Compiling for Java: Problem Sheet

Jeremy Singer

1. **(easy)** Study this table of compiler characteristics:

name	compile time for n bytecodes	execution time for generated code
interpreter	0	t
C1 compiler	$5n$	$0.25t$

and also study this table of method characteristics:

method	number of bytecodes	interpreter execution time, per call
`foo()`	30	100
`bar()`	50	120
`baz()`	1000	500

Suppose each method has been executed 10 times so far.

(a) [2 points] Which of the three methods is the *best* candidate for recompilation?

(b) [3 points] Which of the candidates should *not* be recompiled? Why not?

2. [3 points] **(difficult)** Consider this inequality that must hold for a method to be recompiled with the optimizing compiler:

$$t_{\text{opt\_recompile}} + t_{\text{opt\_execution}} < t_{\text{default\_execution}} \quad (1)$$

where $t_{\text{opt\_recompile}} = 20n$ time units, for a method with n bytecode instructions and $t_{\text{opt\_execution}} = 0.1t_{\text{default\_execution}}$

What is the execution time threshold for a method, below which the opt compiler will never be triggered?

3. [1 point] **(moderate)** In a multi-core system, how might we reduce the observable pause time due to JIT compilation?

4. [1 point] **(moderate)** Why might a *hot* method become *cold* during program execution?

5. [2 points] **(difficult)** In this presentation, we have used the *method* as the unit of recompilation. Which other program structures might be suitable for recompilation?