**Spark**

Lets start ASAP wth the following warm up example.

Implement in Spark the following procedure:

1. generated using a random number generator integers from 0 to 99.

Say that this sequence is s= ( 1, 55, 45, 65, 3, 34, 45, ...)

2. generate a large file that has 1 billion TRIPLES of overlapping windows

from this sequence:

S = ( (1,55,45), (55,45,65), (45,65,3), ... )

3. do a computation that reports the total number of appearances of the

triple (11,12,13) in S.This last computation must be in Spark.

How does this sound?

I mean:

if the input is:

a,b,c,d,e,f,g,...

the first triple is:

a,b,c

the second one is:

b,c,d

the third one is:

c,d,e

I have a question about spark cluster. Is it possible to run the cluster on a single laptop? If I am going to use several machines, how many should I use?

Yes, you are going to think that a single laptop is the whole cluster.

The reason is that this is just for DEVELOPMENT purpose.

When you create a PROTOTYPE this just has to be a working

system and how to actually do the actual execution comes later (or never:)).

still a little confused, in spark which of the two modes will be good?

1.local[K] Run Spark locally with K worker threads (ideally, set this to the number of cores on your machine).

2.spark://HOST:PORT Connect to the given Spark standalone cluster master. The port must be whichever one your master is configured to use, which is 7077 by default.

Follow the first one for your implementation and I will explain in person why this is okay.

**compiler**

As discussed you should cover some basic things.

There is a \*single\* textbook that you can get all information.

In fact, that textbook is very practical, but I choose for you other

much simpler sources to cover the basics.

The types of texts you will be reading aren't relevant to our goal

but they are the fastest and easiest way to

(1) cover the basics and

(2) get up to speed so that we can actually get a working prototype.

I suggest that you follow my suggestions with religious commitment.

If by any chance you decide to read things from any other source

then I cannot guarantee that this will resolve your questions.

Most probably it will increase your confusion. I'm sorry but I cannot

waste time correcting things you got from random sources.

(over simplistic)/Simple rule: the following instructions are the only trusted and

quality way to get the truth and nothing else.

Please study in order, and contact me whenever you complete each part:

--------

1-16: just for being able to follow the notation

29-46: you must know exactly what regular expressions are

91-98: you must know exactly what CFGs are. \*You MUST be able to produce CFGs you may be asked to\*.

Do the following exercises (they are simple and chosen with care):

2.1, 2.3, 2.4, 2.9

--------

You \*must\* understand Dynamic Programing

If there are things that confuse you you should contact me ASAP and I'll help you be 100% clear in everything you ask

--------

Think of the following problem:

If you fix in your head a CFG (that is, you will NOT give the CFG as input to your program, but it will be fixed -- choose any CFG you like, make it have 5-6 rules), then write a program in Python that given an input string it outputs YES if this string can be generated by your CFG, or otherwise it outputs NO

This program would be small (no more than 50 lines of code) but it is not trivial to come up with a program that works reasonably fast

(reasonably fast = it finishes its execution within a second after you hit Enter, when your input strings have  reasonable length (e.g. 10,20,50 characters long))

--------

Search in the web to find out about LEX and YACC

Watch online videos, read lectures, and be prepared answer simple questions

I asked Spiros Papadimitriou and he replied with the grammar,

but it turns out that you have to learn the basics of YACC.

I don't want you to use YACC, but you should know YACC in

order to use it with your parser.

So, please spend a day (or half a day) figuring out the syntax

so that you can incorporate the grammar into your own parser.

Here is the grammar description:

<https://github.com/openscad/openscad/blob/master/src/parser.y>

The GitHub source directory (just in case you need it):

<https://github.com/openscad/openscad/tree/master/src>

For examples of grammars you can Google :)

"openscad code examples"

you will also see a wiki (which has some examples)

and you can also find examples in:

[http://www.thingiverse.com](http://www.thingiverse.com/)

You should also find the libraries in Python

for Tokenization and other string handling issues.

I believe Python has tons of ready-to-use such

functions.

I expect that the whole thing is going to take you 2-3

days to finish (of full work). Let us meet up as soon

as you are done.