CS Games 2015 - Parallelism Challenge

Jean-Pierre Deschamps, Félix-Antoine Ouellet

March 14th 2015

1 Background

The challenge will be to code an application that can connect to a simple server and solve the problems sent by it. Points will be given for correct answers and taken for wrong answers.

Since this is a parallelism challenge, the server will send **4** problems from the same category at a time. Thus, participants will have to solve the 4 problems and send the solutions found to the server. A good answer adds X points to a team's score while a bad one withdraws $X/2$. The server will answer with 4 new problems from a new category. The ultimate goal will be to add up the highest number of points within a specified time.

Also, it is important to mention that the server will not wait indefinitely after the contestants. After some time, it will sends 4 new problems and contestants will have points deducted on the assumption that they have not answered quickly enough.

Finally, the final evaluation will take place on a dedicated server with 2 processors, each having a single core.

2 Categories of problems

The following subsections describe each of the 6 categories of problem that you will be asked to solve.

2.1 Maze

In this problem, you are asked to identify if a maze has a solution or not. You will receive 4 different mazes simultaneously. Some will have a solution, others won't. Mazes are encoded in the following way:

size	*data*	*size*	*data*	*size*	*data*	*size*	*data*

Each of the 4 sets of *[size data]* can be decomposed in the following way. The first value will be a 32 bits unsigned integer indicating the number of elements

in a **square** matrix. This means that there will be **N** 32 bits unsigned integers in the matrix. A value of *1* represents a navigable box and *0* represents a wall. The start position will always be *(1, 1)* and the end position will always *($\sqrt{N}$-1, $\sqrt{N}$-1)*. This matrix is restated in its flat line by line. Consequently, for a size N, we have this:

N	*line 1*	*line 2*	···	*line* $\sqrt{N}$

For example, the following data:

36	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*	*0*	*1*	*0*	*0*	*0*	*1*	*1*	*1*	*0*	*0*	*0*	*1*	*0*	*1*	*1*	*0*	*0*	*1*	*0*	*0*	*1*	*0*	*0*	*0*	*0*	*0*	*0*

represent the following maze:

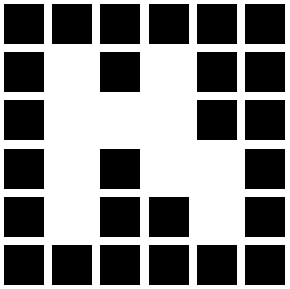


Figure 1: 6x6 maze

The figure represent a zoomed 6x6 pixels image. Black squares are wall, so not walkable. White squares are the walkable area. The start position is the top white square. The end position is the right bottom white square. The answer for this particular maze is *True* because it is possible to go from the start to the end.

The expected answer for this category is a set of 4 bool value : *True* if the maze is solvable, *False* otherwise.

2.2 Sudoku

Sudoku is a grid-like game invented in 1979 by the American Howard Garns but inspired by the Latin square and the 36 officers problem of Swiss mathematician Leonhard Euler. The goal is to fill the grid with a series of different numbers, letters or symbols, which never repeats on the same line, or column or even in a sub-grid. Most of the time, the symbols are numbers ranging from 1 to 9, the sub-grids then being a 3x3 square. However, it is possible to generalize this principle to smaller or larger grids.

In this problem, the goal won't be to solve a sudoku but rather to validate whether the proposed solutions comply with the rules of the game or not. You will receive 4 different sudokus simultaneously. Some will be valid, others won't. Sudokus will be encoded in the following way:

Size	*Data*	*Size*	*Data*	*Size*	*Data*	*Size*	*Data*

Each of the 4 sets of *[size data]* can decomposed in the following way. The first value will be an 32 bits unsigned integer indicating the total number of element in the sudoku. This means that here will be **N** unsigned 32 bits integers.

For example, the following data:

16	*3*	*1*	*2*	*4*	*4*	*2*	*3*	*1*	*1*	*3*	*4*	*2*	*2*	*4*	*1*	*3*

represents the following pseudo-base 4 sudoku:

3	1	2	4
4	2	3	1
1	3	4	2
2	4	1	3

Table 1: 4x4 pseudo-base 4 sudoku

The expected answer for this category is a set of 4 bool value : *True* if the sudoku comply with game rules, *False* otherwise.

2.3 Value in an array

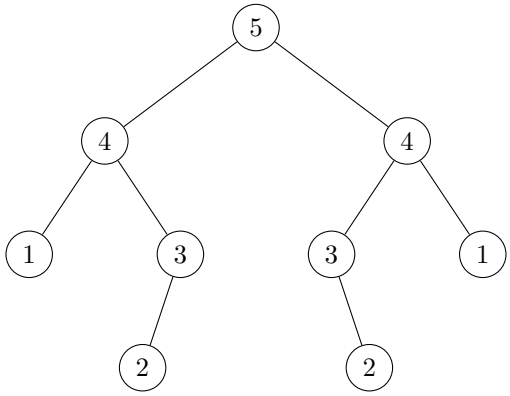
In this problem, we simply ask if a given value is in an array or not. The wanted values and arrays will be encoded in the following way:

value	*size*	*data*	*value*	*size*	*data*	*value*	*size*	*data*	*value*	*size*	*data*

The expected answer for this category is a set of 4 bool value : *True* if the value can be found in the array, *False* otherwise.

2.4 Symmetric Trees

In this problem, you are asked to identify which trees in a set of trees are symmetrical. In other words, which are the trees in which the of the right subtree of the root is the mirror of the left subtree. The figure below provides an example of a symmetrical tree that could be transmitted to you:



In addition, it is important to clarify that the trees used in this problem contain only positive integer values.

Concerning the sent message, the client can be expected to see the data in the following form

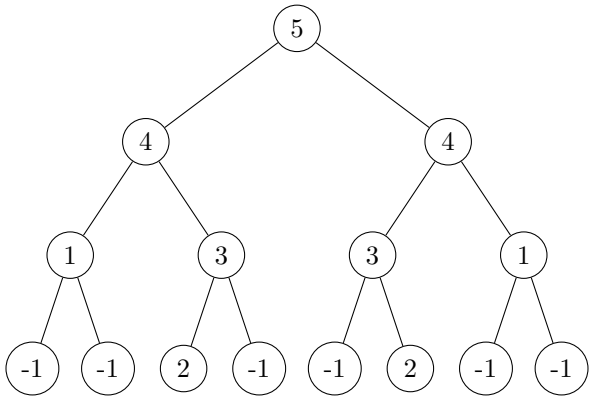
Size1	*Data1*	···	*Size4*	*Data4*

It must therefore be expected to see four times a 32 bits unsigned integer indicating the number of elements in a tree followed by the tree itself. Tree's data are 32 bits signed integers.

Regarding the content of a data section, two things should be clarified. First, even if the tree contains only positive values in its nodes, the client will retrieve negative values in the data section.

Indeed, the null subtrees will be characterised by a value of -1. Second, the values ofthe nodes will be presented following a depth-first traversal or, if you prefer, pre-order traversal.

The combination of these two elements will result in trees for this given problem to look like the one presented below:



This will results in a message's data section resembling the one given below:

5	*4*	*1*	*-1*	*-1*	*3*	*2*	*-1*	*4*	*3*	*-1*	*2*	*1*	*-1*	*-1*

The expected answer for this category is a set of 4 bool value : *True* if the tree is symmetric, *False* otherwise.

2.5 Identical Sequences

As part of this problem, you will be asked to identify which strings of characters are different. Thus, you will receive four versions of a certain character string. In each of these sequences, the characters have been mixed in random way and some of these characters have been set to upper case. Moreover, in one of these chains, some characters have been replaced with new characters. Characters are encoded in UTF8. What distinguish a sequence from another is different characters. Changes in case (UPPER, lower) and location of a single character are not consider as a difference in this problem. Only added or removed character is consider to be a valid difference. A concrete example of what you might received is given below

PASSword
WoRdPaSs
dpRAosWS
pawossrX

Finally, a problem of this category will be sent in a message of the following form:

Size1	*Data1*	···	*Size4*	*Data4*

The first value will be an 32 bits unsigned integer indicating the size the strings. Following this will be 4 sequences of characters.

The expected answer for this category is a set of 4 bool value : *True* if the sequence differ, according to the previous rules, from the others, *False* otherwise.

2.6 Encoding

The goal of this problem is to determine whether a character string, once decoded, is of a given length.

The algorithm used to encode character strings is the Run-Length Encoding algorithm (RLE). To describe shortly, this algorithm transforms a string of characters containing multiple ranges of repeated characters into a chain of characters where these repetitions have been replaced by the number of repetitions and character that repeats. To give a concrete example of the thing, this chain of characters:

aaaaaabbcccccddddddddddddd

would be transformed into this encoded chain of characters:

6a2b5c13d

In a message, a problem of this type has the following format:

Value1	*Size1*	*Data1*	···	*Value4*	*Size4*	*Data4*

As can be seen, the message will begin with an unsigned 32 bits integer indicating the size of the character string sought. Then, we will find a unsigned 32 bit integer indicating the size of the encoded string and the encoded string (UTF8), 4 times.

The expected answer for this category is a set of 4 bool value : *True* if the sequence differ from the others, *False* otherwise.

3 Network protocol

When connecting to the server, it will immediately start to send problem to solve. To do this, the server use a very simple raw protocol.All values are in *little-endian*. Boolean shall be encoded as an unsigned 8bits integer, where the value 1 means *True* and 0 means *False*.

In general, the server sends problems in this form :

ID	*... data bloc ...*

ID is an identifier encoded as a unsigned 32 bits integer. It can only take value in the range $[0, 5]$. This range is guaranteed. This value indicate the received problem's category. The corresponding category for a given ID is described in the following table :

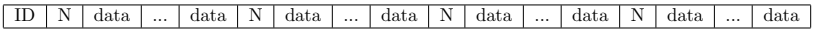
ID	**Type**
0	Maze
1	Sudoku
2	Value in array
3	Symmetric trees
4	Identical Sequences
5	Encoding

The *data bloc* is specific to each problem's categories. Overall, this bloc will be a sequence of 4 set *size, data* where *size* is an unsigned 32 bits integer indicating the number of element in *data*. The complete definition of each problem category is described below.

3.1 Maze

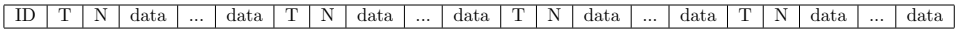
- *ID* is an unsigned 32 bits integer with a constant value of 0.
- *N* are unsigned 32 bits integer indicating the data count. There's 4 of them in the entire message.
- *data* are unsigned 32 bits integer representing maze's positions. There's *N* of them. They can only take a value of 0 or 1.

3.2 Sudoku



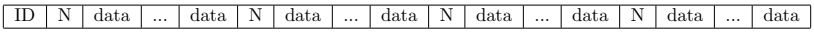
- *ID* is an unsigned 32 bits integer with a constant value of 1.
- *N* are unsigned 32 bits integer indicating the number of values in the sudoku. There's 4 of them in the entire message.
- *data* are unsigned 32 bits integer representing values of the sudoku grid. There's *N* of them.

3.3 Value in an array



- *ID* is an unsigned 32 bits integer with a constant value of 2.
- *T* are unsigned 32 bits integer. They are the wanted values of the beside's array. There's 4 of them in the entire message.
- *N* are unsigned 32 bits integer indicating the array's length. There's 4 of them in the entire message.
- *data* are unsigned 32 bits integer representing values in the array. There's *N* of them.

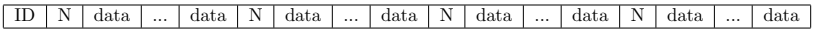
3.4 Symmetric Trees



- *ID* is an unsigned 32 bits integer with a constant value of 3.

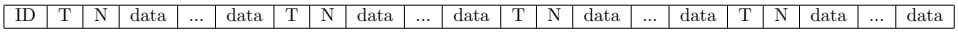
- N are unsigned 32 bits integer indicating the tree's size. There's 4 of them in the entire message.
- *data* are signed 32 bits integer representing values in the tree. There's N of them. They can only take strictly positive values and the special value -1.

3.5 Identical Sequences



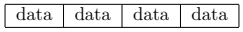
- *ID* is an unsigned 32 bits integer with a constant value of 4.
- N are unsigned 32 bits integer indicating the sequence's length. There's 4 of them in the entire message.
- *data* are UTF8 characters. There's N of them.

3.6 Encoding



- *ID* is an unsigned 32 bits integer with a constant value of 5.
- T are unsigned 32 bits integer indicating the wanted length. There's 4 of them in the entire message.
- N are unsigned 32 bits integer indicating the encoded value's length. There's 4 of them in the entire message.
- *data* are UTF8 characters. There's N of them.

3.7 Answers format



- *data* are bool values encode as unsigned 8 bits integer. There's 4 of them in the entire message. They can only take a value of 0 or 1.

4 Programs

You'll receive a demo client. This program simply allow to connect yourself with the server, receive value and send a random answer to the server using Boost library. To run, the client only need a single parameter : the server's address. The source code will be provided. It is possible to start the client with the following command :

you@0xD:~$./client localhost

You'll also receive a test server. This program allow you to simulate the evaluation's environment on which your program will be tested. This server needs the binary data file names in this specific order : Maze, Sudoku, Value in an array, Symmetric trees, Identical sequences and Encoding. It is possible to start the server with the following command :

you@0xD:~$./server maze_small.bin sudoku_small.bin array_small.bin password_small.bin tree_small.bin RLE_small.bin

Two data sets will be provided, one containing few small problems, the other one with a larger number of big problems. The final evaluation will be done with a secret data set specially built for this.

5 Remarks

We would like to thank Reginald "Reggie" Fils-Aimé for his support in this challenge's creation. His contribution, even modest, is essential for the program efficiency. You'll quickly understand why. We would like to also thank Patrice Roy for his ideas which lead to this competition. Thanks Pat! In case of an accident, please stay calm. Do not Panic. Please stay at your seat. Wait for instructions. Eat your right hand. Proceed with calm and follow instructions carefully. Panic.

Thank you for your collaboration
0xD

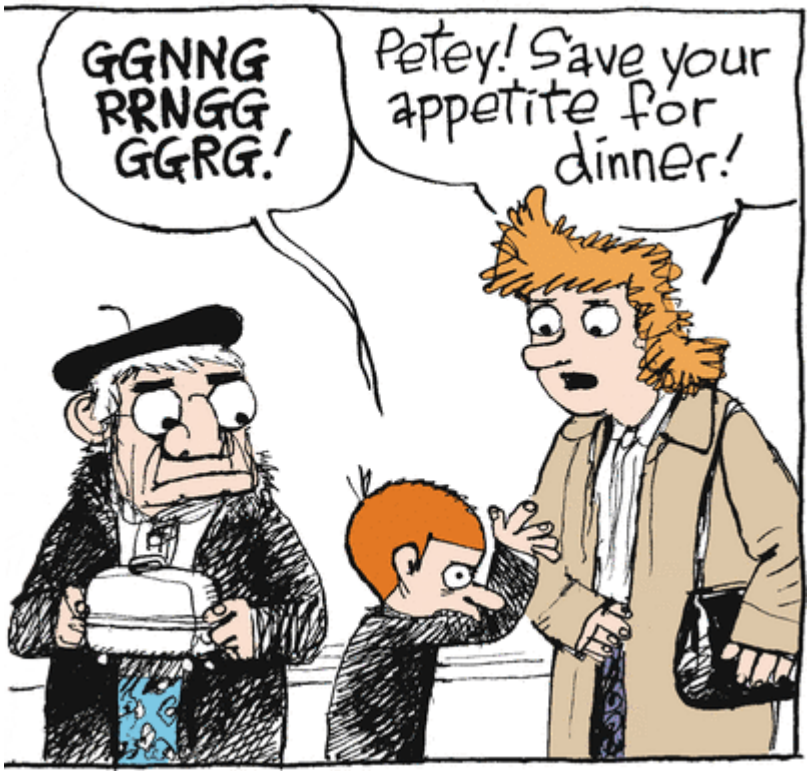


Figure 2: Thompson, Richard. "Cul de Sac", November 23rd 2011