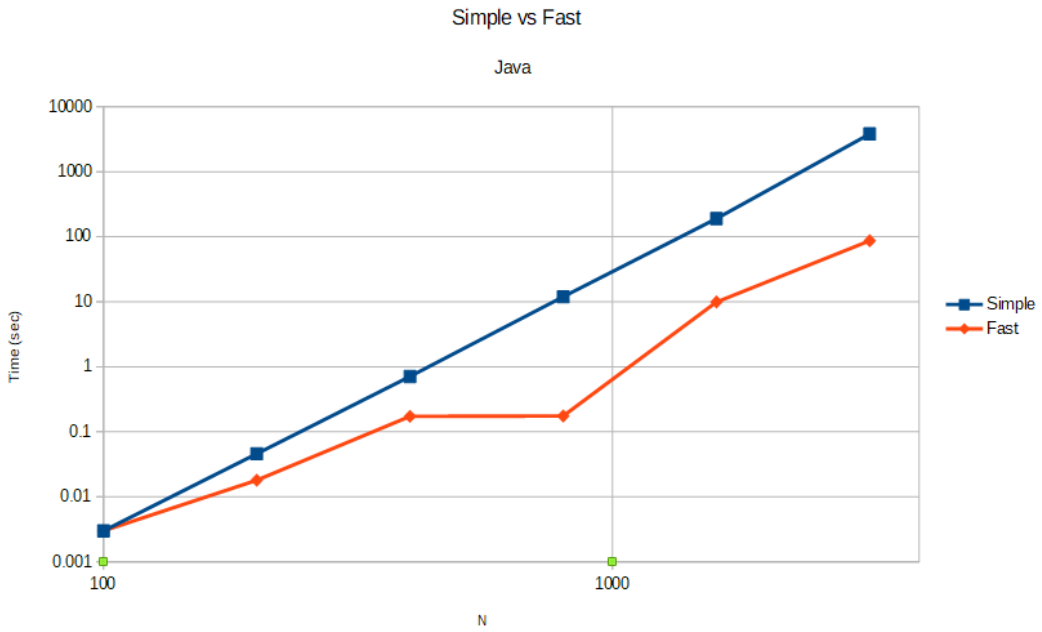
Our program `Simple.java` solves the Four-sum problem using four nested loops.

Being `Simple.java` a brute-force Four-sum implementation we can bound the number of array accesses by ~ 1/3 N^3.

Experiments

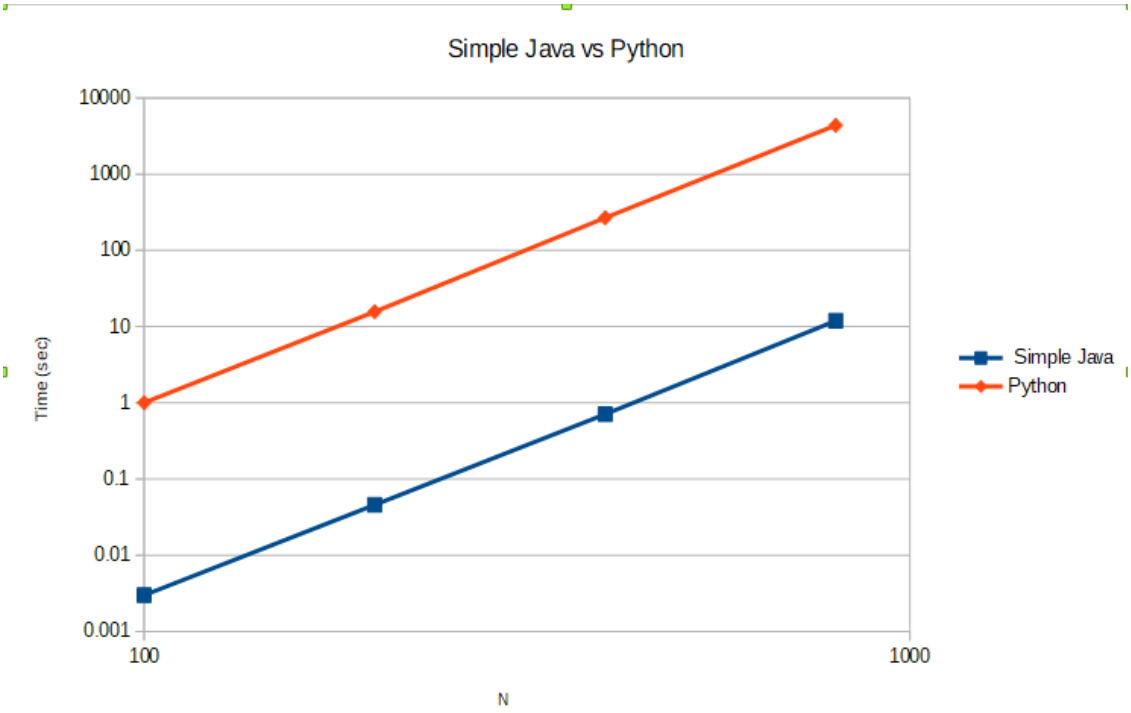
The following table summarises the empirical performance data on the input files in the data directory. We have run each file once, and report the minimum, maximum, and average running time over the files for each input size. Note that all tests have been conducted on the …-4.txt files. I.e. ints-100-4.txt, ints-200-4.txt and so on. The first 100-800 had several warm-ups before the actual benchmarking tests, however the warm-up was reduced to a single warm-up in test 1600, and completely removed from 3200, because of time restrains.
The 'Times ran' columns are the amount of time we ran our code, to find the min, max and avg..
It was especially interesting to experience the difference between the algorithms in real time. The time spend exploded at N=3200 for the Simple algorithm. This four-sum algorithm took over 60 mins to finish for the Simple, whereas the faster one, got the job done in just 86 seconds.

N	Times ran	Min	Max	Simple	Times ran	Min	Max	Fast
100	1000	0,002	0,009	0,003	1000	0,002	0,006	0,003
200	100	0,044	0,067	0,046	100	0,017	0,028	0,018
400	100	0,694	0,789	0,710	100	0,171	0,184	0,173
800	4	11,81	12,19	11,94	40	0,133	0,198	0,175
1.600	2	190,4	190,8	190,6	6	9,862	9,993	9,892
3.200	1	3808	3808	3808	2	86,61	87,07	86,80



And here are the results comparing the simple Java and the Python implementations:

	Java Simple			*Python*		
File	Min	Max	Avg.	Min	Max	Avg.
100-4	0,002	0,009	0,003	1.005	1.009	1.006
200-4	0,044	0,067	0,046	15.1	16.6	15.7
400-4	0,694	0,789	0,710	232	287	268
800-4	11,81	12,19	11,94	-	-	4368
1600-4	190,4	190,8	190,6	-	-	-
3200-4	-	-	3808	-	-	-



Improvements

Using the binary search-based idea sketched in [SW, 1.4] for the Three-sum problem, we can improve our running time to ~ N3 log N.
There were also improvements in the Python implementation of Four-sum: the brute-force version was excessively slow, so two faster versions have been added, using the “Itertools” library. These two faster implementations reduce the running time by 50-70% already on very small inputs. For this reason, it was chosen to use the Itertools faster version of the Python program, in order to compare its performance with Simple.java.
The above tables report our maximum running times on the previously mentioned test inputs.
The resulting logarithmic plot shows these values graphically. Note that the points are on a line of slope 0.6.