PROJECT REPORT-SUDOKU SOLVER

Mehrnoosh Shakarami ,Prajakta Dhanawade, Shruti Desai(Group 5)

Abstract

The purpose of this document is to present implementation of Sudoku Solver puzzle-in Java, C, Python, Prolog, Javascript programming languages. Invoking different language implementations using Java driver program and analysis based on comparison of time taken to execute each of the above approaches.

Comparison on the basis of Time

On giving single input text file we get following time calculations:

Time taken to

Language	**Time taken(in seconds)**
C	0.000Sec
Java	0.012Sec
Python	0.459Sec
Javascript	2.368Sec
Prolog	0.031Sec

Analysis on the basis of time:

The time required for executing Sudoku solver in different languages increments as follows:
C < Java <Python< Prolog <Java Script .

- C language takes less time compared to all other language as C does not have garbage collection ,dynamic typing and other facilities which makes other languages easy to write .For example in our program we index into an array, in Java it takes some method call in the virtual machine, bound checking and other sanity checks. But in C, even trivial things are not put in safety and that makes C faster compare to java even though we are calling C code from java.
- Java is built on C, Python is built on C (or Java, or .NET, etc.) etc. Some things in C are still written in Assembly language, which tends to be even faster.
- Python programs are generally expected to run slower than Java programs Because of the run-time typing, Python's run time must work harder than Java's
- Prolog is faster compared to java as it has its own debugger,native OS control,C Interface.
- JavaScript finds a home on the server and in the browser and takes most time. The runtime currently supports Objects, Arrays, invoking scripts, calling JS functions from Java and registering Java Functions as callbacks from JS. There is also a small library for converting Objects and Arrays to Java Maps and Lists. Finally, the runtime supports remote debugging which makes it slower compare to other language.

For C, Java and JavaScript, Prolog, we have used Backtracking algorithm for incrementally building candidates to the solutions, and abandoning each partial candidate c.
For Python we have used Constraint propagation a process of finding a solution to a set of constraints that impose conditions that the variables must satisfy
The strategy used is:-
(1) If a square has only one possible value, then eliminate that value from the square's peers. (2) If a unit has only one possible place for a value, then put the value there.

Comparison on basis of number of bank spaces:

Language	**Time taken with Input file with less blank spaces**	**Time taken with Input file with more blank spaces**
C	0.000Sec	0.000Sec
Java	0.000Sec	0.012Sec
Python	0.213Sec	0.459Sec
Javascript	1.256Sec	2.368Sec

Contribution:

Code Implementation in C	Shruti Desai
Code Implementation in Java	Shruti Desai
Code Implementation in Javascript	Prajakta Dhanwade
Code Implementation in Python	Mehrnoosh Shakarami
Code Implementation in Prolog	Mehrnoosh Shakarami
Java to C calling through JNI	Shruti Desai
Java to Javascript Calling	Prajakta Dhanawade
Java to Python Calling	Mehrnoosh Shakarami
Driver Program	Shruti,Prajakta
Report	Mehrnoosh,Prajakta,Shruti

References:

Algorithm for C , Java, Javascript:
http://codereview.stackexchange.com/questions/37430/sudoku-solver-in-c
Algorithm for Python:
http://norvig.com/sudoku.html
Algorithm for Prolog:
http://www.snakedj.ch/2010/09/13/swi-prolog-sudoku-solver/