1 Introduction

CIS 413/513 is a very theoretical and proof-oriented course, but data structures have a practical side as well. In order to be useful, data structures must be implemented in libraries with good interfaces. Once implemented, we can compare the actual running time of different methods. Methods that are asymptotically faster in theory may actually be slower in practice, given reasonably sized inputs.

In this project, you will explore interesting data structures in practice by implementing one, comparing it to alternatives experimentally, and preparing a 5-page report (more or less) based on your experience. The following sections describe the different aspects of the project.

2 Implementation

You must choose a data structure to implement and implement it in your choice of language. Almost any language is acceptable, as long as you can explain to me how to make your code run. Your implementation should expose a clear and easy-to-use interface to the user based on the standard operations supported by the data structure. You must also include a small example problem that uses the data structure, so I can see how it works.

Recommended data structures are Fibonacci heaps and splay trees, but you are free to propose another interesting structure (skip lists, kd-trees, union-find, etc.).

3 Experiments

To evaluate your data structure, you must design and run several experiments that compare its performance to alternative data structures. These experiments should be designed to showcase the relative strengths and weaknesses of your chosen data structure. Have fun with this!

For example, if your data structure is splay trees, then you could compare to Java’s TreeSet implementation. Experiments could compare the running time of inserting different numbers of elements. For example, you could show a graph of the running time as a function of the number of elements inserted. A second experiment could measure average look-up time for access frequencies that are more and less uniformly random. For instance, you could choose access frequencies that follow a power law and vary the exponent to get more and less skewed distributions. This could be repeated with different numbers of elements in the splay tree (10,000 vs. 10,000,000).

If your data structure is Fibonacci heaps, then you could compare to d-ary heaps with different values of d (either your own implementation or someone else’s). By varying the relative number of inserts and delete-mins, you should see different relative performance of the two methods. You could create additional experiments on the efficiency of meld and decrease-key to further showcase the strengths of Fibonacci heaps. Use your imagination!

4 Written Report

Your report should briefly describe the data structure and your experience of implementing it and give a detailed description of your experiments and results. You do not need to provide pseudocode of the data
structure or a detailed API reference.

Most of the report should be devoted to your experiments. Describe your methods in clear, plain language, such that another student could reproduce your experiments. Present your results and analyze them. What can you conclude? Are there any alternate explanations for your results? What are the limitations of your experiments? When would you, personally, use this data structure in practice (if ever)? Use line graphs to display trends. Use logarithmic scales where appropriate.

5 Proposal and Project Submission

Before doing the project, you must propose what it is you want to do. Please specify the data structure, the implementation language, the baseline data structures chosen, and what experiments you plan to run. This can be brief – one or two paragraphs. Please email me your proposal by Friday, November 11th.

The final project is due by Friday, December 2nd. You must include both your source code and your written report in the email.

6 Grading

This project is optional. If you complete this project, it will become 25% of your grade. This means that your normally computed grade (homework, tests, class participation) will be multiplied by 0.75, your project grade will be multiplied by 0.25, and the two will be added together to compute your final grade. Here is the breakdown of points for the project:

- Implementation: 30%. Points will be deducted for excessively messy code or unclear interfaces.
- Experiments: 40%. For full credit, your experiments should explore more than one dimension, such as varying number of inserts and number of delete-mins independently.
- Write-up: 30%. Points will be deducted for writing that is hard to understand and conclusions that are not well-supported by clear arguments. Total length: 5 pages, more or less.

I will not take off points for minor typo or grammar errors. With a reasonable effort, it should be possible for most students to get 100% on this project.