Lab 6 - Software Projects, Workflows, and Object-Oriented Design

NERS/ENGR 570 Fall 2020
Oct. 9th, 2020

This lab is an ongoing experiment and will be graded primarily on participation, although there are a few deliverables, and this activity feeds into HW 2.

The class will be working together collaboratively on the following github project.
https://github.com/bkochuna/ners570f20-Lab06

Be sure to consult the following reading material for references with this lab.

- UML Distilled
- Design Patterns (or this Design Patterns)
- Code Complete

Problem Statement Definition
Develop a library to perform sparse matrix-vector multiply (SpMV) on single and double precision data types.

Some high level requirements about the project:

- The library shall be flexible in providing sparse matrices in several formats.
- The library shall make use of Object-Oriented Design (and Programming)
- The library shall make use of Design Patterns
- The library shall be extensible
- The library shall provide an example of to interface with a third-party library
- The library shall use any of C, C++, or Fortran only.

Some detailed functional requirements

- The library shall provide an interface to perform sparse matrix vector multiplication.
- The library shall support calculations in single precision arithmetic.
- The library shall support calculations in double precision arithmetic.
- The library shall provide an interface to return or define a matrix in COO Format
- The library shall provide an interface to return or define a matrix in CSR Format
- The library shall provide an interface to return or define a matrix in CSC Format
- The library shall provide an interface to return or define a matrix in ELLPACK Format

Approach/Format
We will use different break out rooms for the different teams. Each group should expect to interface with the other groups to complete their task.

Suggested approach:

- For each of the things that follow create some issues for each issue form into at least pairs if not larger groups
- (10 min) Start with creating milestones for high level groups of tasks. Some examples
    - Planning, infrastructure, design, Release X.
- Figure out what process you want to use. Some suggested approaches
    - Minimize external tools
    - Time box your activities
    - Agree on intervals for checking in
    - Does the developer write the test or someone else?
- Start with the design--this will help you to understand the problem better
    - If something is unclear check the requirements
    - Development of an architecture
        - The product of this task will be a UML package diagram
        - https://www.safaribooksonline.com/library/view/code-complete-second/0735619670/ch03s05.html
    - Development of a high level design (LOOK FOR PATTERNS!)
        - The product of this task will be UML diagrams. Specifically: class, sequence, state, and communication diagrams
    - Development of a low level design
        - The product of this task will be
            - identification of appropriate algorithms for performing the various computations
            - Identification and justification for choice of concrete data types (e.g. trees, stacks, queues, lists, arrays, maps)
            - identification and justification of a suitable programming language
            - https://www.safaribooksonline.com/library/view/code-complete-second/0735619670/ch07s02.html
- Development of a test plan and tests
    - The product of this task will be a document that defines a test number related to each identified feature of the code.
    - https://www.safaribooksonline.com/library/view/code-complete-second/0735619670/ch22.html
- Try to figure out how to scope work so that a lot of it can be done in parallel by the team.

For overall comments about design:
https://www.safaribooksonline.com/library/view/code-complete-second/0735619670/ch05.html
https://www.safaribooksonline.com/library/view/code-complete-second/0735619670/ch05s02.html
https://www.safaribooksonline.com/library/view/code-complete-second/0735619670/ch05s03.html

At the end of the lab the expectation is that there will be a solid “plan” for implementing the library. The last 15 minutes of the lab we will reconvene to discuss the progress.

Note about UML diagrams
Some recommended UML tools are UMLetino (http://www.umletino.com/umletino.html) LucidChart (https://www.lucidchart.com/pages/examples/uml_diagram_tool), Astah (http://astah.net/), however you can use google slides, powerpoint, visio, Tikz (LaTeX), PlantUML, yUML, or any other UML tool or drawing tool you wish.

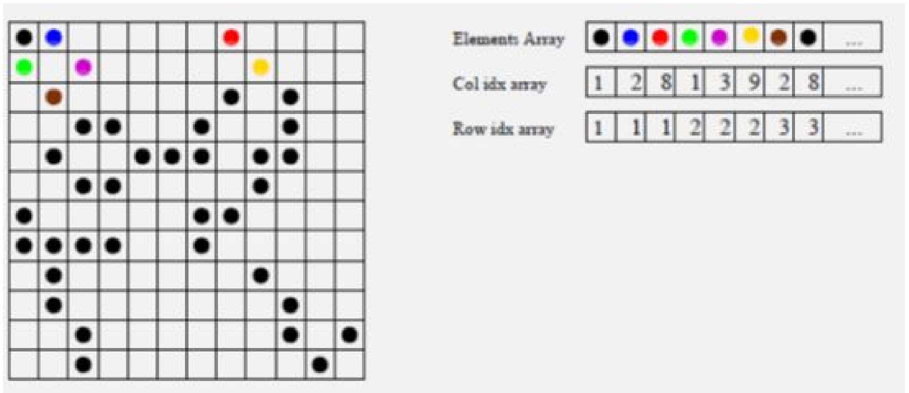
For expediency during class, *it is totally appropriate to sketch things on paper or the white board, or Google Jamboard or something and convert to a more formal picture later.*

Deliverables
The deliverable for this exercise is to provide a document describing what you did during lab include the resulting work product(s) or references to your issues/cards. In this document you will also include an essay response describing:

1. What did you learn by participating in your activities?
2. What approach did the group try to accomplish their objective?
3. What things worked well about this approach?
4. What things did not work well about this approach?
5. What would you suggest be done to improve this process?

APPENDIX A -- Sparse Matrix Storage Formats

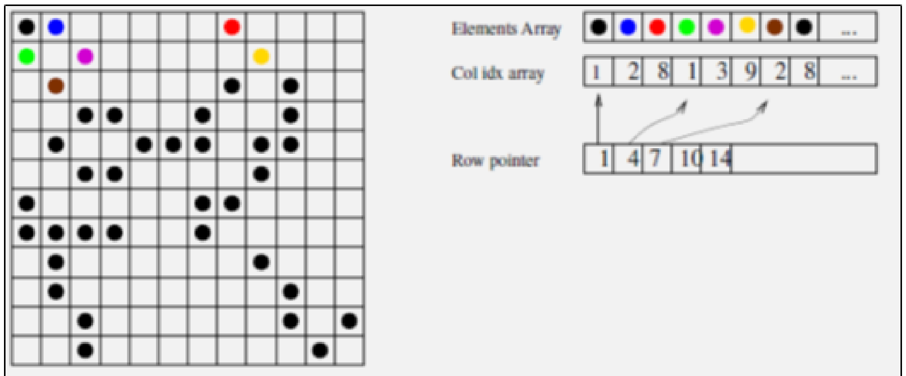
<u>COO Format</u>



```
do  i=1,nz
    ir = ia(i)
    jc = ja(i)
    y(ir) = y(ir) + as(i)*x(jc)
enddo
```

Cost: 5 memory reads, 1 write and 2 flops per iteration.

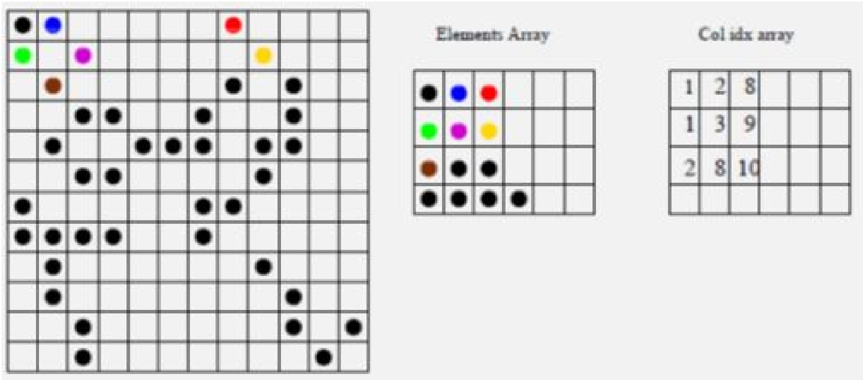
<u>CSR Format</u>



```
do  i=1,m
   do  j=ia(i), ia(i+1)-1
       y(i) = y(i) + as(j)*x(ja(j))
   enddo
enddo
```

Cost: 3 memory read and 1 write per outer iteration, 3 memory read and 2 flops per inner iteration.

ELLPACK Format



```
do  i=1,m
   do  j=1,  maxnz
      y(i) = y(i) +   as(i,j)*x(ja(i,j))
   enddo
enddo
```

Cost: 1 memory read and 1 write per outer iteration, 3 memory read and 2 flops per inner iteration (also, regular access pattern).

APPENDIX B -- Requirements Checklist

Requirements Checklist

The requirements checklist contains a list of questions to ask yourself about your project's requirements. This book doesn't tell you how to do good requirements development, and the list won't tell you how to do one either. Use the list as a sanity check at construction time to determine how solid the ground that you're standing on is—where you are on the requirements Richter scale.

Not all of the checklist questions will apply to your project. If you're working on an informal project, you'll find some that you don't even need to think about. You'll find others that you need to think about but don't need to answer formally. If you're working on a large, formal project, however, you may need to consider every one.

Specific Functional Requirements

- Are all the inputs to the system specified, including their source, accuracy, range of values, and frequency?
- Are all the outputs from the system specified, including their destination, accuracy, range of values, frequency, and format?
- Are all output formats specified for Web pages, reports, and so on?
- Are all the external hardware and software interfaces specified?
- Are all the external communication interfaces specified, including handshaking, error-checking, and communication protocols?
- Are all the tasks the user wants to perform specified?
- Is the data used in each task and the data resulting from each task specified?

Specific Nonfunctional (Quality) Requirements

- Is the expected response time, from the user's point of view, specified for all necessary operations?
- Are other timing considerations specified, such as processing time, data transfer rate, and system throughput?
- Is the level of security specified?
- Is the reliability specified, including the consequences of software failure, the vital information that needs to be protected from failure, and the strategy for error detection and recovery?
- Are minimum machine memory and free disk space specified?

- Is the maintainability of the system specified, including its ability to adapt to changes in specific functionality, changes in the operating environment, and changes in its interfaces with other software?
- Is the definition of success included? Of failure?

Requirements Quality

- Are the requirements written in the user's language? Do the users think so?
- Does each requirement avoid conflicts with other requirements?
- Are acceptable tradeoffs between competing attributes specified—for example, between robustness and correctness?
- Do the requirements avoid specifying the design?
- Are the requirements at a fairly consistent level of detail? Should any requirement be specified in more detail? Should any requirement be specified in less detail?
- Are the requirements clear enough to be turned over to an independent group for construction and still be understood? Do the developers think so?
- Is each item relevant to the problem and its solution? Can each item be traced to its origin in the problem environment?
- Is each requirement testable? Will it be possible for independent testing to determine whether each requirement has been satisfied?
- Are all possible changes to the requirements specified, including the likelihood of each change?

Requirements Completeness

- Where information isn't available before development begins, are the areas of incompleteness specified?
- Are the requirements complete in the sense that if the product satisfies every requirement, it will be acceptable?
- Are you comfortable with all the requirements? Have you eliminated requirements that are impossible to implement and included just to appease your customer or your boss?