A Sample Project Report [*]

J. D. Teresco[†]and O.-W. Kenobi
Department of Computer Science
Siena College
Loudonville, NY 12211

Abstract

It's usually a good idea to include an abstract. This should be a brief summary of your paper. Highlight your important results. A half a page is about right. Maybe less. It should be independent of your subsequent introduction/overview section.

1 Overview

Begin with an introduction to your topic, specifically emphasizing what you have done. Some historical background and references to related projects are appropriate. You should also outline the paper in the overview section. It might go something like this:

In Section 2 we discuss how to build a vector from an array. Section 4 has a lot of examples of things you might want LaTeX [6] to do. In Section 3, we discuss how writing is hard and offer some suggestions on how to do a better job at it. Section 5 describes how to use LaTeX to compile this document into printable postscript. Finally, Section 6 summarizes the results and presents suggestions for future work.

Notice how references are used instead of actual section numbers. This allows you to move, add, or remove sections and have the section numbers stay correct. The names are specified below with label commands.

[*]This work was completed in partial fulfillment of the final project requirement for Computer Science 400 at Siena College, Fall 2017.

[†]Corresponding author, e-mail: jteresco@siena.edu

2 Building a Vector from an Array

This section is here to give examples of math mode, font changes, the verbatim environment, and importing a postscript figure.

This experiment measures the time taken to copy an array of n `int` values to a `Vector` of n `Integer` objects. We will use the `add()` method of `class Vector` to append new values to the end of the vector. Accessing each item in the array is a constant time ($O(1)$) operation, as are the construction of the `Integer` object which will hold the value in the vector and each `add()` call. We do this for each of n items, thus, we expect this operation to take $O(n)$ time. We create our vector to be initially of size n to avoid the need to resize dynamically during the experiments. We use the `Vector` implementation found in the `structure` package.

Now, we describe how the experiment is set up, how you run it, and present and discuss the results. This sentence is mainly inserted to show an example of italics.

A simple Java program was written to copy of an array into the vector. A method called `copy()` takes the value of n as its parameter, creates and fills an array of size n, creates an empty `Vector`, starts a timer, copies the values from the array to the vector, stops the timer, and prints out the time taken. The body of this method is shown in Figure 1.

```
int[] array = new int[n];
int i;
for (i=0; i<n; i++) array[i]=i;

Vector v = new Vector(n);
long start=System.currentTimeMillis();
for (i=0; i<n; i++) {
    v.add(new Integer(array[i]));
}
long end = System.currentTimeMillis();
System.out.println("For n="+n+", time="+(end-start));
```

Figure 1: Java source to copy an array to a Vector.

Your figures should have a caption that describes exactly what is in the figure. A discussion of the figure should be included in your main text, referring to the figure by number, like so:

The method in Figure 1 is executed 10 times for each value of n, and the minimum time is taken.

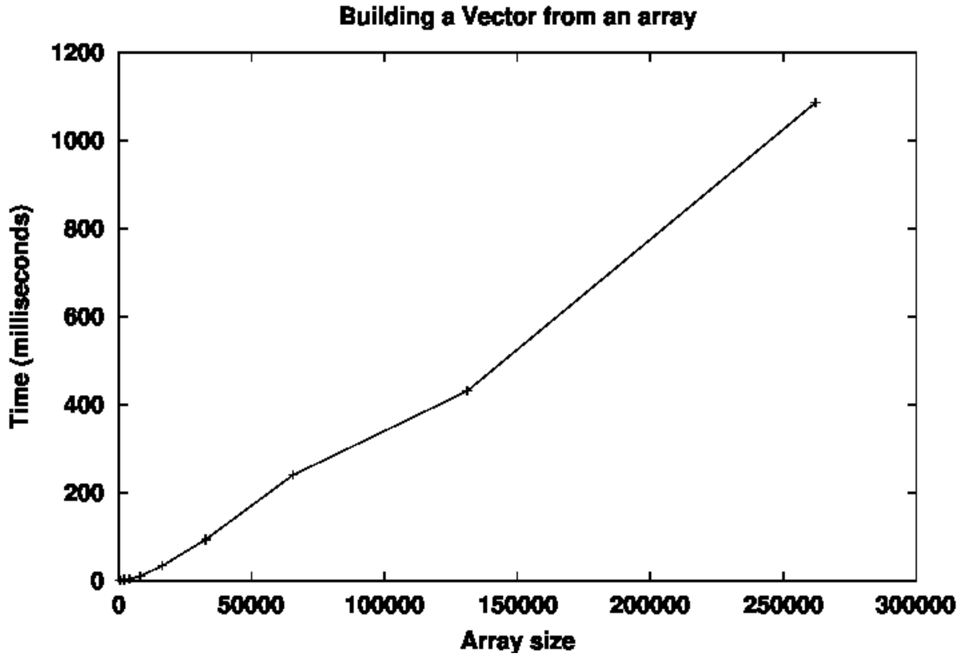

Figure 2: Times in milliseconds to build a Vector from various array sizes. using the code in Figure 1

The runs are all done on a PC with a 700 MHz Intel Pentium III processor, running FreeBSD 4.3 and the Java Development Kit version 1.3.0. We choose array and vector sizes starting at $n = 512$ and double them until our final size $n = 262,144$. These bounds, all powers of 2, were chosen because the 512-element case is the first which takes at least one millisecond, and the 262,144-element is the largest which can execute without generating an out of memory exception from the Java run-time environment. The graph in Figure 2 shows the actual execution times for this method. These times grow linearly with the size of the array. This agrees with the expected $O(n)$ behavior.

3 Writing is Hard

As you undoubtedly know, writing is hard. Turns out, technical writing is extra hard. In addition to all of the good things we should do with spelling, grammar, punctuation, and everything else, we need to focus on writing for a technical audience. There is no room for a wordy writing style. Our goals are to be concise and precise. If you can say it in one short sentence instead of a rambling paragraph, do it!

Here's an example of a before and after. Before editing:

The experiment was actually set up first. After we set up the experiment, we ran the experiment on all of our test cases. These test cases are very interesting, as we carefully selected them to be just that. Running the experiment allowed us to gather the timings we present below. The timings show a trend that looks like a linear relationship between the input parameter and time taken. This matches the theoretical expectation, which is good.

After:

We set up and run our experiment on a set of carefully selected test cases. Results, shown below, suggest a linear relationship between the input parameter and time taken, matching theoretical expectations.

It is perfectly reasonable to write the first paragraph initially. However, subsequent editing, ideally by multiple people, should lead to the much more concise and precise second paragraph.

Technical writing is a learned skill, so do not be surprised if your early versions come under heavy editing and constructive criticism. That said, even the most experienced technical writers need to make multiple editing passes.

Here are a few more tips to keep in mind.

- Define all terms, abbreviations, and acronyms the first time you use them. You should not assume your reader is aware of them, and it is annoying, as a reader, to feel the need to stop reading to look up a term or acronym.
- Avoid contractions.
- Carefully consider the voice to use and be consistent. Descriptions of work done is often done using "we" even in the case of a single author. Avoid using too much of a storytelling approach.

4 Subsections, Math, Tables, Citations

Now back to more of what LaTeX can do and how to present some common components of a technical paper. I have subsections here. I can label those and refer to them as Sections 4.1

and 4.2.

4.1 Part 1

Subsections are fun. So is math mode: $f(x) = x^4 + a_0 + x^{2k}$ is nice, but how about $\frac{\alpha}{2} + \log x^y$? Tip: see the file `isoent-ref.pdf` for 30 pages of special characters and how to make LaTeXprint them.

4.2 Part 2

Did you like the equations in Section 4.1? How about a table?

Architecture	**IPC vehicle**	**Procs.**	**Computers**	**Key**
IBM SP2	switch	2–32	2–32	SP2s
IBM SP2	Ethernet	2–32	2–32	SP2e
Sun Ultra 2/2200	shared memory	1–2	1	SUNsh
Sun Ultra 2/2200	local p4	1–2	1	SUNlp4
Sun Ultra 2/2200	Ethernet p4	1–6	1–3	SUNp4(10)
Sun Ultra 2/2200	fast Ethernet p4	1–6	1–3	SUNp4(100)
SGI Onyx II	shared memory	1–8	1	SGIsh
SGI Onyx II	local p4	1–8	1	SGIlp4

Table 1: Parallel environments for network performance comparisons. Variations in architecture, interprocess communication vehicles, numbers of processes and computers are examined.

I made Table 1. Isn't it nice? It also has a caption that describes exactly what the table is supposed to show. This is an example of how figures and tables can float around. This is usually a good thing, and LaTeXwill do something reasonable. However, it is another reason that captions must describe figures and tables well, and that the text should refer to them as "Table X" rather than "the following table."

It is essential that you include all appropriate references. Maybe you used something by Barnard and Simon [2]. Or perhaps you want to say that several papers discuss a point you are making in more detail [3, 5]. I'll include a few more so you can see examples of more types of citations [1, 4, 7]. Generally, it is better to cite a published work rather than a URL. If you use information from a web page, try to find books, journal articles, conference proceedings, or technical reports that contain the same information. When citing web pages, try to include an accurate title and author. Tip: open the bibtex file in emacs, and the "Entry-Types" menu will give you a bunch of other templates you can use.

Lists of items are also fun. Maybe you want bullets:

- Item 1.
- Item 2.
  - subitem of 2!
  - another subitem of 2.

Maybe you want them numbered:

1. I bet this gets 1.
2. And this gets 2.

Hey! This paragraph isn't indented. Sometimes that is what you want. And you can do it.

5 Compiling

To compile this file into a device independent (`.dvi`) file:

```
latex paper.tex
```

Most of the time, you want PDF:

```
pdflatex paper.tex
```

The procedure above will not include any of your citations. To get those, also run

```
bibtex paper
```

after you have run latex once, then rerun the latex command, at least twice. Since latex is a one-pass compiler, it relies on information it stores from previous runs to generate labels and citations and other information. If things seem weird, just rerun latex twice and most of the time, things will be fine.

The included Makefile should do all of this automatically, as long as you use GNU make (`gmake`).

6 Conclusions

Summary, conclusions, discussion of the results, and ideas for future work should be here.

Again, remember that writing is hard and writing well is very hard. It requires a long cycle of writing, reading, editing, rearranging, rewriting, and so on. If you are working in a group, have all group members read and edit each others' sections. Scientific writing should be clear and concise. If find a paragraph that says what can be said in a sentence, cut it down to one sentence!

Acknowledgments

The authors would like to thank all of our system administrators, past and present, for installing the tools we needed on our systems and for keeping those systems working smoothly while we completed this project.

References

[1] M. Adler and J. Gailly. Gzip home page, http://www.gzip.org. URL, 1999.

[2] S. T. Barnard and H. D. Simon. Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency: Practice and Experience*, 6(2):101–117, 1994.

[3] M. W. Beall and R. M. O'Bara. *SCOREC Mesh Database Version 4.2 Users Guide*. Scientific Computational Research Center, Rensselaer Polytechnic Institute, Troy, NY, 1998.

[4] J. J. Binks, 1999. *Personal communication*.

[5] A. E. Einstein. *Some Really Good Stuff*. PhD thesis, Einstein Institute of Physics, 1920.

[6] L. Lamport. *LaTeX : A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, 1986.

[7] H. Sagan. *Space-Filling Curves*. Springer-Verlag, 1994.