A Case Study of Practical and Theoretical Analysis of Priority Queues

My Final Project for Algorithms II, Tufts University

Cyrus Cousins

Spring 2014

1 Introduction

1.1 Purpose

The intended audience of this work is those who have studied algorithms or data structures before at the introductory level. Unfamiliar concepts may be introduced, but every effort is made to explain such concepts or provide reference materials.

The purpose of this document is to demonstrate the effectiveness of the complexity analysis software I have written, and to think about the analysis of some classic algorithms while we do so.

1.2 Architectural Overview

At its most basic level, the core of my software is a tree based representation of mathematical formulae, supporting all the traditional mathematical binary and unary operators, as well as constants, variables, and finite summations. Variable substitution for arbitrary expressions, simplifications, and calculation of bigO can all be performed on formulae in this system, making it very powerful and useful for algorithmic analysis. I am currently in the process of implementing basic differentiation and recurrence relationships, which should broaden the types of analysis the software is capable of.

In addition to the mathematical engine, the software loads a database of algorithms and abstract data type contracts, which can be analyzed and used to perform substitutions. Normally this is done without a GUI through a \LaTeX backend, which was how most of the documents in this presentation were created, but I have also created a GUI, which is useful for experimenting with different formulae, particularly in practical analysis. The text based component is primarily designed for creating formal documents and batch processes, such as performing many different substitutions, or applying combinations of subroutines to attempt to find an optimal time for an algorithm, and the graphical component is primarily for educational and entertainment purposes, though it is very useful for practical analysis as well.
2 Overview

Herein I show the program I have written to perform traditional theoretical analysis. The standard results for analysis of Dijkstra’s Algorithm with arrays, binary heaps, and Fibonacci heaps are obtained. The Dijkstra’s analysis may be viewed [here](#), and a formal introduction and analysis of each queue analyzed may be viewed [here](#).

Also analyzed are several naïve priority queue based partial sorting algorithms: these can be found [here](#). An applet demonstrating the practical analysis of these algorithms can be found [here](#).

It’s pretty difficult to define exactly what this software can be used for, but I believe the ability to automate the analysis process could:

1. Make automated error checking in textbooks and publications a possibility.
2. Make more intelligent educational tools that help students understand and solve problems, as well as generate unique problems.
3. Aid in writing books and papers by taking simple logical input and performing mathematical operations and typesetting it automatically. The documents I have produced with it certainly aim to do so, and it seems as though the same should be possible for more complicated results.

---

1The applet is very large, and I have found it to not work very consistently. The standalone downloadable application offers much better reliability.