Programming projects

Victor Eijkhout

SDS 322 2022

Introduction

1. Purpose

- Write a relatively serious C++/Fortran program
- Do a scientific simulation
- Write up your findings

2. What are we looking for?

- Pretend that this is a research article: explain the problem, how you go about solving it, what you find.
  Quality of the write-up is very important!
- Code quality: good naming, indentation, use of object-oriented techniques.

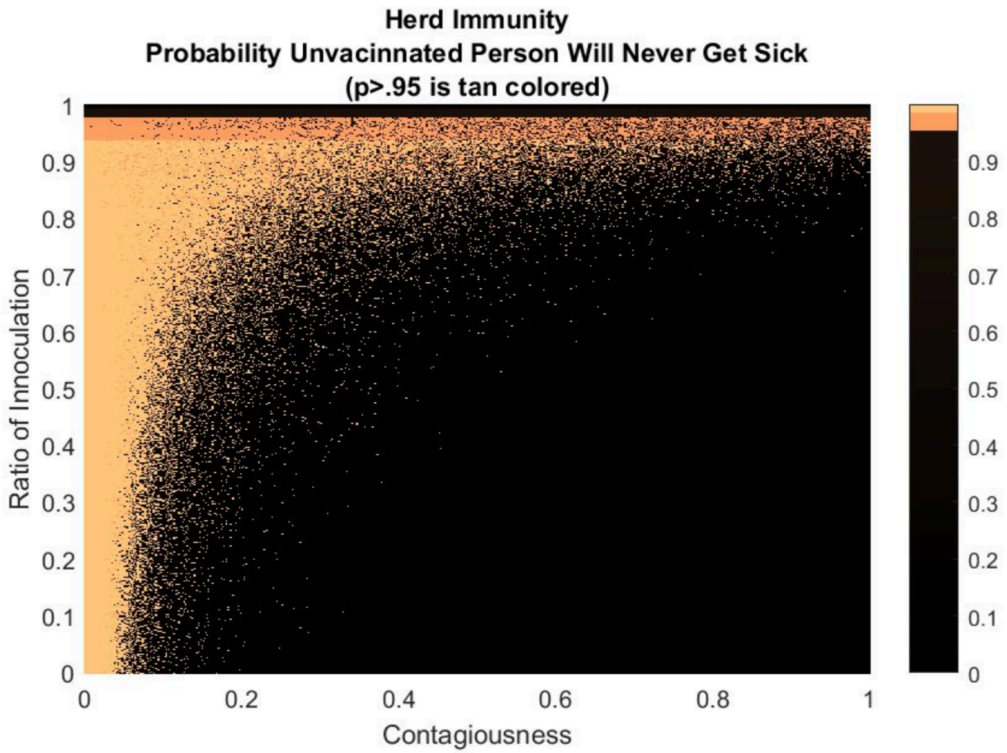
Project: infectious disease simulation

3. What is it about?

- How does an infectious disease spread through the population?
- Does anyone escape being infected? How long does the disease run?

4. What can you do

- Investigate influence of parameters: chance of transmission, incubation period, how many people are vaccinated, ...Plot one thing against another.
  Don't forget to explain what it means!
- Sample graph:



5. Level of difficulty

- If this was your first semester you can do this on your own and be proud of having done a serious simulation.

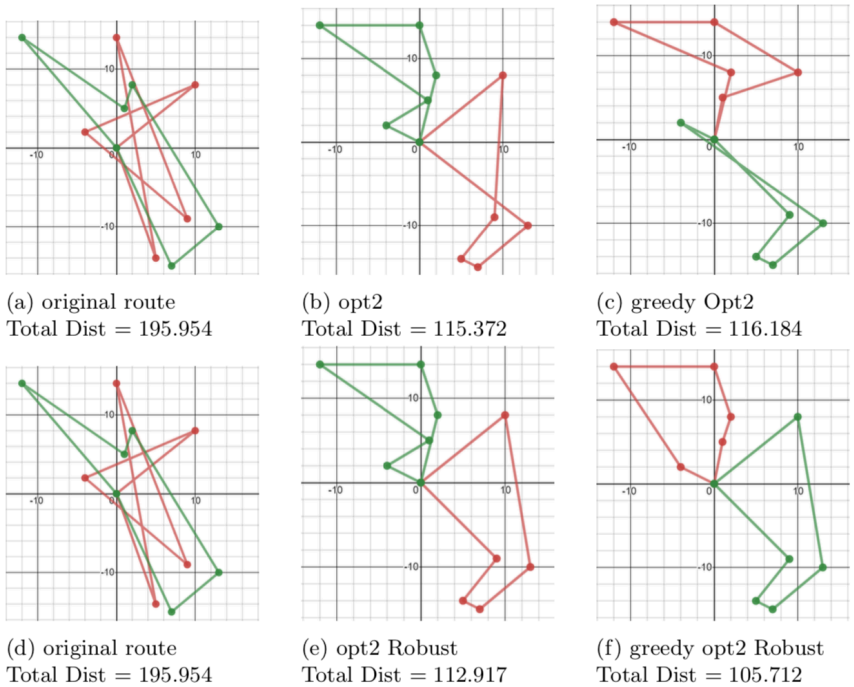
Project: Amazon delivery trucks

6. What is it about?

- How do you plan an optimal route for a delivery truck?
- How about if you have more than one truck?
- How about if you can spread over multiple days?

7. What can you do

- Investigate heuristics for route planning.
- Discuss management and ethics issues.



(a) original route
Total Dist = 195.954

(b) opt2
Total Dist = 115.372

(c) greedy Opt2
Total Dist = 116.184

(d) original route
Total Dist = 195.954

(e) opt2 Robust
Total Dist = 112.917

(f) greedy opt2 Robust
Total Dist = 105.712

- Sample graph:

8. Level of difficulty

- This is tricky coding. Sophisticated use of object-oriented techniques will be a great help.
- You can do a lot of exploring.
- This is a two-person project.

Project: redistricting

9. What is it about?

- Group census districts into congressional districts.
- Is it possible for a minority to gain the upper hand?

10. Level of difficulty

- This is *very* tricky programming.
- Even the naive approach is tricky; using dynamic programming takes it up a notch.
- If you tell you Amazon recruiter that you modeled gerrymandering by dynamic programming after your first semester they'll be impressed.

Project: Google Pagerank

11. What is it about?

- Simulate the internet
- Which web pages are important?

12. What can you do

- Different techniques for modeling the problem.
- Dig into the mathematics of it: relation between graphs and sparse matrices.

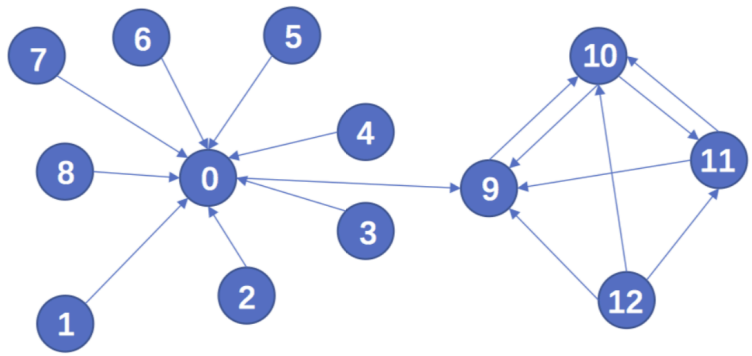


Figure 6: Web with one page artificially made 'important'

- Sample graph:

13. Level of difficulty

- User of pointers in the simple model.
- Opportunity for thinking about computational efficiency.

Project: High performance linear algebra

14. What is it about?

- In linear algebra the naive algorithms are often not efficient.
- Explore algorithms that take architecture into account.

15. What can you do

- Learn about architecture,
- ... and how to code for it.

16. Level of difficulty

- You get to use C++20 features. This is serous geekery.
- If you think programming correctly is hard, try programming efficiently ...

Project: Climate change (Fortran only)

17. What is it about?

- 'Climate has always changed, always will'.
- Make that statement testable, and test it with real data.

18. What can you do

- Handle multiple datasets through array notation.
- Explore numerical algorithms.

19. Level of difficulty

- Not too hard. Single person project.

Project: desk calculator

20. What is it about

- Interpreter for complicated expressions
- Translation between ‘infix’ and ‘postfix’

This can not be chosen if we already did parts in class.