COP 4531 - Assignment 2

Due Date - 10/01/2015

September 24, 2015

Project Description

In this assignment you will implement several algorithms for sorting positive integers, and compare their running times.

You are asked to implement algorithms with different worst case running times.

1. **Insertion sort**, which is $\Theta(n^2)$.
2. One of **Mergesort or Heapsort**, which are $\Theta(n \log n)$.
3. **Counting sort**, which is $\Theta(n + k)$, where all values to be sorted are between 1 and k. **Please implement a linear scan** to proceed counting sort in order to identify k, the largest value in the input.

The goal of the assignment is to study the practical implications of asymptotic running times. To this end, you will conducting two experiments, described below.

Experiments

Prepare a report no longer than 4 pages. In the report, please mention which linearithmic algorithm you implemented. The main part of the report is a summary of two experiments comparing the running times of the three algorithms you implement.

Experiment 1:

Run your algorithms on progressively larger data sets, recording the running times of your solution. The chart should be similar to Slide 8 on the first set of course notes posted on the course webpage. It should have similar format to the table below, indicating actual running times (in seconds, milliseconds, etc) in the cells of the table:

	$\Theta(n \log n)$ algorithm	$\Theta(n^2)$ algorithm
n= 10		
n= 100		
n=1000		
n=10,000		
n=100,000		
n=1,000,000		

Please note that the above values of n are examples, and you may choose your own values, as long as the experiment successfully investigates how the algorithms behave as n grows. Note that when your program

takes too long, you may simply indicate that the running time was above a certain threshold, as seen on Slide 8 of the first set of course notes.

In addition to including the charts, briefly discuss the findings of your experiment in the report.

0.1 Experiment 2:

This time we will be comparing the output of your program based on two parameters, both n (the number of elements to be sorted) and k (representing the largest integer in the input array).

Each chart will correspond to a fixed value of n while varying the values of k. Prepare at least two such charts, one with a small value of n (≤ 100) and one with a large value of n ($\geq 10,000$).

For example, the following chart considers data sets with $n = 100$ and varying values of k.

	$\Theta(n+k)$ alg	$\Theta(n \log n)$ alg	$\Theta(n^2)$ alg
$k = 2$			
$k = 5$			
$k = 10$			
$k = 200$			
$k = 500$			
$k = 5000$			

Please note that the above values of n and k are examples, and you may choose your own values, as long as the experiment successfully compares the three algorithms you implemented on different values of n and k. **In addition to including the charts, briefly discuss your findings in the report.**

Input and output format

We will test your data on .txt files where the first column will contain positive integers and the second column will contain strings of length at most 20 (you may assume that the strings in the second column are composed entirely of lower and upper case letters, with no spaces).

Here is an example:

7 Alex
3 Maya
8 Steve
87 David
2 Mike

Your program should then sort the data based on the first column and output the results to the screen:

2 Mike
3 Maya
7 Alex
8 Steve
87 David

Given a text file "data.txt," we will run your algorithms as follows:

- insertionsort data.txt
- mergesort data.txt
- heapsort data.txt
- countingsort data.txt

Please note that you need to implement only one of MergeSort or HeapSort. Further, if the programming language you use necessitates the use of alternate command lines, please note that at the beginning of your README file.

Programming languages and formatting

You are free to use any programming language and library available on linprog. However, you should implement the algorithms yourself and NOT use any library implementation of the algorithms. Your code must be able to be compiled and run in linprog, and if it cannot, it will be considered broken. As a reminder, please do not copy code from online or your classmates. You may use the pseudocode provided in class to design your solutions.

Submission Guidelines

Your submission must have the following:

- A printed report (up to 4 pages) with the results of your two experiments and a brief discussion of your findings. **Please submit your report at the start of class on Tuesday, September 29th.**
- A README file that describes how the code can be compiled and run. Also list any external dependencies that need to be satisfied for compiling and running the code.
- A Makefile that can be used for compilation. Please note that the TAs will not write Makefiles for compiling code or write any commands except for single word commands like **make** for compiling.
- Your submission should have a single point of entry for running the code, like a main() method in C or its equivalent in any other language.
- **Please e-mail your submission to ssk09c@my.fsu.edu by 3:30 PM on Thursday September 29, 2015**. DO NOT SUBMIT THROUGH DROPBOX. They will not be accepted and will result in late penalty. **Don't forget to bring a hard copy of your report to the start of class on the 29th.**

Mark Breakdown

- Report with two experiments [30 points]
- Design of program [20 Points]
- Error free compilation [5 Points]
- Correctness on supplied data and additional tests [45 Points]

Bonus Contest (up to 25 additional points)

Design and implement a sorting algorithm with efficient worst case solution in terms of both n and k. We will test your algorithm on large data sets with varying values of n and k, and the best implementation will receive 25 bonus point (all excellent solutions stand to earn some bonus points). **Add a description of your algorithm to the report.**

To run the bonus on a file named "data.txt," we will use the following command:
bonus data.txt

HINT: Your algorithm may consist of a linear scan of the data, which is then used to select among several established sorting methods.

1 Test cases

Please go to

`http://www.cs.fsu.edu/~ackerman/Fall2015/Assignments/asgn2.tar.bz2`