Algorithms

CMPT 435

Semester Project - 100 points

Goals

- to have a semester-long programming project that you can work on a little bit each week; and [100 points]
- to develop a simulation of group/pooled testing.

Requirements and Notes

- Read Alan's article on our class web site about group/pooled testing.
- Program a simulation of the testing protocol described in Alan's article. Run your simulation on population of 1000 people (at first) in groups of 8 assuming a 2% infection rate and 100% accurate tests. Output the results in a manner similar to those shown below.

  Using the protocol described in Alan's article, there are three possibilities to consider:
  (1) there are no infected samples
  (2) there is exactly one infected sample
  (3) there are two or more infected samples

  We can determine the likelihood of each possibility based on the number of samples we pool into each group and the infection rate of the disease. For details of those percentages and how and we can derive them, read the article again.

  Program your simulation to test groups of 8 for a disease with an infection rate of 2% . For 1000 people (where 20 of them (2%) are infected and 980 are infection-free), you would make 125 groups of 8 samples each and simulate the tests. Based on the expected values (see the article) we expect the results to be as follows:

```
Case (1): 125 × 0.8500 = 106.25 instances requiring 107 tests (there are no partial tests)
Case (2): 125 × 0.1496 =  18.70 instances requiring 131 tests
Case (3): 125 × 0.0004 =   0.05 round up to 1 instance requiring 11 tests
————————————————————————————————————————————————————————————
249 tests to screen a population of 1000 people for a disease with 2% infection rate.
```

  The output of your simulation should match or be very close to those values. Once you have it working for 1000 people, continue testing with the same parameters for populations of 10,000 and 100,000 and 1M people. With populations of 1,000 to 10,000 you should get results very close to what we expect. For populations of 100,000 or 1M or more you should get results a little different from what we expect. Research the subtle difference between the *binomial distribution* and the *hypergeometric distribution* and use that knowledge to explain this difference in your documentation, along with suggestions on how to improve your simulation. Document this as well as your results for the various population sizes in LaTeX and commit that to your GitHub repository along with your code.

Resources

- Everything we're doing this semester is likely to be of use for you in this project.

Hints

Do not just infect the population and then count the cases. You must actually simulate the testing protocol and count the tests used by case along the way.

Start early. Do a little bit every day.

Submitting Your Work

Commit your final work in your private GitHub repository on or before the due date (see our syllabus).

Algorithms

CMPT 435

Questions, Answers, and Comments

Q: What data structure should I use to store each group of 8?

A: An array or a list of some sort would be good. Actually, you could hold the entire population to be tested in an array or a list.

Q: Does our program have to come up with the likelihood of each case occurring (0.85, 0.1496l, 0.0004) or can we set them as constants?

A: Your program/simulation should produce those numbers as output by counting the number of occurrences of each case. You'll know you've got it right when your output matches the statistically expected values.

Q: Should the total number of people being tested get input by the user, or set in the code?

A: To avoid complicating things with I/O, take a command-line parameter for the population size. If the program is called “sim" . . .

```
> sim 1000
```

. . . would run the simulation on 1000 people, while . . .

```
> sim 1000000
```

. . . would run the simulation on 1 million people.

Q: Will randomness be apart of our program?

A: Yes. You need to use a random() function. There is probably one built in to your programming language.

Q: How will individual positive or negative people be represented? (ex. 1's and 0's ?)

A: It's up to you, but I like 1s and 0s.

Q: What's the best way for those 1000 people to be broken up into groups of 8 and then inserted into a data structure?

A: Randomly. Kinda. It's actually not all that important. Let's say you are running the simulation on 1000 people, take the first 8 as a group, then the next 8 as a group, and so through the 125th group.

Q: If the theoretical approach to the pooled testing is assuming 100% testing accuracy, would the entire nature of the test (being binary) fall apart due to the inaccuracies of actual testing?

A: No. But a simulation that accounts for test accuracy would be more accurate than one that assumes 100% accuracy. But we've got to start somewhere. For now, assume 100% test accuracy. Once you've got that simulation working, consider taking into account the accuracy of the test as a refinement later.

Algorithms

CMPT 435

Q: I looked over the semester project assignment and your article. After reading both a few of times I feel I am missing something. I was under the impression that we would be using a binary tree to isolate the infected individuals. I feel like this is also in the "recipe for the test plan" section of the article. However, at certain points during the reading I got the impression that we should split the initial population into 125 groupings of 8. Which is it?

A: It's a subtle point that I did not make very well. We **could** build a binary tree and use its nodes as containers for the tests, as seen in the figure below. But we don't need to. I'm using a binary tree as a *conceptual representation* to explain our divide and conquer approach. If we test a group of eight and there's an infection then we'll test four and four, and when there is one or more infection(s) from those tests (and there must be) then we'll do individual tests. The process is conceptually similar a binary tree (at least at levels 0 and 1 of a binary tree). But as a matter of code, we can test `population[0]` - `population[7]` as one test, and if it's positive, use `population[0]` - `population[3]` and `population[4]` - `population[7]` for the next two tests. This illustration from the article, below, shows that, with the colored rectangles in the circles.

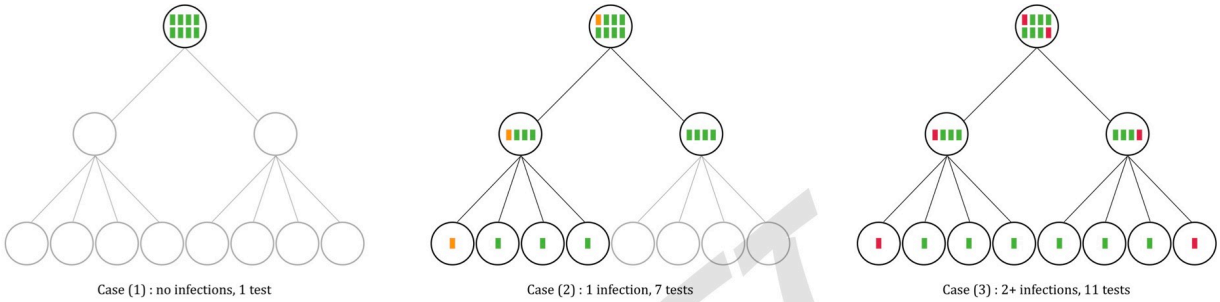

Figure 2 - Testing Possibilities - Each black circle is one test.

Q: Any idea to reduce the amount of testing while getting accurate results can be extremely beneficial in combatting this crisis. I find it interesting that a World War II era technique remains valid today. I have a question about the experiment: I understand testing randomly would be the most efficient way to test a large group but wouldn't it be better to test groups of people who live around the same area (i.e., students from the same resident hall)?

A: Excellent insight. That's exactly what we did at Marist College. We grouped dormitories into clusters (e.g., Champagnat and Midrise form one cluster, Leo and Sheahan and Marian form another) and made random selections from within those clusters.