Programming Comps — January 14-17, 2008

1 Background

You have become a world-renown scientist and have been asked to provide an independent study of the current reviewing procedures of conferences. In order to acquire data on the influence of positive personal bias in the reviewing process, you want to gather data how close authors of accepted papers are to members in the program committee for various conferences.

The basis for your assessment is provided in the DBLP data base. You find a local copy of the information collected in the data base at /u/sello/pub/dblp.xml.gz (format: bibtex in xml, see also the corresponding file dblp.dtd). See http://dblp.uni-trier.de/db/about/simpleparser/ for an algorithm (that you may use!) to parse the dblp data file. Your task is to use the dblp data to see how related program committees are to the group of authors who got their papers in.

2 Algorithm Description

You have decided to break the problem down into three different tasks:

1. You want to be able to answer queries of the form "who collaborated with a specific researcher within the past x years," whereby x and the researcher's name are parameters of the query ($x < 0$ iff there is no restriction on the collaboration time frame).

2. Next, you want to compute the collaboration distance between two given researchers. You set up an undirected graph where nodes represent researchers and an undirected arc between researchers exists if and only if they have collaborated within the past x years. The collaboration distance is given as the smallest number of edges that connects the two researchers in the graph. In case that there is no path connecting two researchers with length ten or less, you set the distance to ten.

3. For two groups of researchers, you determine the expected collaboration distance for two randomly selected members of the two groups (that is, you compute the average distance of all pairs of researchers in the two groups).

The complicated step that needs to be implemented very efficiently is task 2. You have decided to employ a modified bidirectional search. That is, you want to conduct a breadth-first search simultaneously from both researchers, stopping when the two search fringes meet, when the fringe of at least one search becomes empty, or when bidirectional search has revealed that the collaboration distance is at least ten. Do not set-up the entire graph but only those parts that you need! Also, modify the classical bidirectional search algorithm so that, in each iteration, you expand only the search fringe which is smaller.

3 Your Task

It is your task to **implement this algorithm**. Do not make up a different approach to tackle the problem! If you do, you will fail the exam. The choice of algorithm is fixed, now all that is required is that you implement it. The following features must be supported by your implementation:

- Hand in **one** file named collaboration08.tar.gz containing all files of your code, a file named Readme.txt explaining in detail how to compile it, as well as a file named Description.pdf that describes your implementation in detail, including a justification for your choice of language and the structure of your code. Do not include any copies of the DBLP data base!
- Your code must compile and run on standard department linux machines with 2GB main memory. If your program does not compile or run on a machine with those specifications, you will not pass the exam. When started, your executable will be provided with a command line argument containing the path names of the directory containing the dblp data files.
- Your executable should prompt the user with three options: List all collaborators in the past x years (ask for the value of x and the name of the researcher). Compute the collaboration distance between two researchers (ask for x and the two names). Compute the average collaboration distance between two groups of researchers (ask for x and two filenames containing the list of names in both groups).
- For the first two types of requests, give a selection of potential people in your database if the name was not unique or was only given partially. For example when the researcher's name is given as "Paul Erd" your program should find "Paul Erdös" and offer his name as an optional candidate who was meant. Similarly, the search for "tintemel" should return, among others "Ugur Cetintemel.") Associate each potential with a number and let the user clarify the selection by entering the corresponding number. For the second type of request, return the shortest collaboration path between the two researchers.
- For each request, report the CPU-time of your program. Repeated requests of all types should be possible, until the user enters "quit" which terminates the program. Make the user interface self-explanatory and efficient so that it is easy and fun to use your program. Prompt the user with a hint what is going on when conducting lengthy computations during which your program may not respond.
- Use efficient data structures. Save memory and time! In your Description.pdf, explain and justify the data structures you decided to use.

4 Evaluation Criteria

Your implementation will be evaluated based on **all** of the following criteria:

- Appropriate choice of language. You must choose a programming language that is feasible for the task. The primary focus on ease of programming and maintainability is only okay within reasonable performance limits!
- Structure, readability, and maintainability of your code.
- Correctness, memory efficiency, and speed.

It is important that your work passes a critical threshold with respect to all criteria. If you cannot complete all specifications, focus on the main algorithm and functionality that is required. A super-efficient code that is poorly documented and hardly maintainable will be regarded as a failure. So will a software engineering masterpiece that does not work as specified or that does not respond in reasonable time. Definitely do not code in assembler or shell scripts! And now: **Good luck!**