Dissertation Type: Research

DEPARTMENT OF COMPUTER SCIENCE

Dictionary Matching with Fingerprints

Dominic Joseph Moylett

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Engineering in the Faculty of Engineering.

Tuesday 14th April, 2015

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of MEng in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Dominic Joseph Moylett, Tuesday 14$^{\text{th}}$ April, 2015

Contents

List of Figures

List of Tables

List of Algorithms

List of Listings

Executive Summary

A compulsory section, of at most 1 page

This section should précis the project context, aims and objectives, and main contributions and achievements; the same section may be called an abstract elsewhere. The goal is to ensure the reader is clear about what the topic is, what you have done within this topic, *and* what your view of the outcome is.

The former aspects should be guided by your specification: essentially this section is a (very) short version of what is typically the first chapter. The latter aspects should be presented as a concise, factual bullet point list. The points will of course differ for each project, but an example is as follows:

- I spent 120 hours collecting material on and learning about the Java garbage-collection sub-system.
- I wrote a total of 5000 lines of source code, comprising a Linux device driver for a robot (in C) and a GUI (in Java) that is used to control it.
- I designed a new algorithm for computing the non-linear mapping from A-space to B-space using a genetic algorithm, see page 17.
- I implemented a version of the algorithm proposed by Jones and Smith in [6], see page 12, corrected a mistake in it, and compared the results with several alternatives.

Supporting Technologies

- I used the GNU Multiple Precision Arithmetic Library (GMP) to support my implementation of Karp-Rabin fingerprints.
- I used the C Minimum Perfect Hashing Library (CMPH) for static perfect hashing.
- I used an open-source implementation of Red-Black Trees from http://en.literateprograms.org/Red-black_tree_(C)?oldid=19567, with some minor adaptations.

Notation and Acronyms

CMPH	:	C Minimum Perfect Hashing Library
GMP	:	GNU Multiple Precision Arithmetic Library
T	:	A text string of n characters
t_i	:	The i-th character in T
$\mathcal{P}$	:	A list of k patterns
P_i	:	The i-th pattern in $\mathcal{P}$, a text string of m characters
$p_{i,j}$	:	The j-th character in P_i
$\phi(T)$	:	The Karp-Rabin fingerprint of a text string T

Acknowledgements

First and foremost, I would like to thank my supervisors: Dr. Raphaël Clifford and Dr. Benjamin Sach. This project would have been impossible without their work and advice. Alongside them, I would like to mention Dr. Markus Jalsenius for his assistance during the summer project that led to this work and Dr. Allyx Fontaine, who contributed to the paper on which my project is based and advised me alongside Benjamin every week.

Everyone on my course has had an impact on me over the past four years. In particular, I would like to mention William Coaluca, Stephen de Mora, Nicholas Phillips, James Savage and Ashley Whetter. I have put countless hours into many projects with one or more of them.

I would like to acknowledge David Beddows, Derek Bekoe, Timothy Lewis and Jonathan Walsh for remaining a stable household for the past three years - four in the case of David and Timothy.

Last, but most certainly not least, I would like to thank my family for the infinite support, happiness and love they have given me my entire life.

Chapter 1

Contextual Background

A compulsory chapter, of roughly 10 pages

This chapter should describe the project context, and motivate each of the proposed aims and objectives. Ideally, it is written at a fairly high-level, and easily understood by a reader who is technically competent but not an expert in the topic itself.

In short, the goal is to answer three questions for the reader. First, what is the project topic, or problem being investigated? Second, why is the topic important, or rather why should the reader care about it? For example, why there is a need for this project (e.g., lack of similar software or deficiency in existing software), who will benefit from the project and in what way (e.g., end-users, or software developers) what work does the project build on and why is the selected approach either important and/or interesting (e.g., fills a gap in literature, applies results from another field to a new problem). Finally, what are the central challenges involved and why are they significant?

The chapter should conclude with a concise bullet point list that summarises the aims and objectives. For example:

> The high-level objective of this project is to reduce the performance gap between hardware and software implementations of modular arithmetic. More specifically, the concrete aims are:
>
> 1. Research and survey literature on public-key cryptography and identify the state of the art in exponentiation algorithms.
> 2. Improve the state of the art algorithm so that it can be used in an effective and flexible way on constrained devices.
> 3. Implement a framework for describing exponentiation algorithms and populate it with suitable examples from the literature on an ARM7 platform.
> 4. Use the framework to perform a study of algorithm performance in terms of time and space, and show the proposed improvements are worthwhile.

Chapter 2

Technical Background

2.1 Pattern Matching: Formal Definitions

Pattern matching with a single pattern is a simple problem to describe intuitively: We have a text and a pattern, and we want to output any indexes where the pattern occurs in the text.

More formally, we refer to the text by T, and define it as a string of n characters $t_0...t_{n-1}$. Likewise, the pattern is referred to as P, and is a string of m characters $p_0...p_{m-1}$. The aim of the text indexing problem is to output indexes $i \in \{m-1, ..., n-1\}$ such that $t_{i-m+1}...t_i = P$.

It is worth noting that there are many other ways of defining this problem. The most notable differences in this paper are that the text and pattern are indexed at zero instead of one, and that the index at the end of the pattern's occurance is returned instead of the index at the start. Both of these are done to be intentionally to be consistent with the code implemented: The zero-indexing is because the implementations are written in C, which also uses zero indexing, and reporting the index at the end of the occurance is to cater for the streaming model, on which more information will be provided later.

2.1.1 Dictionary Matching: Formal Definitions

Like pattern matching, dictionary matching is also simple to describe intuitively: We have one text as before, but now we have multiple patterns, and we want to output any indexes where a pattern occurs in the text.

Formally, this is defined as follows: We have a text n characters long $T = t_0...t_{n-1}$, and a set of k patterns $\mathcal{P} = \{P_0, ..., P_k\}$ of respective lengths $M = \{m_0, ..., m_k\}$. Hence a given pattern P_i is a string of m_i characters $p_{i,0}...pi, m_i - 1$. We output an index $j \in \{\min(M), ..., n_1\}$ if $\exists i \in \{0, ..., k-1\}$ such that $t_{j-m_i+1}...t_j = P_i$.

Note that for this work, we do not care about what pattern has occured in the text, only that a pattern has occured. This is due to a limitation with one of the algorithms, which will be discussed later.

2.2 The Streaming Model

Data streaming is a way of reducing space consumption for certain problems. Under this model, required space is reduced by not processing the entire problem input at once. Instead, the input is provided to the algorithm in portions, delivered via a stream of data. The algorithm processes one portion of the input at a time, and it is required that the algorithm is not allowed to store the entire input.

Under this model, we measure performance by two properties:

- **Space:** The size of the data structure
- **Time:** The time taken to process each portion in the stream

It is easy to see how pattern matching and in turn dictionary matching can be performed in this model. We can process the text by individual characters. During preprocessing we store the pattern and initialise a circular buffer buf which is m characters long. At index j when we receive character t_j we perform the algorithm described in Algorithm 2.1. A dictionary matching variant can be done by storing a circular buffer which is $\max(M)$ characters long and repeating Algorithm 2.1 k times. These algorithms use $O(m)$ and $O(\sum_{i=0}^{k-1} m_i)$ respectively, both in terms of space and time per character.

```
rotate buf by one
append t_i to buf for i = 0 upto m − 1 do
    if buf_i ≠ p_i then
        return -1
    end
end
return j
```

Algorithm 2.1: A naïve solution to single pattern matching.

```
newstate ← −1
for i = 0 upto k − 1 do
    state ← −1
    j ← 0
    while goto(state, p_{i,j}) ≠ fail do
        state ← goto(state, p_{i,j})
        j ← j + 1
    end
    while j < m_i do
        newstate ← newstate + 1
        goto(state, p_{i,j}) ← newstate
        state ← newstate
        j ← j + 1
    end
    output(state) = {P_i}
end
forall the a ∈ Σ such that goto(−1, a) = fail do
    goto(−1, a) = −1
end
```

Algorithm 2.2: Constructing the goto function for Aho-Corasick.

Of course, these are poor solutions to both pattern and dictionary matching. We can do much better in terms of both time and space complexity.

2.3 The Aho-Corasick Algorithm for Dictionary Matching

The Aho-Corasick Algorithm for Efficient String Matching[1] - known hereafter as Aho-Corasick - is a deterministic algorithm for dictionary matching. Published in 1975, the algorithm works as a generalisation of Knuth-Morris-Pratt, extending the state machine from single patterns in KMP to multiple patterns.

Preprocessing consists of three algorithms. The first, Algorithm 2.2, produces the goto function, which determines what to do if the next character in the stream is a match. This in essence works by building a suffix tree: We traverse the tree until we either reach the end of the pattern or we hit a leaf, and then append the rest of the pattern to the leaf. Note that Σ refers to the alphabet of the patterns and $fail$ is a default fail state for if the goto function cannot find a character for that state.

The second, Algorithm 2.3 constructs the failure function for when the next character cannot be found and the output function for whether or not there is a match. This is similar to how the failure table is computed in Knuth-Morris-Pratt, by using previously computed failure tables to find the longest prefix that is also a suffix of that point in the pattern.

From these two algorithms alone it is possible to perform dictionary matching, using a computation method again similar to Knuth-Morris-Pratt: For each character t_j in the text when we are in state s, we check if $\text{goto}(s, t_j) = fail$. If that is the case, we call $s \leftarrow \text{failure}(s)$ repeatedly until the previous check no longer holds. We then update our state $s \leftarrow \text{goto}(s, t_j)$, and if $\text{output}(s) \neq \mathit{empty}$ then we return j, otherwise we return -1. This runs in amortised $O(|\Sigma|)$ time per character, and worst case $O(|\Sigma| \max(M))$ time per character, as can be seen via Knuth-Morris-Pratt arguments.

To improve on this running time, Algorithm 2.4 is used, which combines the goto and failure functions to produce a next function, which given any state and character returns the next state. Computation now simply becomes as each character t_j comes in when we are in state s, call $s \leftarrow \text{next}(s, t_j)$ and return j if $\text{output}(s) \neq \mathit{empty}$. This runs in worst case $O(|\Sigma|)$ time per character, where the bottleneck is finding

```
queue ← empty
foreach a ∈ Σ such that goto(−1, a) = s ≠ −1 do
    queue ← queue ∪ {s}
    failure(s) ← −1
end
while queue ≠ empty do
    r ← pop(queue)
    foreach a ∈ Σ such that goto(r, a) = s ≠ fail do
        queue ← queue ∪ s
        state ← failure(r)
        while goto(state, a) = fail do
            state ← failure(state)
        end
        failure(s) ← goto(state, a)
        output(s) ← output(s) ∪ output(failure(s))
    end
end
```

Algorithm 2.3: Constructing the failure and output functions for Aho-Corasick.

```
queue ← empty
foreach a ∈ Σ do
    next(−1, a) = goto(−1, a)
    if goto(−1, a) ≠ −1 then
        queue ← queue ∪ {goto(−1, a)}
    end
end
while queue ≠ empty do
    r ← pop(queue)
    foreach a ∈ Σ do
        if goto(r, a) = s ≠ fail then
            queue ← queue ∪ s
            next(r, a) = s
        end
        else
            next(r, a) = next(failure(r), a)
        end
    end
end
```

Algorithm 2.4: Constructing the next function for Aho-Corasick.

the value associated with character t_j in the next function. In both the case with the goto and failure functions and the case with only the next function, space complexity is $O(\sum_{i=0}^{k-1} m_i)$.

Chapter 3

Project Execution

A topic-specific chapter, of roughly 20 pages

This chapter is intended to describe what you did: the goal is to explain the main activity or activities, of any type, which constituted your work during the project. The content is highly topic-specific, but for many projects it will make sense to split the chapter into two sections: one will discuss the design of something (e.g., some hardware or software, or an algorithm, or experiment), including any rationale or decisions made, and the other will discuss how this design was realised via some form of implementation.

This is, of course, far from ideal for *many* project topics. Some situations which clearly require a different approach include:

- In a project where asymptotic analysis of some algorithm is the goal, there is no real "design and implementation" in a traditional sense even though the activity of analysis is clearly within the remit of this chapter.
- In a project where analysis of some results is as major, or a more major goal than the implementation that produced them, it might be sensible to merge this chapter with the next one: the main activity is such that discussion of the results cannot be viewed separately.

Note that it is common to include evidence of "best practice" project management (e.g., use of version control, choice of programming language and so on). Rather than simply a rote list, make sure any such content is useful and/or informative in some way: for example, if there was a decision to be made then explain the trade-offs and implications involved.

3.1 Example Section

This is an example section; the following content is auto-generated dummy text. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit.

foo

Figure 3.1: This is an example figure.

foo	bar	baz
0	0	0
1	1	1
⋮	⋮	⋮
9	9	9

Table 3.1: This is an example table.

Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

3.1.1 Example Sub-section

This is an example sub-section; the following content is auto-generated dummy text. Notice the examples in Figure 3.1, Table 3.1, Algorithm 3.1 and Listing 3.1. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan

for $i = 0$ **upto** n **do**
  $t_i \leftarrow 0$
end

Algorithm 3.1: This is an example algorithm.

```
for( i = 0; i < n; i++ ) {
  t[ i ] = 0;
}
```

Listing 3.1: This is an example listing.

bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Example Sub-sub-section

This is an example sub-sub-section; the following content is auto-generated dummy text. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Example paragraph. This is an example paragraph; note the trailing full-stop in the title, which is intended to ensure it does not run into the text.

Chapter 4

Critical Evaluation

A topic-specific chapter, of roughly 10 pages

This chapter is intended to evaluate what you did. The content is highly topic-specific, but for many projects will have flavours of the following:

1. functional testing, including analysis and explanation of failure cases,
2. behavioural testing, often including analysis of any results that draw some form of conclusion wrt. the aims and objectives, and
3. evaluation of options and decisions within the project, and/or a comparison with alternatives.

This chapter often acts to differentiate project quality: even if the work completed is of a high technical quality, critical yet objective evaluation and comparison of the outcomes is crucial. In essence, the reader wants to learn something, so the worst examples amount to simple statements of fact (e.g., "graph X shows the result is Y"); the best examples are analytical and exploratory (e.g., "graph X shows the result is Y, which means Z; this contradicts [1], which may be because I use a different assumption"). As such, both positive *and* negative outcomes are valid *if* presented in a suitable manner.

Chapter 5

Conclusion

A compulsory chapter, of roughly 2 pages

The concluding chapter of a dissertation is often underutilised because it is too often left too close to the deadline: it is important to allocation enough attention. Ideally, the chapter will consist of three parts:

1. (Re)summarise the main contributions and achievements, in essence summing up the content.
2. Clearly state the current project status (e.g., "X is working, Y is not") and evaluate what has been achieved with respect to the initial aims and objectives (e.g., "I completed aim X outlined previously, the evidence for this is within Chapter Y"). There is no problem including aims which were not completed, but it is important to evaluate and/or justify why this is the case.
3. Outline any open problems or future plans. Rather than treat this only as an exercise in what you *could* have done given more time, try to focus on any unexplored options or interesting outcomes (e.g., "my experiment for X gave counter-intuitive results, this could be because Y and would form an interesting area for further study" or "users found feature Z of my software difficult to use, which is obvious in hindsight but not during at design stage; to resolve this, I could clearly apply the technique of Smith [7]").

Bibliography

[1] Alfred V. Aho and Margaret J. Corasick. Efficient string matching: An aid to bibliographic search. *Commun. ACM*, 18(6):333–340, June 1975.

Appendix A

An Example Appendix

Content which is not central to, but may enhance the dissertation can be included in one or more appendices; examples include, but are not limited to

- lengthy mathematical proofs, numerical or graphical results which are summarised in the main body,
- sample or example calculations, and
- results of user studies or questionnaires.

Note that in line with most research conferences, the marking panel is not obliged to read such appendices.