Proposal To Include Java

As A Language For The IOI.

The following proposal for the inclusion of Java as a new language for the IOI competition is made in the light that the technical difficulties that militated against its inclusion in earlier discussions have now been removed.

Background

Java has recently replaced Pascal as the language taught in South African schools. As a result, many potential IOI contestants are not familiar with any of the official IOI languages. This year, one student who was in the running for a place on the South African IOI team withdrew because she did not have the time to learn a new language.

As Java becomes more common, there will be a growing number of contestants who are either forced to learn a new language for the IOI or who are turned away because they are unable to master a new language in time.

We propose that Java be added as a 4th language at the IOI, alongside Pascal, C and C++.

Compilers

The gcj front-end to the GCC compiler suite provides a suitable compiler for the execution environment. It is shipped as part of gcc 3.x (which is used at IOI 2003), and produces machine-native binary code using the same back-end as gcc and g++. Since it produces native code, it can be handled in the grading system almost identically to C and C++.

The free interpreter Kaffe (which uses just-in-time compilation) was also considered. However it is significantly slower than gcj in some cases, and the use of an interpreter adds complications to the grading system.

The gcj front-end does not currently have a DOS/Windows port. However as Java is a well-defined language there should be no problems with using an alternative compiler and interpreter on the development machines.

IDE's

There are many free IDE's suitable for use as IOI competition development environments all supported under windows and Linux. Eclipse (IBM- open source) is one example – it is an open extensible IDE for building integrated development environments (IDEs) for languages like C++ and Java. Jcreator is another popular free IDE. There is good support for Java in Emacs. Jedit is a popular free extensible editor with good support for editing and debugging.

Possible objections

Java is too slow

We have run several benchmarks using equivalent solutions to several problems in the South African Olympiad program. In our test environment, gcj 3.3.1 was roughly 1.6 to 2.4 times slower than gcc/g++ 3.3.1 and 1.1 to 2.0 times slower than Free Pascal 1.0.6. Pascal was up to 1.5 times slower than C/C++. See attached graphs for details.

The IOI seldom sets time limits that require detailed fine-tuning to be met. Instead, any algorithm from a particular complexity class with a reasonable implementation should be able to easily meet the constraints, while a poor algorithm will be far outside the constraints. The difference in speed should not prevent the same time limit from being applied across all languages.

Java uses too much memory

As for the issue of execution speed, the IOI does not generally aim to make the memory limit a matter of squeezing out every byte. It is intended to constrain the class of algorithms. In many cases the memory limit has no impact on the problem, and exists largely to prevent programs from consuming all the memory on the grading system.
In our experience supporting Java in the SACO, memory constraints have never been an issue. With the IOI now using limits of 64MB by default (as opposed to 16MB in the SACO), the overhead becomes even less of an issue.

Java is difficult to support in the environment

The gcj front-end for gcc produces native binary code, and hence can be used in place of gcc, g++, or fpc with a substitution of command line template. Java is supported in the South African Computer Olympiad (SACO), and the only extra code in the evaluation engine is to support interpreted Java. Russ Cox has indicated that supporting Java in the IOI2003 grader will not pose any problems that cannot be solved.

Java is too powerful relative to the other languages

An objection to Java in the past has been that it provides a large number of utility classes, such as linked lists, trees, maps and so on. However with the change to GCC in 2001, the Standard Template Library became available to C++ programmers. This library supports many of the same features as the Java standard libraries, and so Java programmers would receive no benefits over C++ programmers in this area. Java programmers would have an advantage over Pascal and C programmers, but this is the same advantage that C++ programmers currently enjoy.

Java does simplify memory management. However memory management is seldom an issue in IOI problems, as arrays can be allocated with a fixed maximum size and other data structures tend to be grow-only.

Students who use Java may be at a disadvantage

Since Java is slower than Pascal and C/C++ and uses more memory, some tasks may be easier to do in a language other than Java. However, we feel that giving students more options cannot be a disadvantage. Students would lose nothing as they are still free to use other languages.

There is no need for Java

Some countries have indicated that very few of their students have a need for Java, while others (such as South Africa) would benefit from Java. However, supporting Java as an IOI language does no harm to those countries with have no need for it.

Appendix A:

Benchmarking Of Java For IOI-Style Problems

The following compiler versions and options were used. For gcc/g++ and Free Pascal, these are the same options used at IOI 2003. The versions are not identical but are from the same major branch.

- gcc/g++ 3.3.1: -pipe -O2 -lm
- fpc 1.0.6: -So -O1 -XS
- jikes 1.15 and kaffe 1.0.5: no options
- gcj 3.3.1: -pipe -O2 -fno-bounds-check -fno-store-check

The last two options for gcj eliminate some safety checks that Java would normally perform.

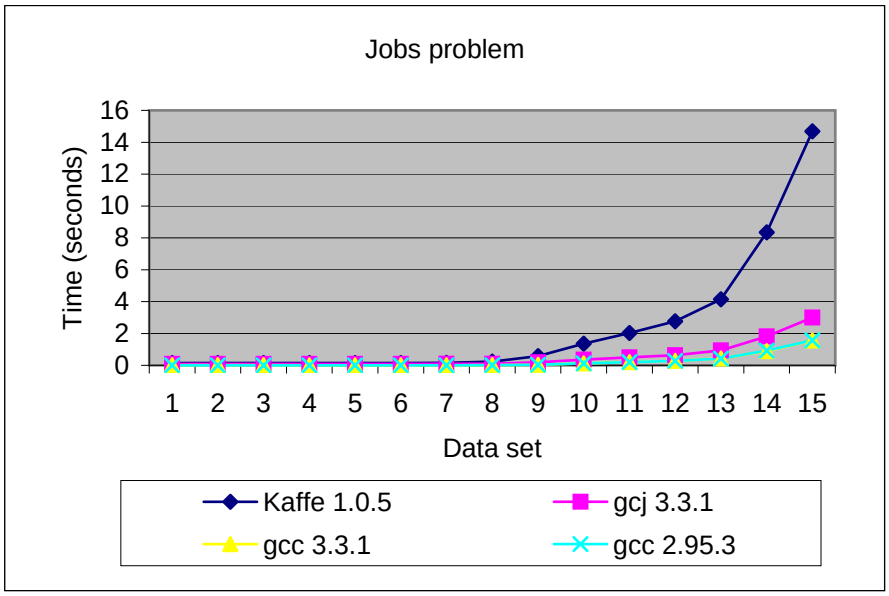
The compilers were run on three algorithms, which were translated line by line. The 'sticks' algorithm was not included for Free Pascal as it does not have a sort function.

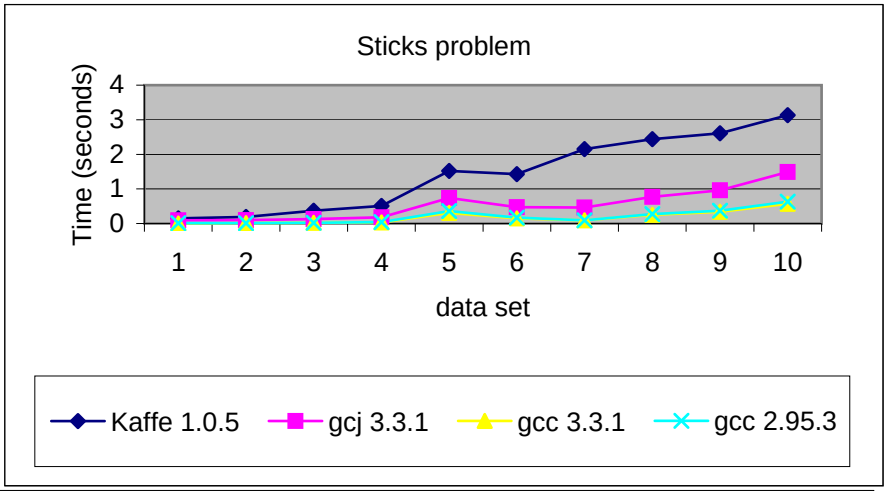
1. Jobs: a two-processor scheduling problem where the dependency graph is a tree. The solution requires linear time in the size of the input, and consists largely of array accesses and very simple arithmetic.
2. Sticks: given a set of horizontal and vertical line segments, count the number of intersections and count the components in the graph formed with segments as vertices and intersections as edges. The algorithm sorts the sticks by left endpoint, then applies a plane sweep. This is intended to be a worst-case scenario for Java, as invoking a call-back function for comparison will be slow (a dynamic function call and a typecast with run-time type checking).
3. Sticks2: the same problem as Sticks, but using a naïve algorithm. It tests every pair of line segments for intersection (order • (n^2)) and an $O(n^2)$ algorithm to track components. This was used to compare the compilers for algorithms that make heavy use of arithmetic rather than of array manipulation.

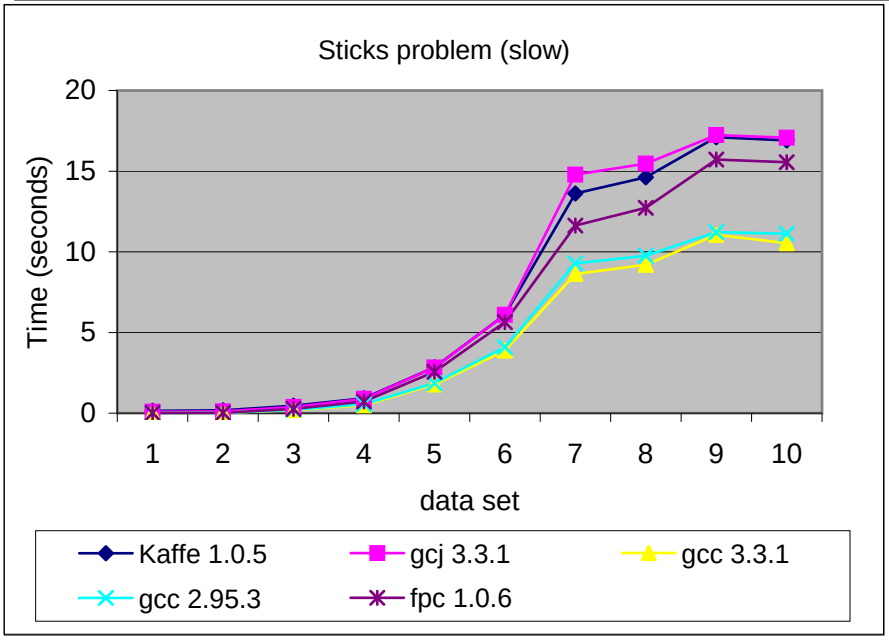
The problems were allowed to run for longer than is normal at an IOI. This was done to minimise the influence of constant-time overheads and noise in the timing.

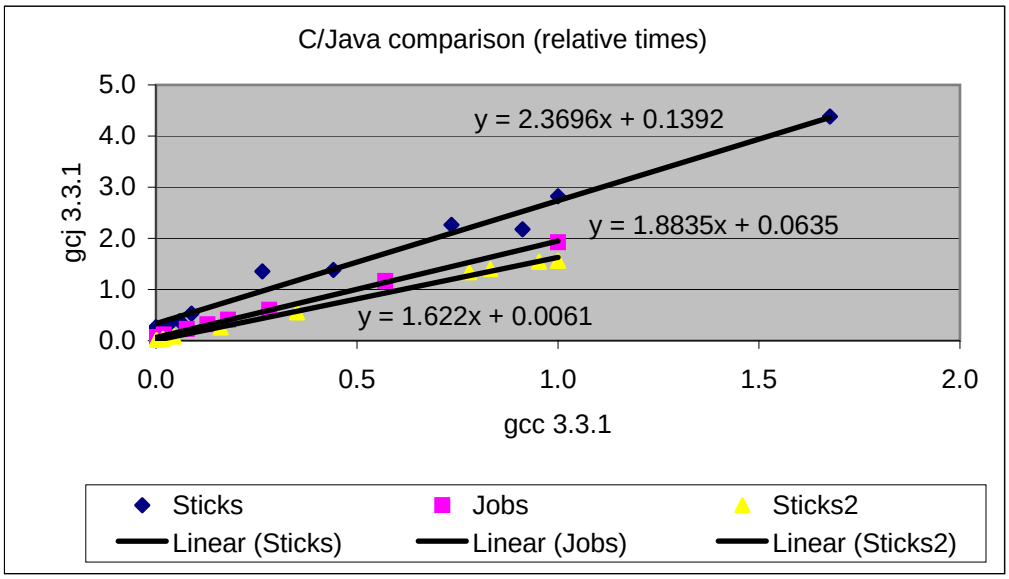
	Kaffe 1.0.5	gcj 3.3.1	gcc 3.3.1	gcc 2.95.3	fpc 1.0.6
Jobs					
1	0.14	0.09	0.01	0.00	0.00
2	0.15	0.09	0.01	0.00	0.00
3	0.15	0.09	0.01	0.00	0.00
4	0.14	0.09	0.00	0.00	0.00
5	0.15	0.09	0.00	0.00	0.00
6	0.15	0.09	0.00	0.00	0.00
7	0.16	0.10	0.00	0.00	0.00
8	0.23	0.10	0.01	0.01	0.01
9	0.58	0.19	0.03	0.03	0.03
10	1.36	0.36	0.12	0.12	0.12
11	2.03	0.50	0.20	0.20	0.20
12	2.76	0.64	0.28	0.28	0.27
13	4.15	0.94	0.44	0.42	0.42
14	8.34	1.82	0.89	0.93	0.87
15	14.69	3.00	1.56	1.57	1.48
Sticks					
1	0.15	0.09	0.00	0.01	
2	0.19	0.10	0.01	0.01	
3	0.37	0.13	0.02	0.02	
4	0.51	0.18	0.03	0.04	
5	1.52	0.74	0.31	0.36	
6	1.43	0.47	0.15	0.17	
7	2.15	0.46	0.09	0.09	
8	2.44	0.77	0.25	0.27	
9	2.61	0.96	0.34	0.37	
10	3.13	1.49	0.57	0.63	
Sticks2					
1	0.14	0.09	0.01	0.01	0.01
2	0.18	0.11	0.02	0.02	0.02
3	0.46	0.39	0.19	0.21	0.26
4	0.92	0.89	0.50	0.56	0.73
5	2.84	2.83	1.78	1.86	2.56
6	6.09	6.09	3.88	4.09	5.64
7	13.62	14.78	8.62	9.28	11.63
8	14.62	15.47	9.20	9.76	12.73
9	17.11	17.24	11.06	11.22	15.72
10	16.91	17.07	10.54	11.12	15.56

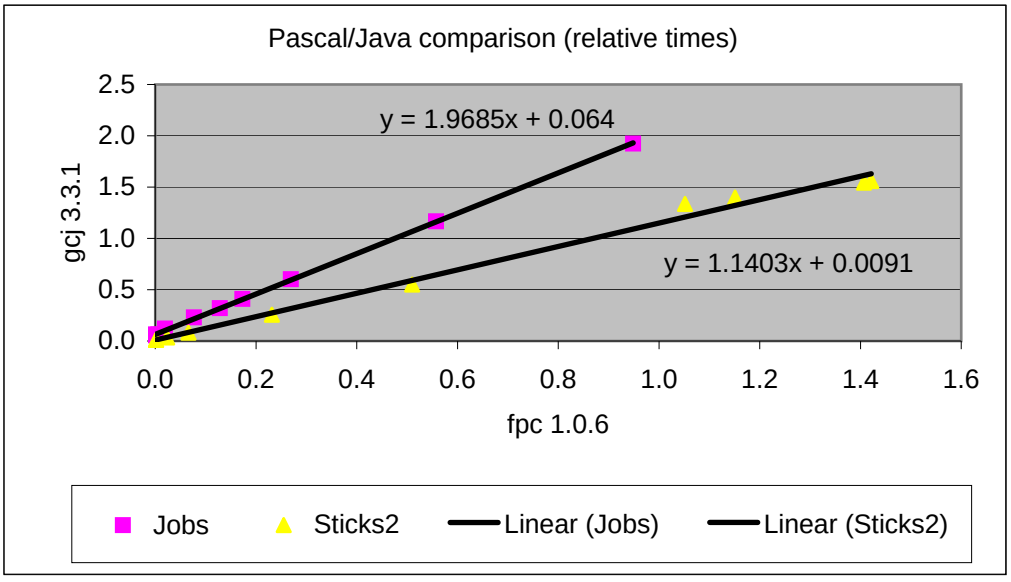
gcc	-pipe -O2 -lm
gcj	-pipe -O2 -fno-bounds-check -fno-store-check
fpc	-So -O1 -XS

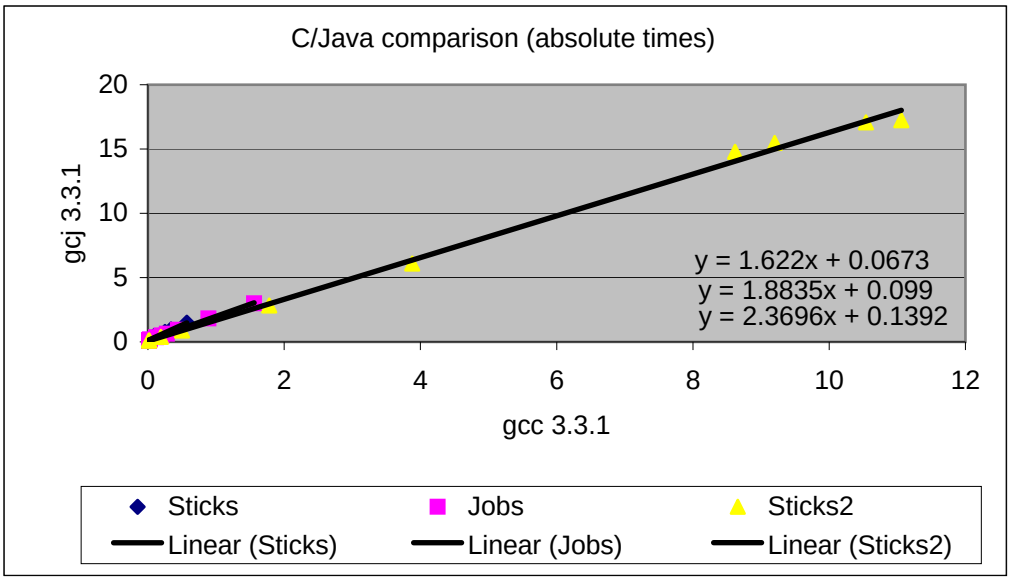

Jobs problem
Time (seconds)
0
2
4
6
8
10
12
14
16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Data set
Kaffe 1.0.5
gcj 3.3.1
gcc 3.3.1
gcc 2.95.3

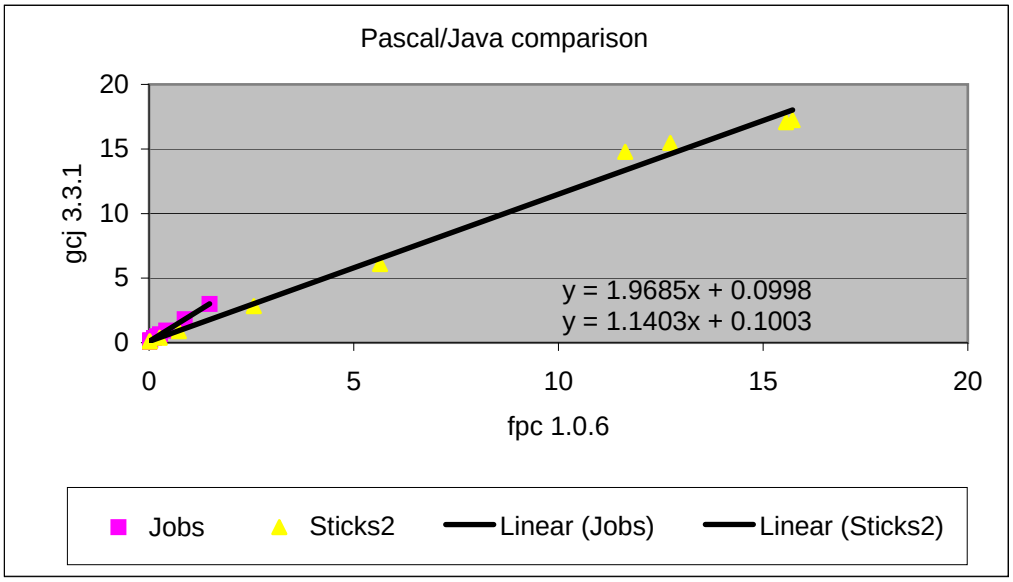

Sticks problem
Time (seconds)
0
1
2
3
4
1 2 3 4 5 6 7 8 9 10
data set
Kaffe 1.0.5
gcj 3.3.1
gcc 3.3.1
gcc 2.95.3


Sticks problem (slow)
Time (seconds)
0
5
10
15
20
1 2 3 4 5 6 7 8 9 10
data set
Kaffe 1.0.5
gcj 3.3.1
gcc 3.3.1
gcc 2.95.3
fpc 1.0.6


C/Java comparison (relative times)
y = 2.3696x + 0.1392
y = 1.8835x + 0.0635
y = 1.622x + 0.0061
gcj 3.3.1
gcc 3.3.1
5.0
4.0
3.0
2.0
1.0
0.0
0.0
0.5
1.0
1.5
2.0
Sticks
Jobs
Sticks2
Linear (Sticks)
Linear (Jobs)
Linear (Sticks2)


Pascal/Java comparison (relative times)
y = 1.9685x + 0.064
y = 1.1403x + 0.0091
gcj 3.3.1
fpc 1.0.6
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
Jobs
Sticks2
Linear (Jobs)
Linear (Sticks2)


C/Java comparison (absolute times)
gcj 3.3.1
gcc 3.3.1
y = 1.622x + 0.0673
y = 1.8835x + 0.099
y = 2.3696x + 0.1392
0
5
10
15
20
0
2
4
6
8
10
12
Sticks
Jobs
Sticks2
Linear (Sticks)
Linear (Jobs)
Linear (Sticks2)


Pascal/Java comparison
gcj 3.3.1
fpc 1.0.6
y = 1.9685x + 0.0998
y = 1.1403x + 0.1003
0
5
10
15
20
0
5
10
15
20
Jobs
Sticks2
Linear (Jobs)
Linear (Sticks2)

Proposal by South Africa to include Java as IOI programming language

	Kaffe 1.0.5	gcj 3.3.1	gcc 3.3.1	gcc 2.95.3	fpc 1.0.6
Jobs					
1	0.0897	0.0577	0.0064	0.0000	0.0000
2	0.0962	0.0577	0.0064	0.0000	0.0000
3	0.0962	0.0577	0.0064	0.0000	0.0000
4	0.0897	0.0577	0.0000	0.0000	0.0000
5	0.0962	0.0577	0.0000	0.0000	0.0000
6	0.0962	0.0577	0.0000	0.0000	0.0000
7	0.1026	0.0641	0.0000	0.0000	0.0000
8	0.1474	0.0641	0.0064	0.0064	0.0064
9	0.3718	0.1218	0.0192	0.0192	0.0192
10	0.8718	0.2308	0.0769	0.0769	0.0769
11	1.3013	0.3205	0.1282	0.1282	0.1282
12	1.7692	0.4103	0.1795	0.1795	0.1731
13	2.6603	0.6026	0.2821	0.2692	0.2692
14	5.3462	1.1667	0.5705	0.5962	0.5577
15	9.4167	1.9231	1.0000	1.0064	0.9487
Sticks					
1	0.4412	0.2647	0.0000	0.0294	
2	0.5588	0.2941	0.0294	0.0294	
3	1.0882	0.3824	0.0588	0.0588	
4	1.5000	0.5294	0.0882	0.1176	
5	4.4706	2.1765	0.9118	1.0588	
6	4.2059	1.3824	0.4412	0.5000	
7	6.3235	1.3529	0.2647	0.2647	
8	7.1765	2.2647	0.7353	0.7941	
9	7.6765	2.8235	1.0000	1.0882	
10	9.2059	4.3824	1.6765	1.8529	
Sticks2					
1	0.0127	0.0081	0.0009	0.0009	0.0009
2	0.0163	0.0099	0.0018	0.0018	0.0018
3	0.0416	0.0353	0.0172	0.0190	0.0235
4	0.0832	0.0805	0.0452	0.0506	0.0660
5	0.2568	0.2559	0.1609	0.1682	0.2315
6	0.5506	0.5506	0.3508	0.3698	0.5099
7	1.2315	1.3363	0.7794	0.8391	1.0515
8	1.3219	1.3987	0.8318	0.8825	1.1510
9	1.5470	1.5588	1.0000	1.0145	1.4213
10	1.5289	1.5434	0.9530	1.0054	1.4069