Foundations of computing: Algorithms and Data Structures

Rasmus Pagh

October 27, 2011

1 Graph-processing challenge 1

A graph is bipartite if it is possible to split the set of vertices into two parts such that all eages are between vertices in different parts. Outline a modification of the algorithm for DFS to determine if a graph is bipartite. Your algorithm should work even if the graph is not connected.

Argue for correctness of your algorithm, and state its complexity.

2 Graph modeling

For software that analyzes web sites, you want to be able to determine how many sites link to a given site `mysite.com`, how many sites that link to a site that links to `mysite.com`, and so on.
You are given a file of the relevant information, where the line:

`site1.com` $\rightarrow$ `site2.com`

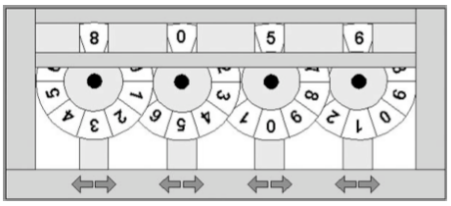
means that there is a link between the sites. Suggest a data structure for representing the link information, and an algorithm for computing the requested numbers. You should not design a new data structure and algorithm, but rely on the toolbox you have seen in the course.

3 Graph-processing challenge 2

Outline a modification of the algorithm for DFS to determine if a graph contains a cycle. Your algorithm should work even if the graph is not connected.

Argue for correctness of your algorithm, and state its complexity.

4 Graph modeling



Consider the machine to the right. Digits ranging from 0 to 9 are printed consecutively (clockwise) on the periphery of each wheel. The topmost digits of the wheels form a four-digit integer. For example, in the following figure the wheels form the integer 8056. Each wheel has two buttons associated with it. Pressing the button marked with a left arrow rotates the wheel one digit in the clockwise direction and pressing the one marked with the right arrow rotates it by one digit in the opposite direction.

We start with an initial configuration of the wheels, with the topmost digits forming the integer $S_1S_2S_3S_4$. To make it interesting, an enemy spy has modified the machine, so that in certain configurations (known to you), the machine explodes in a huge ball of fire, killing everybody around it. You will be given a list of n forbidden configurations $F_{i1}F_{i2}F_{i3}F_{i4}$ $(1 \leq i \leq n)$ and a target configuration $T_1T_2T_3T_4$. Your job is to outline an efficient algorithm to calculate the minimum number of button presses required to transform the initial configuration to the target configuration without passing through a forbidden one. What is the complexity of your algorithm in terms of the number of possible configurations?