Universidade Federal de Minas Gerais
Instituto de ciências exatas
Departamento de ciência da computação
Projeto e análise de algoritmos

Practical Work 3

October 30, 2014

1 An spy history

You are a spy for the NSA, and your job is to intercept communications within the network of the center of operations of the Petroleum Agency of North Korea. Your objective is to detect the control commands that enable and disable the devices that control oil production in the country.

You possess the ability to place an attack of the type 'man in the middle' in which you will have access to the direct transmission of bits (0 and 1). However, you will have the hard task of decoding the captured flow of bits to discover the content of the messages. Fortunately, you have the collaboration of a great team of spies and they have already discovered that there are 2 control commands in the flow of bits. They are the sequence of bits "000" and "11111". This information is important because you will focus on decoding the bits that appear between these control commands.

By analyzing the first captured data, you observed the occurrence of many transmission errors, despite the high quality of the spying devices that you used to capture the messages. Those errors prevent you from identifying if some bits are '0' or '1'. Thus, your problem now is to construct a program that receives a string of bits and returns if it has a control command, it does not have a control command or it is impossible to define if the string has a control command. Recall that the discovered control commands are '000' (i.e., a sequence of three bits 0) and '11111' (i.e., a sequence of five bits 1).

2 Input and Output

The program must solve one instance of the problem per execution. Each instance of the problem is a string that can contain the characters '0', '1' or '-'. The characters '0' and '1' are the bits that were identified correctly and the character '-' means a bit that was not identified, i.e., it can be both '0' or '1'. For example, the string '-00011-110' has 2 bits that were not identified at the first and seventh positions.

The input of the program is a file in which the first line contains the number of problem instances and the following lines are the instances, and each instance contains one string that represents the flow of bits. The string contains between 1 and 100 characters. The output of the program must be one text file named 'output.txt' written on the same directory of program execution. This file contains the responses for each instance of the input file respectively, and there are only three types of response: "true", "false" or "both". This response concerns the presence of control commands in the string of bits. Some examples are presented below.

Example

input.txt:

```
4
1011010111
101101-111
1--10101--
00-1111
```

output.txt:

```
false
both
false
true
```

3 Solutions to be implemented

This problem must be solved using three different programming paradigms: **brute force strategy (bf)**, **dynamic programming (db)** and **greedy algorithm (ga)**.

For the solution with PD discuss the properties of optimal substructure and overlapping of sub-problems. It is also important to show the recurrence equation in which the PD solution is based. For the greedy solution, discuss how good is your solution. If it always provides the correct response, prove the correctness of your solution. Otherwise, propose different kinds of tests and discuss the percentage of correct answers provided by your solution given these tests.

4 Instructions about Implementation

You must write the **3 paradigms implementations on different executable files**, as we explain below.

We will accept only implementations in C, C++ and Java. Be sure that your code can be compiled and executed in GNU/Linux machines. The created programs must receive an input file as parameter and provide the result through an output file named 'output.txt' that MUST follow the format described in Section 2 .

In addition to the code that solves the proposed problem with the different paradigms, you must write shell scripts (.sh) to compile and run your code. These scripts must be named:

- *compile_bf.sh* and *execute_bf.sh infile*, for the brute force approach.
- *compile_dp.sh* and *execute_dp.sh infile*, for the dynamic programming approach.
- *compile_ga.sh* and *execute_ga.sh infile*, for the greedy approach.

It follows below some examples of these scripts:

- **compile_bf.sh**

```
#!/bin/bash
gcc main_bf.c -o main
```

- **execute_bf.sh infile**

```
#!/bin/bash
./main_bf $1
```

If you write your code in Java, you can use the command *java -jar program.jar*, where *program.jar* is the executable file of your project generated on *compile.sh*.

The implementations must be tested on computers of the Department of Computer Science where the postgraduate students have free access. This test is to ensure that the implementation will be compiled and executed in an environment known by the student. For example, two available computers are:

- *cipo.grad.dcc.ufmg.br*
- *claro.grad.dcc.ufmg.br*

5 Documentation

Start your documentation with a brief description of the problem. In the following you can present your three solutions for the problem. It is important to mention in the documentation:

- The description of the problem, how did you model it and how your algorithms work. The explanation of the algorithms must be clear, you can use pseudo-code for this.

- Theoretical analysis of time and space complexity for each solution.
- Experimental analysis of run time for each solution.
- A brief conclusion of the work.
- The documentation must not exceed 8 pages.
- The documentation will be submitted via *minha.ufmg*
- The deadline: **16/11**

6 Parameters of evaluation

The evaluation parameters are divided into the following topics:

- Problem modeling for the three paradigms (35%)
- Theoretical and experimental Analysis (30%)
- Correct operation of the solutions (35%)

Good luck!