Project Propsal

Authors:
Joshua Hernandez [umhern23@myumanitoba.ca]

Paymahn Moghadasian [umpaymah@myumanitoba.ca]

Instructor: Dr. S. Durocher

COMP 4420
Advanced Design and Analysis of Algorithms

Due : Febuary 24, 2014

Project Proposal

Sorting is a well studied topic in algorithms, in particular quicksort has been extensivly studied. The quicksort is particularly interesting because it is a comparatively simple algorithm which has an average asymptotic runtime equal to that of the mergesort and heapsort without any additional memory requirements; quicksorts can be done in place without additional time or code complexity. A major drawback of the quicksort is the it has a worst case complexity of $O(n \log n)$ unlike the mergesort or heapsort[1].

Aumuller and Dietzfelbinger published a paper in ICALP 2013 titled "Optimal Partitioning for Dual Pivot Quicksort" where they explored the asymptotic runtime of various quicksort along with their respective partitioning algorithms[2]. Aumuller et al. mathematically demonstrated that dual pivot quicksorts are asymptotically lower bound by $1.8n \log n + o(n \log n)$ time where n is the number of elements to be sorted. They generalized the math to describe any dual pivot quicksort algorithm and verified the asymptotic lower bounds of the classic quicksort and Yaroslavskiy's variation[3].

We plan to first verify Aumuller's results and secondly to expand the scope of the original paper by mathematically describing all multi-partition variations. We will implement quicksort using various partitioning algorithms, again using varying number of partitions. We hope to determine if varying the number of paritions will yeild to a more efficent quicksort algorithm.

The goal of this project is to investigate multi-paritioned quicksort, determine an optimal number of parititions and bring to light any of the challenges that arise from the mathematical analysis of multi-partition quicksorts. We're particularly interested in determining whether having more than two parititons provides any theoretical or experimental benefits.

- Week 1 (Week of March 2 to 8)
  - Implement basic quicksort
    - Pivot Picking
      - first element as pivot
      - median of first, middle, and last
      - Find others
    - Partitioning algorithm
      - The basic one
    - Test functionaliy on small arrays
  - Implement 2 parition quicksort
    - Piviot picking
      - first and last elements

            - two middle elements of entires from 5 entries
            - Test with 4 entries as well
        * Paritioning Algorithm
            - Basic Paritioning
            - make smalls then bigs
            - flag paritioning algorithm
        * Test functionality on small arrays

- Week 2 (Week of March 9 to 15)
    - Implement 3 and 4 parition quicksorts (more if time allows)
        * Piviot picking
            - Basic Pivot selecion
            - Look and implement other piviot selection algorithms
        * Parition algorithm
            - Basic Parition
            - Look for parition algorithms
        * Test functionallity on small arrays

- Week 3 (Week of March 16 to 22)
    - Run several experinments using all the quicksort algoirthms implemented
    - Run several arrays sizes
    - Array 'types'
        * sorted arrays
        * reverse sorted arrays
        * random arrays
        * partially random arrays
    - Preliminary analysis of data

- Week 4 (Week of March 23 to 29)
    - Analyze data
    - Make Presentation

- Week 5 (Week of March 30 to April 5)
    - Continue Analyze data
    - Write paper

Bibliography

[1] R. Sedgewick, "The analysis of quicksort programs," *Acta Informatica*, vol. 7, no. 4, pp. 327–355, 1977. 1

[2] M. Aumüller and M. Dietzfelbinger, "Optimal partitioning for dual pivot quicksort," in *Proceedings of the 40th International Conference on Automata, Languages, and Programming - Volume Part I*, ICALP'13, (Berlin, Heidelberg), pp. 33–44, Springer-Verlag, 2013. 1

[3] S. Wild and M. E. Nebel, "Average case analysis of java 7's dual pivot quicksort," in *Proceedings of the 20th Annual European Conference on Algorithms*, ESA'12, (Berlin, Heidelberg), pp. 825–836, Springer-Verlag, 2012. 1