Available at the COMP3151 website.

Assignment 2 COMP[39]151 11s2 Life Story Exchange & Zabaglione

Kai Engelhardt

Revision: 1.2 of Date: 2011/10/01 02:50:50

This assignment is worth 15 marks and due **Friday October 21th, 23:59:59** local time Sydney.

1 Problem Statement

The formal part of the *dancing with the shakes* (*DWTS*) evening has ended. It is now time to exchange life stories for a few hours. Those exchanges will take place in pairs of compatible seniors, e.g., those who hadn't exchanged life stories at last month's DWTS. To organise the *life story exchange* (*LSE*), each senior will communicate with only the compatible fellows by way of a few small messages. After this negotiation phase, each senior who is still alive must announce with whom the LSE will then take place, or, if no LSE partner could be negotiated, that a period of contemplative vegetation is in order. The following correctness criteria come to mind:

1. Each senior makes precisely one announcement.
2. If a announces to engage in an LSE with b then b must announce to engage in an LSE with a and a must be compatible with b.
3. If a announces to enter a moment of contemplative vegetation, then no compatible senior will do the same.

Unfortunately, the chef at DWTS central was a little cavalier about cooling the egg yolks for the ever so popular zabaglione. Listeria have demonstrated the concept of exponential growth to those who care to learn about such things. Even more unfortunately, nobody who cares about these things was present in the DWTS kitchen. Those poor seniors who had zabaglione for dessert face an uncertain immediate future. They are prone to sudden departure due to food poisoning. All we know is that they might die between sending messages or even during the actual LSE. For simplicity, we may assume that no senior dies between exchanging a message and making an announcement.

2 Task

2.1 Algorithm Design & Programming

Model the problem using n processes for n seniors. Implement your model in C and MPI using asynchronous message passing primitives. Model deaths by those who had zabaglione as crash failures. Detect deaths using timeouts. It is advisable to have an extra process to populate data structures that are then passed on to the senior processes. Every senior process is to execute the same algorithm. When the program terminates, every senior has announced a conclusion, which is a single line stating either

- which other senior he/she is paired with, or
- that he/she is vegetating, or
- he/she has succumbed to listeria.

Announcements need to be matching those produced in C using either one of the format control strings

```
"%d exchanges life stories with %d.\n"
"%d has a seniors' moment.\n"
"%d dies blaming the zabaglione.\n"
```

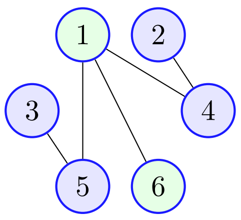
Programs will be run with two arguments

1. the filename of a file containing
   a) the number of seniors, $n < 1000$,
   b) the compatibility information matrix
   c) a list of seniors who had zabaglione, and
2. a mortality probability, a floating point number $m \in [0, 1]$.

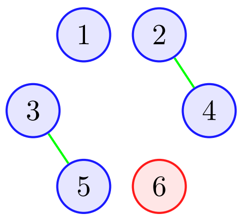
More specifically, the first line of an input file contains an integer n, the second part consists of n lines of n zeros or ones each. The i'th line contains a one (ASCII character '1') in the j'th column if, and only if, the i'th and j'th seniors are compatible. Obviously the main diagonal of the compatibilty matrix consists of zeros only and the matrix is symmetric. After the compatibility matrix, the input file contains numbers referring to the seniors who had zabaglione. Those numbers are separated by newlines or spaces. The mortality probabilty should be used to determine whether a susceptible senior will pass away at all during the simulation. If a senior has been determined to pass away then he must do so before committing to an LSE with a compatible senior. This could be done before the senior exchanges any messages. The actual death will then occur according to an even distribution before sending a message.

```
6
000111
000100
000010
110010
101101
100010
1 6
```

(a) File.



(b) As a graph.



(c) Possible solution.

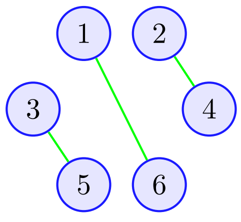
```
4 exchanges life stories with 2.
6 dies blaming the zabaglione.
3 exchanges life stories with 5.
5 exchanges life stories with 3.
1 has a seniors' moment.
2 exchanges life stories with 4.
```

(d) Output.

Figure 1: Example.

Example 1 See Fig. 1(a) for an example of an input file with 6 seniors of which number 1 and 6 had zabaglione and Fig. 1(b) for a graphical representation of that example. Fig. 1(c) shows a possible solution for the case in which senior 6 expired. Fig. 1(d) shows the announcements made.

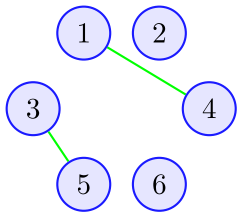
Had senior 6 survived long enough, Fig. 1(c) would not have been acceptable: Seniors 6 and 1 would have had to engage in LSE as shown in Fig. 2(a) and 2(b). In both cases, the solutions found are *optimal* in the sense that there does not exist a solution engaging more seniors in LSE given the fatalities. A *suboptimal* solution is shown in 2(c) and 2(d). This solution is still admissible because seniors 2 and 5 are incompatible.



(a) Optimal solution.

```
4 exchanges life stories with 2.
5 exchanges life stories with 3.
3 exchanges life stories with 5.
1 exchanges life stories with 6.
6 exchanges life stories with 1.
2 exchanges life stories with 4.
```

(b) Output.



(c) Suboptimal solution.

```
4 exchanges life stories with 1.
6 has a seniors' moment.
3 exchanges life stories with 5.
5 exchanges life stories with 3.
2 has a seniors' moment.
1 exchanges life stories with 4.
```

(d) Output.

Figure 2: Example continued.

The order of announcements is irrelevant. Seniors are always numbered 1 to n. Your program need not find optimal pairings only (unless you are keen on bonus marks; see below).

2.2 Analysis and Verification

Analyse, verify, and compare your solution. The analysis should cover message and time complexity of your programs. Explore how many fatalities can be tolerated. Model your algorithm in Promela. Formulate and attempt to verify relevant safety and liveness properties.

3 Evaluation Method & Marking Scheme

Your submission must compile and function on standard CSE lab machines as well as the vina cluster. The CSG has installed OpenMPI on all lab machines including the vina cluster.

The intended compilation commands are similar to:

```
make
latex '\nonstopmode\input{lse.tex}'
latex '\nonstopmode\input{lse.tex}'
```

We'll run your programs using command lines slightly more elaborate than:

```
mpirun -v -nolocal -machinefile thevinas -np 6 lse test006 0.5
```

Promela files will be evaluated with `ispin`. This assignment is worth 15 marks, which are distributed as follows:

C+MPI 5 marks

Promela 4 marks

Description 1 mark

Complexity analysis 1 mark

Verification 4 marks

3.1 Bonus Marks

5 bonus marks are on offer for programs that provably always find optimal pairings when this is theoretically possible. Note that a rigorous proof is needed to claim any bonus mark.

Warning: Due to the level of mathematical skills required to score the bonus marks, these marks are harder to get than any other 5 marks in this or previous assignments.

4 Deliverables

`lse.c` is your C+MPI solution.

`Makefile` allows me to run `make` (without any arguments) to produce an executable named `lse`, typically by invoking `mpicc`.

`lse.pml` is your Promela model, presumably with LTL formulae corresponding to relevant properties.

`lse.tex` is a LaTeX document with your names or student numbers mentioned in the `\author` command. It contains your report.

5 Submission Instructions

The `give` command to be run is:

```
% 3151
% give cs3151 ass2 lse.c Makefile lse.pml lse.tex
```

The command above submits the bare minimum. Should you feel the need to include more files, e.g., for vector diagrams or header files, just list them as well. The overall size limit is 1MB.

6 Disclaimer

1. No egg yolks were harmed in the production of this specification.
2. All characters and events in this assignment specification —even those based on real people— are entirely fictional.