Shell Sort Code Documentation

Pritam Dey, Ayan Paul

March 2020

Contents

1 quick_sort.cpp

Performs Quick sort using Lomuto partitioning scheme.
Input: Array
Output: Numeric vector of the following format {number of swaps, number of comparisons}

2 quick_sort_wc.cpp

Performs Quick sort using Lomuto partitioning scheme
Input: Array
Output: 1

3 quick_sortH.cpp

Performs Quick sort using Hoare's partitioning scheme.
Input: Array
Output: Numeric vector of the following format {number of swaps, number of comparisons}

4 quick_sortH_wc.cpp

Performs Quick sort using Hoare's partitioning scheme
Input: Array
Output: 1

5 shell_sort.cpp

Performs Shell to the given array using the given gap sequence.
Input: Array, gap sequence
Output: Numeric vector of the following format {number of swaps, number of comparisons}

6 shell_sort_wc.cpp

Performs Shell to the given array using the given gap sequence.
Input: Array, gap sequence
Output: 1

7 gap sequences.R

This file contains the four functions for Shell sorts with four different gap sequences. This functions take only an array as input, then calls ShellSort function

that is implemented in Rcpp with the given array, and respective gap sequence. This functions will be used for further analysis.

8 plot dataset.R

This file contains script for loading data and plotting them.

9 SortPlot Comps.R

This file contains script for computing number of comparisons (swaps/assignments) made by different algorithms. There are two main functions in this file.

9.1 avg1

This function returns scaled version of average number of comparisons (Assignments/Swaps) for different algorithms specified in the input.
Input:

- funcs: List of sorting algorithms that will be applied to randomized arrays.
- r1: Number of random arrays to be generated for sorting.
- l: Length of the random array

Output: $\frac{average\ number\ of\ comparisons(swaps/assignments)}{log_2 N!}$

9.2 sort_plot1

This function plots scaled version of average number of comparisons (Assignments/Swaps) for different algorithms specified in the input.
Input:

- seqn: Sequence of numbers that determines different length of arrays to be generated in the simulation.
- funcs: List of sorting algorithms that will be applied to randomized arrays.
- r1: Number of random arrays to be generated for sorting.

Output: plots $\frac{average\ number\ of\ comparisons(swaps/assignments)}{log_2 N!}$ against length of array sizes in log scale.
Also saves the data generated in the simulation in a folder named "data", with the name of the file in a special format.

10 SortPlot Time.R

This file contains script for computing average running time for sorting by different algorithms. There are two main functions in this file.

10.1 avg

This function returns scaled version average running time for different algorithms specified in the input.
Input:

- funcs: List of sorting algorithms that will be applied to randomized arrays.
- r1: Number of random arrays to be generated for sorting.
- r2: Number of times each array to be sorted by a particular algorithm.
- l: Length of the random array

Output: $\frac{average\ running\ time}{NlogN}$

10.2 sort_plot

This function plots scaled version average running time for different algorithms specified in the input
Input:

- seqn: Sequence of numbers that determines different length of arrays to be generated in the simulation.
- funcs: List of sorting algorithms that will be applied to randomized arrays.
- r1: Number of random arrays to be generated for sorting.
- r2: Number of times each array to be sorted by a particular algorithm.

Output: plots $\frac{average\ running\ time}{NlogN}$ against length of array sizes in log scale. Also saves the data generated in the simulation in a folder named "data", with the name of the file in a special format.