CS 014: Introduction to Data Structures and Algorithms
Fall 2017
Lab 5

Objectives

In this lab, you will implement basic sorting algorithms and compare their running time.

Deliverables

- (10%) You have to attend the lab on Monday.
- (65%) The final source code.
- (5%) You must adhere to the following submission format. You need to submit a single file on iLearn named "CS014_lab4_<UCR Net IDs>.zip" where <UCR Net IDs> are the students' UCR Net IDs separated by underscore. The ZIP file should contain the source code in .cc, .hh, or whatever extension for source files you. The ZIP file should also contain a PDF report that contains the filled-in table (see below) and the log-log chart for the running times of the algorithms. On the cover page, your report should mention:
    - Your TA name.
    - Your lab section number.
    - You name and UCR Net ID
- (20%) Report the final running times in a table and plot a log-log chart.
- Note: No in-lab submission is required for this lab

Groups

- This lab will be done individually.

Due date

- The deliverables are due on Tuesday 11/07 by 11:59 PM Pacific Time. However, you are highly encouraged to deliver it during the lab on Monday 11/06 to save your time.

Problem definition

We studied several sorting algorithms in class. In this lab you will implement a few of these algorithms and compare their running times for various input sizes. The algorithms we will consider are bubble sort, insertion sort, selection sort, and Shell sort.

Steps

1. Create a workspace in Cloud9 and initialize it with the code that is attached to this lab. You can do this by pasting the URL 'https://github.com/aseldawy/CS014-Lab5.git' in the field 'Clone from

Git or Mercurial URL.' Instead, you can download the source code from iLearn and upload it to an empty workspace in Cloud9.

2. To compile the code, type "make" in the shell. In Cloud9, you can access the shell which is usually at the bottom of the screen.
3. To run the code, type "./sorting" in the shell.
4. The code has two parts, the first part tests the correctness of the sorting algorithm and the second part measures the actual running time as the array size changes.
5. The code that you need to implement is in the five functions named 'bubbleSort', 'insertionSort', 'selectionSort', 'shellSort1', and 'shellSort2'. You are required to implement the first four functions. The fifth function 'shellSort2' is optional. If you do it correctly, you will get an additional 10% bonus for this lab.
6. You can find more details about each algorithm in the course slides, the textbook, or the Internet.
7. What is the worst-case asymptotic running time for each of the algorithms you implemented? You can refer to the textbook or the slides for the running time of the Shell sort.
8. After making sure that your implementations of the sorting algorithms work correctly, collect the running times and fill in the table below.

Array size	Bubble Sort	Insertion Sort	Selection Sort	Shell Sort 1	Shell Sort 2
1					
2					
4					
8					
16					
32					
64					
128					
256					
512					
1024					
2048					
4096					
8192					
16384					
32768					
65536					

9. The final step is to plot a log-log graph where the array size is on the x-axis and the total running time is on the y-axis.
10. Comment on the behavior of the different algorithms as the input size increases and how it is related to the asymptotic running time of each algorithm.