

Department of Computer Science and Engineering, SVNIT, Surat

MTech I (1st semester)

Design and Analysis of Algorithms

Lab Assignment No 2 - Divide and Conquer Design Technique

AUTUMN Semester 2022-23

Dated Uploaded: 18th Nov 2023

Instructions:

1. The date of submission **will be specified only two days before**. Therefore, right from the time the assignment is uploaded the students must start implementing the assignments.
2. For every delayed submission beyond the deadline, 10 marks per day will be deducted from the maximum marks of the assignment, without any exception, whatsoever may be the scapegoat.
3. You can use any programming language, except Python.
4. For every program whether specified or not, it is necessary to time your program on input dataset of a large size and give a critique of the theoretical asymptotic analysis of the algorithm that your program is based on and the empirical timing analysis of the program that you have written. The a-priori estimates and the a-posteriori analysis must be in sync with each other.
5. All the assignments must be submitted in the form of a zip file containing the Program Source, the screenshot of the output that you obtained, the DataSet/Input Test data used and a ReadMe .txt file explaining what platform to use, what are the input parameters required for execution and how to execute it. Also write your conclusion in ReadMe file. For the theoretical questions in Part B, write the answers using Latex in a .Tex file and submit the .tex as well as .pdf files - to be included in the zip file as above.
6. Perform usual error checking. Don't go overboard on this, but don't let your program die because of divide by zero.
7. Remember, your programs could be checked by a code-cheating program, so please follow the code of academic integrity.
8. There will be a viva for each assignment. This viva would be conducted for this assignment on a future date as specified.
9. **Maximum Points: 100**

-
1. Design a divide-and-conquer algorithm for computing the number of levels in a binary tree. (In particular, the algorithm must return 0 and 1 for the empty and single-node trees, respectively.) Analyze the time efficiency class of your algorithm. Now, implement the algorithm you designed and find out the time taken for empty, one-level, two-level and three-level binary tree. Compare it with the theoretical analysis that you would have done before.
 2. Write programs to count the number of inversions in a given array using the brute force approach and using the algorithm applying the merge-sort discussed in the class (with time complexities $O(n^2)$ and $O(n \lg n)$ respectively). Compare both the programs with timing analysis with input size starting with $n = 10$ upto $n = 10^5$. Comment on the results obtained.
 3. Observe that the algorithm PARTITION() for Quicksort given in the text is different from the one discussed in the class. Implement two different Quicksort routines that use the two different PARTITION routines. Test your implementation on an input vector of size at least a 2048 integers and time the programs. Analyze the result and document it.
 4. Compare the times in executing the programs in problems at Sr nos 3,4,5 and analyze your results. Give appropriate inferences.
 5. Implement the Counting sort covered in the class and analyze the running time of your code.