Shiv Nadar University

CSD101: Introduction to Computing and Programming

Lab #10

Time complexity and actual time taken

Max marks:80
Due on/before:22.00, 8-Nov-2021. 31-Oct-2021

1. In this lab you will check the actual time taken by different sorting programs for different sized sequences populated randomly. Do this as follows:

    1. Implement Bubble sort, selection sort, quick sort and merge sort as functions. Code for most of these is already available on the course website, on the 'In class' page. You may have to make minor changes to each program to use it for this assignment.
    2. For each algorithm you should generate sequences of length 10, 100 and 1000 populated by random numbers. See below for how to do this.
    3. Run each algorithm on each sequence and store the time taken by each algorithm. See below for how execution time can be estimated.
    4. Repeat the previous two steps 1000 times and find the average time taken by each algorithm for each size.
    5. Report your results in the form of a 4×3 table for the 4 algorithms and the 3 sequence sizes.
    6. Also report which algorithm is fastest for each size.

    You should also observe the differences between theoretical time complexity and actual time taken by the different algorithms. Recall that bubble sort and selection sort have worst case and average case complexities of $\mathcal{O}(n^2)$. Quick sort has a worst case complexity of $\mathcal{O}(n^2)$ and an average case complexity of $\mathcal{O}(n\,log(n))$. Merge sort has a complexity of $\mathcal{O}(n\,log(n))$ for both worst and average cases.

Generating random numbers

```
//Requires <stdlib.h> and <time.h>
#include <stdlib.h>
#include <time.h>

int randNo;
/* srand() seeds the random generator. This should be done only once in
the program. This ensures that the same sequence of random numbers is not generated every time the pr
srand(time(0));

//Every time rand() is executed a random integer is generated.
randNo = rand();//generates a random integer
```

Estimating execution time

```
//Requires <time.h>
#include <time.h>

double execTime;
clock_t startTime, endTime;
//Code that is not to be timed

startTime=clock()

//Code whose execution time is to be measured

endTime=clock()
execTime=(double)(endTime-startTime)/CLOCKS_PER_SEC
/* execTime will be the execution time in secs for the code whose
   execution time is to be measured.
   Note that you should measure exec time only for the sorting code
   and not for random sequence generation.
*/
```

[80]