BLG381E, Advanced Data Structures, Fall 2012

Project 1

Handed out: 10.10.2012

Due: 23.10.2012, until 5.00 PM

Problem: Assume that you have N homework assignments to complete, where each assignment A_i requires T_i hours of time for completion. You want to make a schedule for these assignments by using the shortest job next (SJN) rule, where the scheduling is done with respect to increasing order of processing time.

For example (N=5):

A_i	T_i
1	3
2	5
3	2
4	4
5	2

using the shortest job next rule

A_i	T_i
3	2
5	2
1	3
4	4
2	5

In this project, you will implement two sorting algorithms as C++ methods:

a) Insertion sort
b) Merge sort

In your main program, number of homework assignments (N) will be taken from the user and N random numbers will be generated for the completion time of each homework assignment to build an $Nx2$ matrix like the one given above. Then, the rows of the matrix will be sorted with respect to SJN rule using your implementation of insertion sort and merge sort (the user will select which sorting method to be used by entering 'I' or 'M' from the keyboard), and the resulting schedule matrix will be printed on the screen.

Detailed Instructions:

- All your code must be written in C++ using object oriented approach and able to compile and run on linux using g++.
- Submissions will be done through the Ninova server. You must submit all your program and header files. You must also submit a softcopy report.
- In your report, you are expected to analyze and compare the running times of two sorting algorithms with respect to their computational complexity.
  - ***a.*** Give the asymptotic upper bound on the running time for both insertion sort and merge sort (which you can find in the lecture slides) and show that your implementation of these algorithms fit these values.
  - ***b.*** Run each two search methods for 10 times for each different value of N as $\{10, 20, 50, 100, 1000, 5000, 10000\}$ and calculate the average time of execution for each value of N.
    Note: You can use the clock() function under ctime library for calculating time of execution for your search functions. Refer to http://www.cplusplus.com/reference/clibrary/ctime/clock for more details.
  - ***c.*** After calculating execution times you will prepare two line plots in Excel in order to visualize the runtime complexity of merge sort and insertion sort for different values of N. Then you will interpret the results with respect to the asymptotic upper bounds you have given in ***a***.