DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN
UNIVERSIDAD POLITÉCNICA DE VALENCIA

P.O. Box: 22012 E-46071 Valencia (SPAIN)

Informe Técnico / Technical Report

Ref. No.: DSIC-II/10/09
Pages: 2

Title: An Empirical Evaluation of Algorithmic Debugging Strategies

Author(s): Josep Silva

Date: October 14, 2009
Keywords: Algorithmic Debugging, Strategies

Vº Bº
Leader of research Group

Author(s)

An Empirical Evaluation of Algorithmic Debugging Strategies

1 A Comparison of Algorithmic Debugging Strategies

This article presents a collection of experiments that were conducted in order to measure and compare the performance of a set of algorithmic debugging strategies. This empirical study aims at determining precisely how the strategies work in practice. For instance, theoretical studies [4] concluded that Hirunkitti's version of D&Q is better than Shapiro's version. But the question is *How much better?* Similarly, the theoretical study revealed that Heaviest First is better than Top-Down; but, again, *How much better? Is the difference significant in practice?*

We conducted several experiments to answer these questions. Basically, the experiments were done by debugging several programs with all the strategies in order to know which strategies work better in practice (as an average). The main problem was that no debugger implemented all the strategies—in fact, the most advanced debugger was Mercury's debugger (http://www.cs.mu.oz.au/research/mercury) [3] which only implements four strategies—; and also that there was no maintained implementation of some strategies such as Divide by YES & Query.

Therefore, we had to implement all the strategies in order to ensure that all of them can be applied to the same target language, and hence the results are comparable. We implemented the strategies in the JDD debugger (http://www.dsic.upv.es/∼jsilva/JDD) [1] which is a Java debugger; and we selected 25 Java benchmarks to perform the experiments.

The results are shown in Figure 1, where rows represent benchmarks, and columns represent algorithmic debugging strategies. The strategies are the following: Top-Down (TD), Divide & Query by Shapiro (D&Q Sha), Divide & Query by Hirunkitti (D&Q Hir), Divide by YES & Query (DbyY&Q), Heaviest First (HF), Less YES First (LYF), Hat Delta YES (HD - Yes), Hat Delta NO (HD - No), Hat Delta Average (HD - Avg), Single Stepping (SS).

The results of the comparison provide interesting information related to the strategies which confirm the theoretical study; but they also offer additional information that complements the theoretical study. First, the experiments confirm that single stepping is not usable in practice. It needs many more questions than any other strategy for larger programs. Top-Down, which is the most used strategy is also not a good option. For instance, in the experiments it needs 50% more questions than Divide & Query. The experiments confirm that hirunkitti's Divide & Query is better than Shapiro's Divide & Query; but the difference is very small. The idea of using rules to prune the tree turns out to be useful. Indeed, Divide by Rules and Query—which had not been implemented by any debugger—shows to be the best strategy in practice. The other strategies got similar results. In the case of Hat Delta heuristics, we see that it is better to prune the tree by counting YES answers. This result confirms the same re-

sult obtained by Davie and Chitil [2] when they compared the three Hat Delta heuristics.

Benchmark	Nodes	TD	D&Q Sha	D&Q Hir	DbyY&Q	HF	LYF	HD - Yes	HD - No	HD - Avg	SS
NumReader	11	10	4	4	4	9	9	10	10	10	5
Orderings	65	4	3	3	6	4	5	4	4	4	1
Factoricer	62	9	7	7	6	9	9	9	9	9	14
Sedgewick	39	23	4	3	3	5	5	22	8	8	19
Clasifier	30	14	13	14	14	15	15	5	14	5	13
LegendGame	96	23	24	22	22	21	22	23	23	23	86
Cues	18	15	15	15	15	16	16	4	15	4	15
Romanic	123	14	7	8	7	9	9	14	14	14	23
Transformation	105	33	10	10	10	11	22	8	10	23	39
Risk	68	64	63	61	61	61	62	61	61	59	68
TestMath	43	14	4	4	4	7	7	12	14	12	23
TestMath2	205	22	7	7	4	8	7	12	22	12	108
Figures	72	15	3	4	4	5	6	10	15	10	38
FactCalc	34	19	6	4	13	5	5	10	19	10	21
SpaceLimits	126	25	23	18	18	19	19	11	25	11	43
FactTransf	23	6	8	8	8	9	9	6	6	6	6
RndQuicksort	35	9	3	4	3	6	6	9	9	9	7
BinaryArrays	73	22	21	21	21	22	22	22	22	22	70
Fib-Fact-Ana	243	7	8	8	4	9	5	6	7	6	204
NewtonPolino	45	4	2	3	2	4	3	4	4	4	2
RegresionTest	17	4	4	4	3	6	5	4	4	4	3
BoubleFibArrays	31	5	3	3	3	5	5	5	5	5	5
ComplexNumbers	67	5	5	5	5	6	6	4	5	4	3
StatsMeanFib	89	5	11	10	4	3	3	5	5	5	88
Integral	24	3	2	2	2	3	3	3	3	3	1
AVERAGE	69,8	15,0	10,4	10,1	9,8	11,1	11,4	11,3	13,3	11,3	36,2

Fig. 1. Performance of algorithmic debugging strategies

All the information related to this experiment, together with the source code of the benchmarks is publicly available at:

`http://www.dsic.upv.es/~jsilva/algdeb`

References

1. R. Caballero. Algorithmic Debugging of Java Programs. In *Proc. of the 2006 Workshop on Functional Logic Programming (WFLP'06)*, pages 63–76. Electronic Notes in Theoretical Computer Science, 2006.
2. T. Davie and O. Chitil. Hat-delta: One Right Does Make a Wrong. In *Seventh Symposium on Trends in Functional Programming, TFP 06*, April 2006.
3. I. MacLarty. *Practical Declarative Debugging of Mercury Programs*. PhD thesis, Department of Computer Science and Software Engineering, The University of Melbourne, 2005.
4. J. Silva and O. Chitil. Combining Algorithmic Debugging and Program Slicing. Technical Report DSIC-II/04/06, UPV, 2006. Available from URL: `http://www.dsic.upv.es/~jsilva/research.htm#techs`.